Altova UModel 2023 Basic Edition

Benutzer- und Referenzhandbuch

Altova UModel 2023 Basic Edition
Benutzer- und Referenzhandbuch

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Published: 2022

© 2016-2022 Altova GmbH

Inhaltsverzeichnis

1 Einführung 10

1.1 Anmerkungen zur Unterstützung 11

2 UModel Tutorial 14

2.1 Erste Schritte 15
2.2 Use Cases 16
2.3 Klassendiagramme 25
2.3.1 Erstellen von abgeleiteten Klassen 34
2.4 Objektdiagramme 40
2.5 Komponentendiagramme 48
2.6 Deployment-Diagramme 54
2.7 Forward Engineering (Modell zu Code) 59
2.8 Reverse Engineering (Code zu Modell) 69

3 Grafische Benutzeroberfläche von UModel 78

3.1 Fenster "Modell-Struktur" 80
3.2 Fenster "Diagramm-Struktur" 84
3.3 Fenster "Favoriten" 85
3.4 Fenster "Eigenschaften" 86
3.5 Fenster "Stile" 87
3.6 Fenster "Hierarchie" 88
3.7 Fenster "Übersicht" 90
3.8 Fenster "Dokumentation" 91
3.9 Fenster "Meldungen" 92
3.10 Diagrammfenster 94
3.11 Diagrammbereich 95

4 UModel Befehlszeilenschnittstelle 97

4.1 Datei: New / Load / Save-Optionen ... 102

5 Anleitung zur Modellierung von... 104

5.1 Elemente ... 106

5.1.1 Erstellen von Elementen ... 106

5.1.2 Einfügen von Elementen aus dem Modell in ein Diagramm ... 107

5.1.3 Umbenennen, Verschieben und Kopieren von Elementen ... 109

5.1.4 Löschen von Elementen ... 110

5.1.5 Konvertieren von Elementen ... 111

5.1.6 Suchen und Ersetzen von Text ... 111

5.1.7 Überprüfen, wo und ob Elemente verwendet werden ... 113

5.1.8 Einschränken von Elementen ... 114

5.1.9 Erstellen von Hyperlinks zu Elementen ... 115

5.1.10 Dokumentieren von Elementen ... 118

5.1.11 Ändern des Stils von Elementen ... 119

5.2 Diagramme ... 122

5.2.1 Erstellen von Diagrammen ... 122

5.2.2 Generieren von Diagrammen ... 123

5.2.3 Öffnen von Diagrammen ... 125

5.2.4 Löschen von Diagrammen ... 126

5.2.5 Ändern des Stils von Diagrammen ... 127

5.2.6 Ausrichten von Modellierungselementen und Anpassen der Größe ... 128

5.2.7 Typ-Autokomplettierung in Klassen ... 130

5.2.8 Vergrößern und Verkleinern von Diagrammen ... 132

5.3 Beziehungen ... 133

5.3.1 Erstellen von Beziehungen ... 133

5.3.2 Ändern des Stils von Linien und Beziehungen ... 134

5.3.3 Anzeigen von Elementbeziehungen ... 136

5.3.4 Assoziationen ... 137

5.3.5 Collection-Assoziationen ... 140

5.3.6 Enthältbeziehung ... 143

5.4 Stereotype und Eigenschaftswerte ... 145
5.4.1 Eigenschaftswerte ... 146
5.4.2 Anwenden von Stereotypen ... 147
5.4.3 Anzeigen oder Ausblenden von Eigenschaftswerten ... 149

6 Projekte und Code Engineering 152

6.1 Verwalten von UModel-Projekten ... 153
6.1.1 Erstellen, Öffnen und Speichern von Projekten ... 153
6.1.2 Öffnen von Projekten über eine URL ... 154
6.1.3 Verschieben von Projekten in ein neues Verzeichnis ... 158
6.1.4 Anwenden von UModel-Profilen ... 160
6.1.5 Aufteilen von UModel-Projekten ... 160
6.1.6 Inkludieren von Unterprojekten ... 164
6.1.7 Freigeben von Paketen und Diagrammen ... 166
6.1.8 Verbesserung der Performance ... 169
6.2 Generieren von Programmcode ... 170
6.2.1 Definieren eines Pakets als Namespace Root ... 170
6.2.2 Hinzufügen einer Code Engineering-Komponente ... 171
6.2.3 Überprüfen der Projektsyntax ... 173
6.2.4 Codegenerierungsoptionen ... 176
6.2.5 Beispiel: Generieren von C#-Code ... 177
6.2.6 Beispiel: Generieren von Java-Code ... 182
6.2.7 SPL-Vorlagen ... 192
6.3 Importieren von Quellcode ... 194
6.3.1 Optionen für den Code-Import ... 196
6.3.2 Beispiel: Importieren eines C#-Projekts ... 199
6.4 Importieren von Java-, C#- und VB.NET-Binärdateien ... 206
6.4.1 Hinzufügen von benutzerdefinierten Java Runtimes ... 207
6.4.2 Optionen für den Import von Binärdateien ... 208
6.4.3 Beispiel: Import von .NET Assemblys ... 211
6.4.4 Beispiel: Import von Java .class-Dateien ... 214
6.5 Synchronisieren von Modell und Quellcode ... 221
6.5.1 Tipps zur Synchronisierung ... 222
6.5.2 Refactoring und Synchronisierung von Code ... 224

6.5.3 Codesynchronisierungseinstellungen ... 225
6.6 UModel-Elemententsprechungen ... 228
6.6.1 C#-Entsprechungen ... 228
6.6.2 VB.NET-Entsprechungen ... 248
6.6.3 Java-Entsprechungen ... 263
6.6.4 XML Schema-Entsprechungen ... 268
6.7 Zusammenführen von UModel-Projekten ... 280
6.7.1 3-Weg-Projektzusammenführung ... 281
6.7.2 Beispiel: Manuelle 3-Weg-Projektzusammenführung ... 283
6.8 UML-Vorlagen ... 286
6.8.1 Vorlagensignaturen ... 287
6.8.2 Vorlagenverwendung ... 288
6.8.3 Vorlagenverwendung in Operationen und Eigenschaften ... 288

7 Generieren von UML-Dokumentation 290

7.1 Optionen zur Generierung von Dokumentation ... 294
7.2 Anpassen der Ausgabe mit StyleVision ... 299

8 UML-Diagramme 301

8.1 Verhaltensdiagramme ... 302
8.1.1 Aktivitätsdiagramm ... 302
8.1.2 Zustandsdiagramm ... 319
8.1.3 Protokoll-Zustandsautomat ... 342
8.1.4 Use Case-Diagramm ... 347
8.1.5 Kommunikationsdiagramm ... 347
8.1.6 Interaktionsübersichtsdiagramm ... 351
8.1.7 Sequenzdiagramm ... 356
8.1.8 Zeitverlaufsdiagramm ... 385
8.2 Strukturdiagramme ... 394
8.2.1 Klassendiagramm ... 394
8.2.2 Kompositionsstrukturdiagramm ... 410
8.2.3 Komponentendiagramm ... 413
8.2.4 Deployment-Diagramm ... 413

8.2.5 Objektdiagramm ... 414
8.2.6 Paketdiagramm ... 414
8.2.7 Profildiagramm ... 421
8.3 Zusätzliche Diagramme ... 435
8.3.1 XML-Schema-Diagramme ... 435

9 Austausch von Metadaten zwischen XMI und XML 454

10 Versionskontrolle 456
10.1 Einrichten der Versionskontrolle ... 458
10.2 Unterstützte Versionskontrollsysteme ... 459
10.3 Versionskontrollbefehle ... 461
10.3.1 Aus Versionskontrolle öffnen ... 461
10.3.2 Versionskontrolle aktivieren ... 464
10.3.3 Aktuellste Version holen ... 465
10.3.4 Abrufen ... 465
10.3.5 Ordner abrufen ... 466
10.3.6 Auschecken ... 468
10.3.7 Einchecken ... 469
10.3.8 Auschecken rückgängig ... 470
10.3.9 Zu Versionskontrolle hinzufügen ... 471
10.3.10 Von Versionskontrolle ausgliedern ... 474
10.3.11 Aus Versionskontrolle freigeben ... 475
10.3.12 Verlauf anzeigen ... 476
10.3.13 Unterschiede anzeigen ... 478
10.3.14 Eigenschaften anzeigen ... 479
10.3.15 Status aktualisieren ... 480
10.3.16 Versionskontrollmanager ... 480
10.3.17 Versionskontrolle wechseln ... 480
10.4 Versionskontrolle mit Git ... 482
10.4.1 Aktivieren der Git-Versionskontrolle mit dem Git SCC Plug-in ... 483
10.4.2 Hinzufügen eines Projekts zur Git-Versionskontrolle ... 483
10.4.3 Klonen eines Projekts anhand der Git-Versionskontrolle ... 485

11 UModel Diagrammsymbole 487

11.1 Aktivitätsdiagramm 488
11.2 Klassendiagramm 489
11.3 Kommunikationsdiagramm 490
11.4 Kompositionsstrukturdiagramm 491
11.5 Komponentendiagramm 492
11.6 Deployment-Diagramm 493
11.7 Interaktionsübersichtsdiagramm 494
11.8 Objektdiagramm 495
11.9 Paketdiagramm 496
11.10 Profildiagramm 497
11.11 Protokoll-Zustandsdiagramm 498
11.12 Sequenzdiagramm 499
11.13 Zustandsdiagramm 500
11.14 Zeitverlaufsdiagramm 501
11.15 Use Case-Diagramm 502
11.16 XML-Schema-Diagramm 503

12 Menüreferenz 504

12.1 Datei 505
12.2 Bearbeiten 507
12.3 Projekt 509
12.4 Layout 512
12.5 Ansicht 514
12.6 Extras 515
12.6.1 Benutzerdefinierte Tools 515
12.6.2 Anpassen 515
12.6.3 Symbolleisten und Fenster wiederherstellen 524
12.6.4 Optionen 525
12.7 Fenster 536
12.8 Hilfe 538

13 SPL-Referenz 543

13.1 Grundlegende SPL-Struktur 544
13.2 Variablen 545
13.3 Operatoren 554
13.4 Bedingungen 555
13.5 Collections und foreach 556
13.6 Subroutinen 558
13.6.1 Deklaration einer Subroutine 558
13.6.2 Subroutinenaufruf 559

14 Lizenzinformationen 560

14.1 Electronic Software Distribution 561
14.2 Software-Aktivierung und Lizenzüberwachung 562
14.3 Altova Endbenutzer-Lizenzvereinbarung 564

Index 565

1 Einführung

Altova Web site: UML-Tool

Altova UModel 2023 Basic Edition ist eine UML-Modellierapplikation mit visueller Benutzeroberfläche und zahlreichen, benutzerfreundlichen Funktionen, mit denen das Erlernen von UML kein Problem mehr ist. UModel bietet viele hochspezifische Funktionen für die Implementierung der nützlichsten Aspekte der UML 2.5-Spezifikation. UModel ist eine 32/64-Bit Windows-Applikation, die auf Windows 7 SP1 mit Plattformupdate, Windows 8, Windows 10, Windows 11 und Windows Server 2008 R2 SP1 mit Plattformupdate oder höher läuft. Die 64-Bit-Version steht für die Enterprise und Professional Version zur Verfügung. Eine Übersicht über die UModel-Funktionen findne Sie unter Anmerkungen zur Unterstützung 11.

UML®, OMG™, Object Management Group™ und Unified Modeling Language™ sind entweder eingetragene Markenzeichen oder Markenzeichen der Object Management Group, Inc. in den USA und/oder anderen Ländern.

Letzte Aktualisierung: 06.10.2022

 © 2016-2022 Altova GmbH

1.1 Anmerkungen zur Unterstützung

UModel ist eine 32/64-Bit-Windows-Applikation, die auf den folgenden Betriebssystemen läuft:

- Windows Server 2008 R2 SP1 mit Plattformupdate oder höher
- Windows 7 SP1 mit Plattformupdate, Windows 8, Windows 10, Windows 11

64-Bit-Unterstützung steht für die Enterprise und die Professional Edition zur Verfügung.

UML-Diagramme

UModel unterstützt alle vierzehn Diagramme der UML 2.5-Spezifikation sowie zusätzliche spezielle Diagrammarten.

Strukturdiagramme	Verhaltensdiagramme	Zusätzliche Diagrammarten
Klassendiagramme	Aktivitätsdiagramm	XML-Schema-Diagramme
Komponentendiagramm	Kommunikationsdiagramm	BPMN (Business Process Modeling Notation) 1.0 / 2.0-Diagramme (*UModel Enterprise und Professional Edition*)
Kompositionsstrukturdiagramm	Interaktionsübersichtsdiagramm	SysML 1.2-, 1.3-, 1.4-, 1.5-, 1.6-Diagramme (*UModel Enterprise und Professional Edition*)
Deployment-Diagramm	Sequenzdiagramm	Datenbankdiagramme (*UModel Enterprise und Professional Edition*)
Objektdiagramm	Zustandsdiagramme (Zustandsdiagramm und Protokoll-Zustandsdiagramm)	
Paketdiagramm	Zeitverlaufsdiagramm	
Profildiagramm	Use Case-Diagramm	

UModel wurde so konzipiert, dass der Benutzer bei der Modellierung vollkommen freie Hand hat:

- UModel-Diagramme können jederzeit und in jeder Reihenfolge erstellt werden; es muss bei der Modellierung keine vorgeschriebene Reihenfolge eingehalten werden.
- Die Syntaxfarbe in Diagrammen kann angepasst werden. So können Sie etwa Modellierungselemente und deren Eigenschaften (Schriftart, Farbe, Umrandung, usw.) auf hierarchische Weise auf Projekt-, Node/Zeilen-, Elementfamilien- und Elementebene anpassen, siehe Ändern des Stils von Elementen 119.
- Die Funktion zum unbegrenzten Rückgängigmachen und Wiederherstellen hält nicht nur Änderungen am Inhalt fest, sondern auch alle Änderungen am Stil von Modellelementen.
- Modellierungselemente unterstützen Hyperlinks, siehe Erstellen von Hyperlinks zu Elementen 115.

Code Engineering und Import von Binärdateien

UModel unterstützt die Codegenerierung und das Reverse Engineering von in den folgenden Sprachen geschriebenem Programmcode:

Sprache	Code Engineering	Import von Binärdateien
C#	1.2, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 7.1, 7.2, 7.3, 8.0, 9.0[1], 10	Dieselben Sprachversionen wie beim Code Engineering[2]
C++ (*UModel Enterprise Edition*)	C++98, C++11 und C++14, C++17, C++20 Nur teilweise Unterstützung für C++20: Module werden nicht unterstützt.	Nicht anwendbar
Java	1.4, 5.0 (1.5), 6 (1.6), 7 (1.7), 8 (1.8), 9 (1.9), 10, 11, 12, 13, 14, 15, 16, 17, 18, 19	Dieselben Sprachversionen wie beim Code Engineering[3]
Visual Basic .NET	7.1 oder neuer	Dieselben Sprachversionen wie beim Code Engineering
XML-Schemas[4]	1.0	Nicht anwendbar
Datenbanken[5] (*UModel Enterprise und Professional Edition*)		Nicht anwendbar

Fußnoten zur Tabelle:

1. Wenn Sie anhand von C# 9.0-Code kompilierte Binärdateien importieren, werden alle *Datensätze* als *Klassen* importiert. Dies liegt daran, dass Datensätze in der Assembly als Klassen gekennzeichnet sind, wodurch sie von Klassen nicht unterschieden werden können.
2. Beim C# Code Engineering und Import von Binärdateien werden NET Framework, .NET Core, .NET 5 und .NET 6 unterstützt. Beachten Sie, dass, je nach Bedarf, .NET Framework, .NET Core, .NET 5 oder .NET 6 installiert sein muss. Auch Binärdateien anderer hier nicht erwähnter .NET-Implementierungen werden wahrscheinlich ebenfalls importiert. Siehe auch Importieren von Java-, C#- und VB.NET-Binärdateien 206.
3. Auch Binärdateien für andere Java Virtual Machines als Oracle JDK wie OpenJDK, SapMachine, Liberica JDK und andere können importiert werden, siehe Hinzufügen von benutzerdefinierten Java Runtimes 207.
4. Im Fall von XML-Schemas bedeutet Code Engineering, dass Sie ein Schema (oder mehrere Schemas aus einem Verzeichnis) in UModel importieren, das Modell anzeigen oder ändern und die Änderungen wieder in die Schema-Datei schreiben können. Wenn Sie Daten aus dem Modell in einer Schema-Datei synchronisieren, wird die Schema-Datei immer durch das Modell überschrieben. Siehe auch XML-Schema-Diagramme 435.
5. Im Fall von Datenbanken bedeutet Code Engineering, dass Sie (i) eine Datenbank in UModel modellieren können mit der Option, die Datenbank mit Hilfe eines anhand des Modells generierten Skripts zu aktualisieren oder (ii) eine bestehende Datenbankstruktur in ein Modell importieren, Änderungen daran vornehmen und anschließend ein anhand des Modells generiertes Skript an der Datenbank anwenden können. Einige Datenbankobjekttypen werden beim Modellieren nicht unterstützt

 © 2016-2022 Altova GmbH

Allgemeine Anmerkungen:

- Die Synchronisierung zwischen Code und Modell kann auf Ebene eines Projekts, Pakets oder sogar Klassen erfolgen. Bei UModel müssen kein Pseudocode oder Kommentare im generierten Code vorhanden sein, damit ein Round-Trip Engineering durchgeführt werden kann.
- Ein einziges Projekt kann gleichzeitig sowohl Java-, C#- oder VB.NET-Code unterstützen.
- UModel unterstützt die Verwendung von UML-Vorlagen und deren Mapping von oder auf generische Java-, C#- und Visual Basic-Typen.
- Beim Import von Quellcode können optional Klassen- 407 und Paketdiagramme 418 generiert werden. Nachdem der Quellcode in das Modell importiert wurde, können auch Sequenzdiagramme 371 generiert werden.
- Sie können Programmcode anhand von Sequenzdiagrammen 378 und Zustandsdiagrammen 331 generieren.
- UModel-Projekte können in mehrere Unterprojekte aufgeteilt werden, sodass mehrere Entwickler gleichzeitig an verschiedenen Teilen eines einzigen Projekts arbeiten können. Sie können die Änderungen anschließend wieder in einem gemeinsamen Modell miteinander integrieren. Außerdem können UModel-Projekte in Form einer 2- oder 3-Weg-Zusammenführung miteinander zusammengeführt werden, siehe Zusammenführen von UModel-Projekten 280.
- Die Codegenerierung in UModel basiert auf Spy Programming Language (SPL)-Vorlagen und kann angepasst werden.

Generierung von UML-Dokumentation

Sie können anhand von UModel-Projekten Dokumentation in HTML, RTF, Microsoft Word 2000 oder höher generieren. Zur Konfiguration des Detailliertheitsgrads der generierten Dokumentation, des Aussehens und anderer Einstellungen stehen verschiedene Optionen zur Verfügung. Die Generierung von Dokumentation im PDF-Format und eine genaue Anpassung der Vorlagen für die Dokumentgenerierung kann mit Hilfe von Altova StyleVision (https://www.altova.com/de/stylevision) durchgeführt werden. Nähere Informationen dazu finden Sie unter Generieren von UML-Dokumentation 290.

Interoperabilität

Außerdem bietet UModel Unterstützung für den Import oder Export von Projekten von oder in das XML Metadata Interchange (XMI)-Format, siehe Austausch von Metadaten zwischen XMI und XML 454.

2 UModel Tutorial

In diesem Tutorial wird beschrieben, wie Sie verschiedene UModel-Diagramme mit UModel erstellen, während Sie gleichzeitig mit der grafischen Benutzeroberfläche von UModel vertraut gemacht werden. Außerdem lernen Sie, wie Sie anhand eines UML-Modells Code generieren (Forward Engineering) und wie Sie vorhandenen Code in ein UML-Modell importieren (Reverse Engineering). Hinsichtlich Code Engineering erfahren Sie außerdem, wie Sie ein vollständiges Round-Trip Engineering durchführen (entweder Modell->Code->Modell oder Code->Modell->Code). Grundlegende UML-Kenntnisse werden für dieses Tutorial vorausgesetzt.

Das Tutorial ist in die unten aufgeführten Abschnitte gegliedert. In den Anfangsabschnitten des Tutorials arbeiten Sie mit einem mit UModel vorinstallierten Beispielprojekt. Wenn Sie mit UModel schnell ein neues Modellierungsprojekt von Grund auf neu erstellen möchten, überspringen Sie diese Abschnitte und gehen Sie direkt zu Forward Engineering (Modell zu Code) 59.

- Erste Schritte 15
- Use Cases 16
- Klassendiagramme 25
- Erstellen von abgeleiteten Klassen 34
- Objektdiagramme 40
- Komponentendiagramme 48
- Deployment-Diagramme 54
- Forward Engineering (Modell zu Code) 59
- Reverse Engineering (Code zu Modell) 69

In diesem Tutorial werden die folgenden Beispielprojektdateien aus dem Verzeichnis **C:\Benutzer\<Benutzername>\Dokumente\Altova\UModel2023\UModelExamples\Tutorial** verwendet:

BankView-start.ump	Dies ist die UModel-Projektdatei, die den Anfangszustand des Tutorial-Beispiels darstellt. Zu diesem Zeitpunkt sind mehrere Modelldiagramme, sowie Klassen, Objekte und Modellelemente vorhanden. Während Sie das Tutorial durcharbeiten, werden Sie mit UModel neue Elemente oder Diagramme hinzufügen oder vorhandene bearbeiten. Bitte beachten Sie, dass dieses Projekt absichtlich nicht fertig gestellt wurde, daher werden Validierungsfehler und Warnungen angezeigt, wenn Sie die Projektsyntax über den Menübefehl **Projekt | Projektsyntax überprüfen** überprüfen. Im Tutorial wird erläutert, wie Sie diese Fehler beheben.
BankView-finish.ump	Dies ist die UModel-Projektdatei, die den Endzustand der Tutorial-Beispieldatei darstellt.

Anmerkung: Alle UModel-Beispieldateien stehen anfangs im Verzeichnis **C:\ProgramData\Altova\UModel2023** zur Verfügung. Wenn ein Benutzer die Applikation zum ersten Mal startet, werden die Beispieldateien in **C:\Benutzer\<Benutzername>\Dokumente\Altova\UModel2023\UModelExamples** kopiert. Sie sollten die Beispieldateien im Anfangsverzeichnis daher nicht verschieben, bearbeiten oder löschen.

 © 2016-2022 Altova GmbH

2.1 Erste Schritte

Wenn Sie UModel nach der Installation zum ersten Mal öffnen, wird ein neues Standardprojekt "NeuesProjekt1" geöffnet. Wenn Sie UModel später wieder starten, wird das zuletzt geladene Projekt geöffnet. Um UModel-Projekte (.ump-Dateien) zu erstellen, zu öffnen und zu speichern, verwenden Sie die Windows-Standardbefehle auf dem Menü **Datei** oder der Symbolleiste.

2.2 Use Cases

In diesem Tutorialabschnitt wird gezeigt, wie Sie ein Use Case-Diagramm erstellen. Dabei werden Sie mit den Basisfunktionalitäten der Benutzeroberfläche von UModel vertraut gemacht. Es werden die folgenden Aufgaben erläutert:

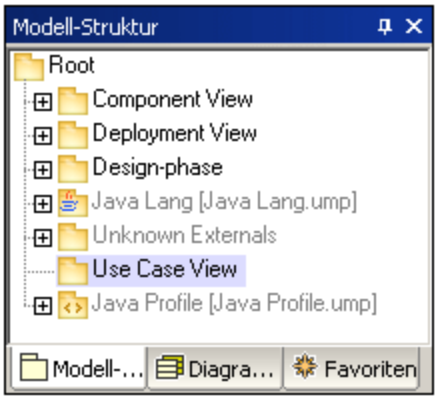
- Hinzufügen eines neuen Pakets zum Projekt
- Hinzufügen eines neuen Use Case-Diagramms zum Projekt
- Hinzufügen von Case-Elemente zum Diagramm und Definieren der Abhängigkeiten zwischen den Elementen
- Ausrichten und Anpassen der Elemente im Diagramm
- Ändern des Stil von Diagrammen in einem UModel-Projekt

Um fortzufahren, starten Sie UModel und öffnen Sie das Projekt **BankView-start.ump** (siehe auch Öffnen des Tutorial-Projekts 15).

Hinzufügen eines neues Pakets zu einem Projekt

Wie Sie bereits aus UML wissen, ist ein Paket ein Behälter für Klassen und andere UML-Elemente, sie z.B. Use Cases. Beginnen wir also, indem wir ein Paket erstellen, in dem ein neues Use Case-Diagramm gespeichert wird. Beachten Sie, dass es in UModel nicht unbedingt notwendig ist, dass sich ein bestimmtes Diagramm in einem bestimmten Paket befindet; Aus Gründen der Übersichtlichkeit und Einheitlichkeit empfiehlt es sich jedoch, Diagramme in Paketen zu organisieren.

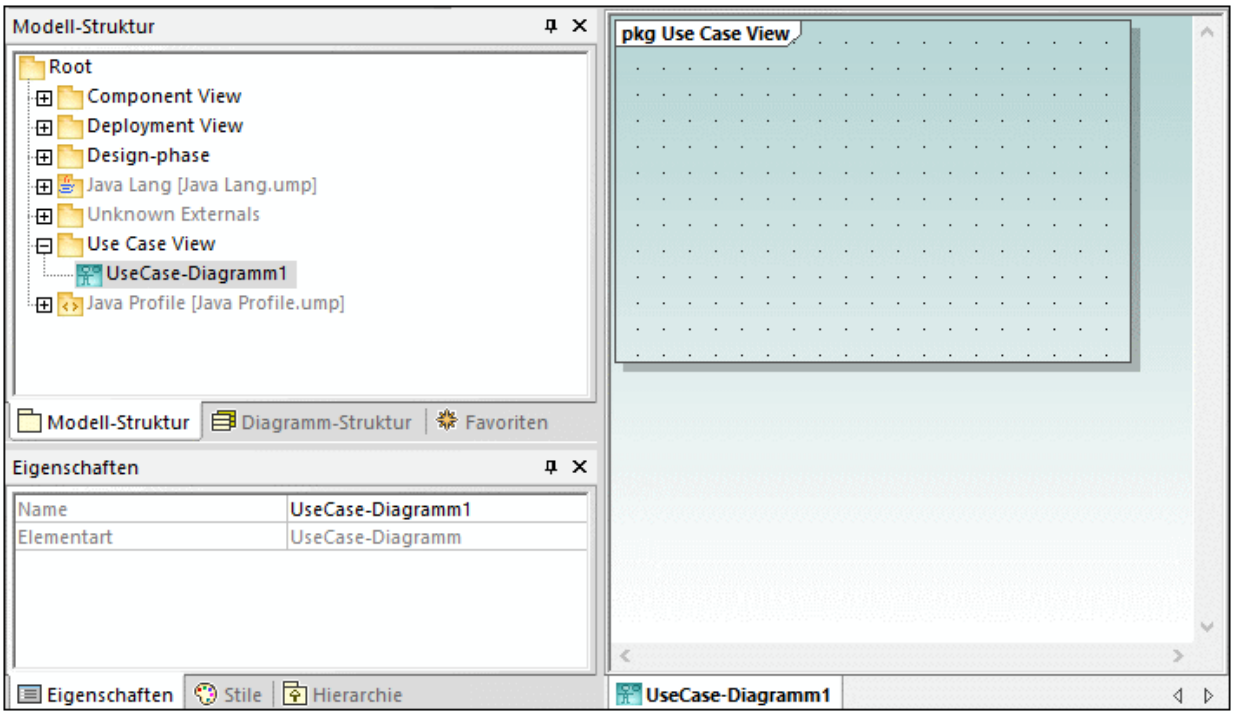
1. Rechtsklicken Sie im Fenster "Modell-Struktur" auf das **Root**-Paket und wählen Sie **Neues Element | Paket**.
2. Geben Sie den Namen des neuen Pakets ein, (in diesem Beispiel. "Use Case View") und drücken Sie die Eingabetaste.



Hinzufügen eines Use Case-Diagramms zu einem Paket

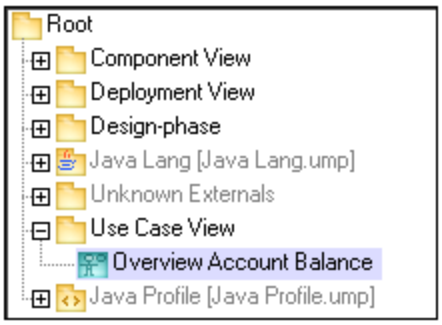
1. Rechtsklicken Sie auf das zuvor erstellte Paket "Use Case View".
2. Wählen Sie den Befehl **Neues Diagramm | UseCase-Diagramm**.

 © 2016-2022 Altova GmbH



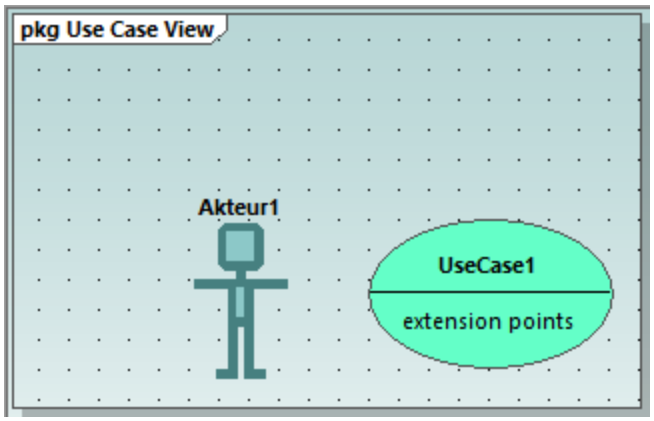
Dem Paket wurde nun im Fenster **Modell-Struktur** ein Use Case-Diagramm hinzugefügt und es wurde ein neues **Diagramm**-Fenster angelegt, das automatisch einen Standardnamen erhielt.

3. Doppelklicken Sie im Fenster "Modell-Struktur" auf den Diagrammnamen, ändern Sie ihn in "Overview Account Balance" und drücken Sie die **Eingabetaste**.

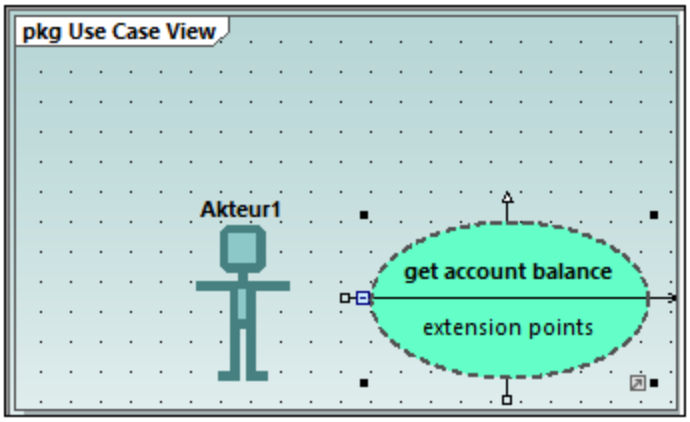


Hinzufügen von Use Case-Elementen zum Use Case-Diagramm

1. Rechtsklicken Sie in das neu erstellte Diagramm und wählen Sie **Neu | Akteur**. Das Akteurelement wird an der Mausposition eingefügt.
2. Klicken Sie auf das Use Case-Symbol in der Symbolleiste und dann in das Diagrammfenster, um das Element einzufügen. Es wird ein Element "UseCase1" eingefügt. Beachten Sie, dass das Element und sein Name aktuell ausgewählt sind und dass seine Eigenschaften im Fenster **Eigenschaften** angezeigt werden.



3. Ändern Sie den Titel in "get account balance" und drücken Sie zur Bestätigung die Eingabetaste. Doppelklicken Sie auf den Titel, wenn er nicht aktiv ist.



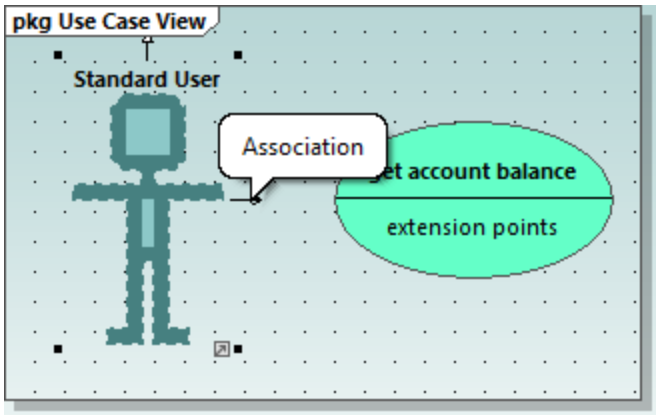
Anmerkung: Durch Drücken von Strg + Eingabetaste können Sie einen mehrzeiligen Use Case-Namen erstellen.

Bearbeiten von UModel Elementen: Ziehpunkte und Bereiche

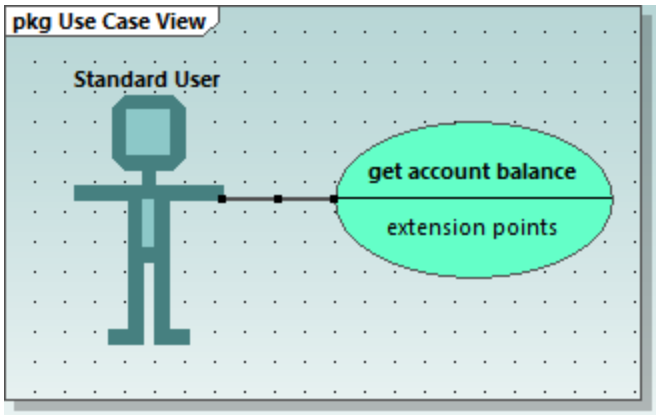
Wenn ein Element in der Diagrammanzeige ausgewählt ist, werden verschiedene Ziehpunkte und andere Elemente zum Bearbeiten angezeigt. Ziehpunkte dienen zum Erstellen von Beziehungen zwischen Elementen oder zum Anzeigen bzw. Ausblenden bestimmter Bereiche des Elements, wie unten gezeigt.

1. Doppelklicken Sie auf den Text "Akteur1" des Akteur-Elements, ändern Sie den Namen in "Standard User" und drücken Sie zur Bestätigung die **Eingabetaste**.
2. Platzieren Sie den Mauszeiger über den "**Ziehpunkt**" rechts vom Akteur. Es erscheint ein Tooltip mit dem Inhalt "Assoziation".

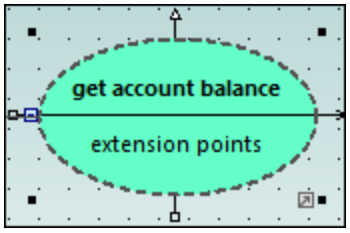
 © 2016-2022 Altova GmbH



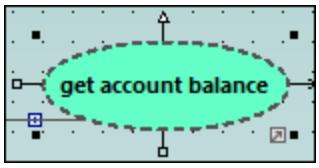
3. Klicken Sie auf den Ziehpunkt, ziehen Sie die Assoziationslinie nach rechts auf den Anwendungsfall (Use case) "get account balance". Zwischen Dem Akteur und dem Anwendungsfall wurde nun eine Assoziation erstellt. Die Eigenschaften der Assoziation sind auch im Fenster "Eigenschaften" zu sehen. Die neue Assoziation wurde unter dem Element "Beziehungen" des Use Case View-Pakets hinzugefügt.



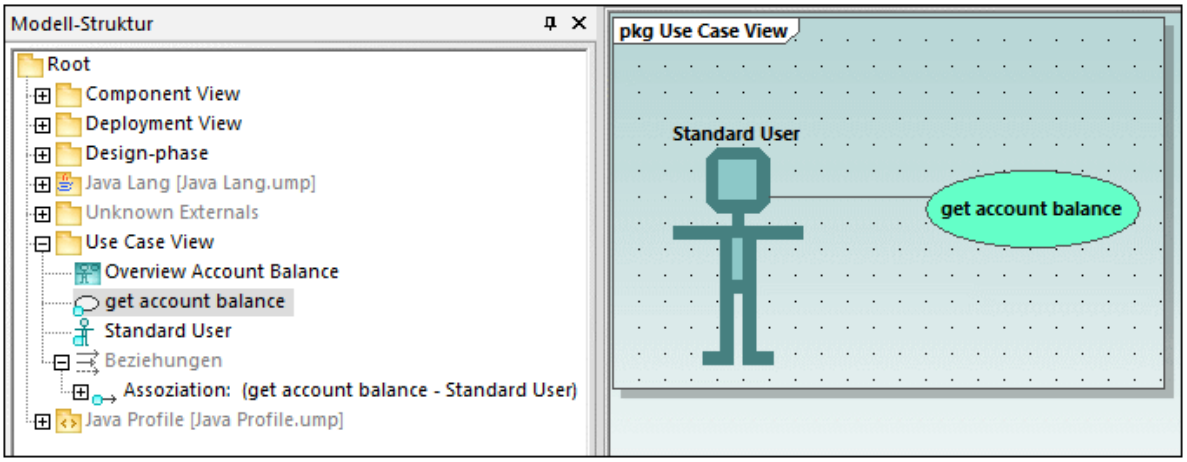
4. Klicken Sie auf den Anwendungsfall (Use Case) und ziehen Sie ihn nach rechts, um ihn neu zu positionieren. Die Eigenschaften der Assoziation sind auf dem Assoziationsobjekt zu sehen.
5. Klicken Sie auf den Anwendungsfall, um ihn auszuwählen und anschließend auf das Symbol zum Reduzieren am linken Rand der Use Case-Ellipse.



Der Bereich "extension points" wird nun ausgeblendet.



Ein blauer Punkt neben einem Elementsymbol im Fenster "Modell-Struktur" bedeutet, dass das Element im dem aktuellen Diagrammfenster sichtbar ist. So sehen Sie etwa in der Abbildung unten drei Elemente im Diagramm. Diese drei Elemente sind in der Modell-Struktur mit einem blauen Punkt gekennzeichnet:

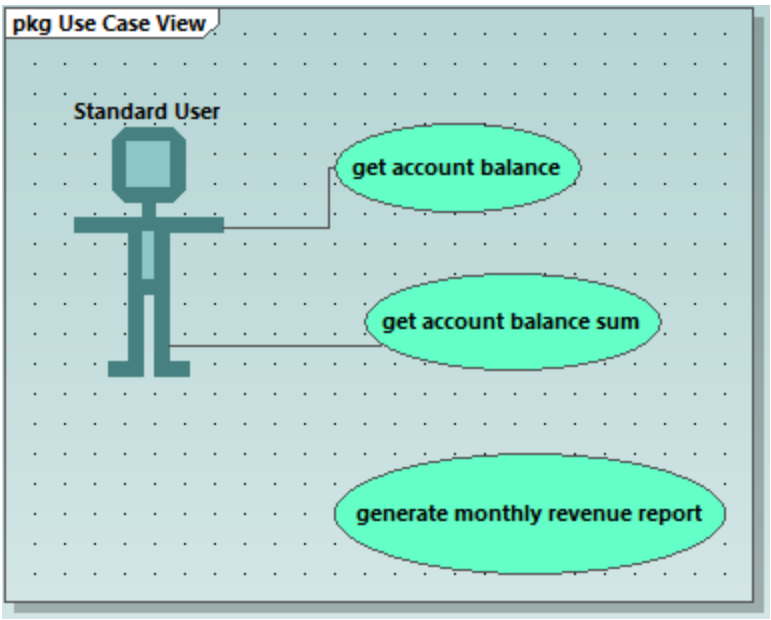


Wenn Sie die Größe des Akteurs anpassen, wird auch das Textfeld angepasst, das auch mehrere Zeilen enthalten kann. Mit **Strg + Eingabetaste** wird eine Zeilenschaltung in den Text eingefügt.

So stellen Sie das Use Case-Diagramm fertig:

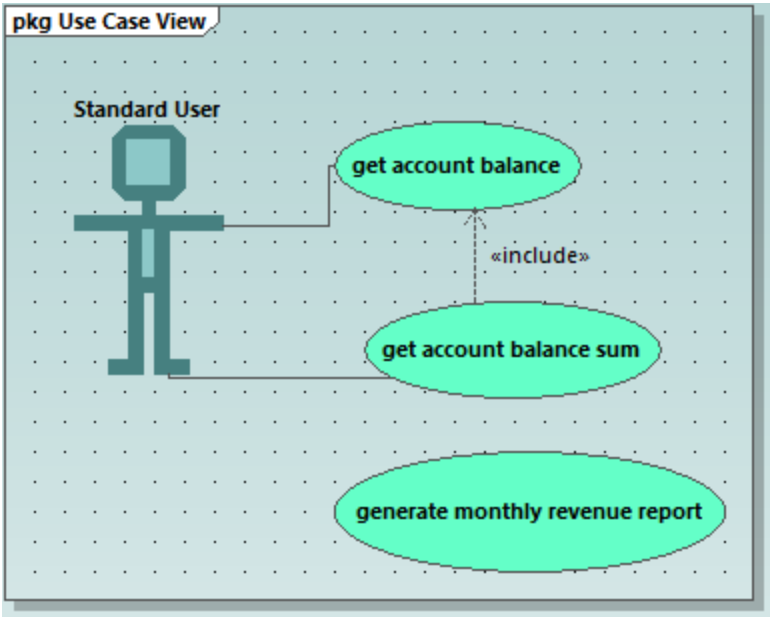
1. Klicken Sie in der Symbolleiste auf das Use Case-Symbol und halten Sie **gleichzeitig** die Strg-Taste gedrückt.
2. Klicken Sie im Diagramm an zwei verschiedenen Stellen vertikal übereinander in das Fenster, um zwei weitere Use Cases hinzuzufügen. Lassen Sie anschließend die Strg-Taste los.
3. Geben Sie dem ersten Use Case den Namen "get account balance sum" und dem zweiten den Namen, "generate monthly revenue report".
4. Klicken Sie auf das Einklappsymbol der einzelnen Use Case-Ellipsen um den extension-Bereich auszublenden.
5. Klicken Sie auf den Akteur und erstellen Sie mit Hilfe des Assoziationsziehpunkts eine Assoziation zwischen "Standard User" und "get account balance sum".

 © 2016-2022 Altova GmbH



So erstellen Sie eine "include"-Abhängigkeit zwischen Use Cases (und somit einen untergeordneten Use Case)

- Klicken Sie auf den **Include**-Ziehpunkt der "get account balance sum" Use Case-Ellipse (unter der Ellipse) und ziehen Sie die Abhängigkeit auf "get account balance". Es wird eine "include"-Abhängigkeit erstellt und auf dem strichlierten Pfeil wird das include-Stereotyp angezeigt.

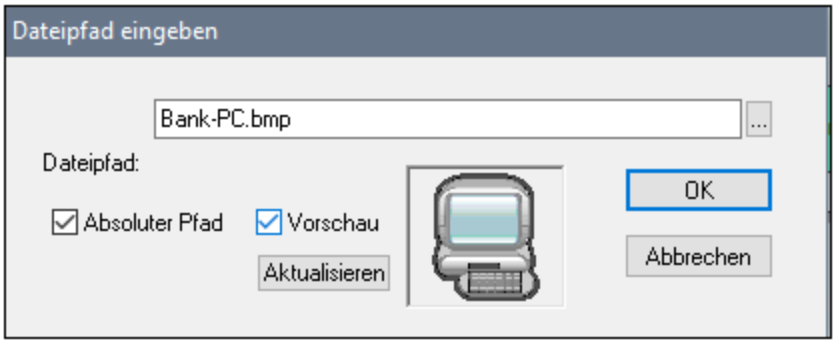


Einfügen von benutzerdefinierten Akteuren

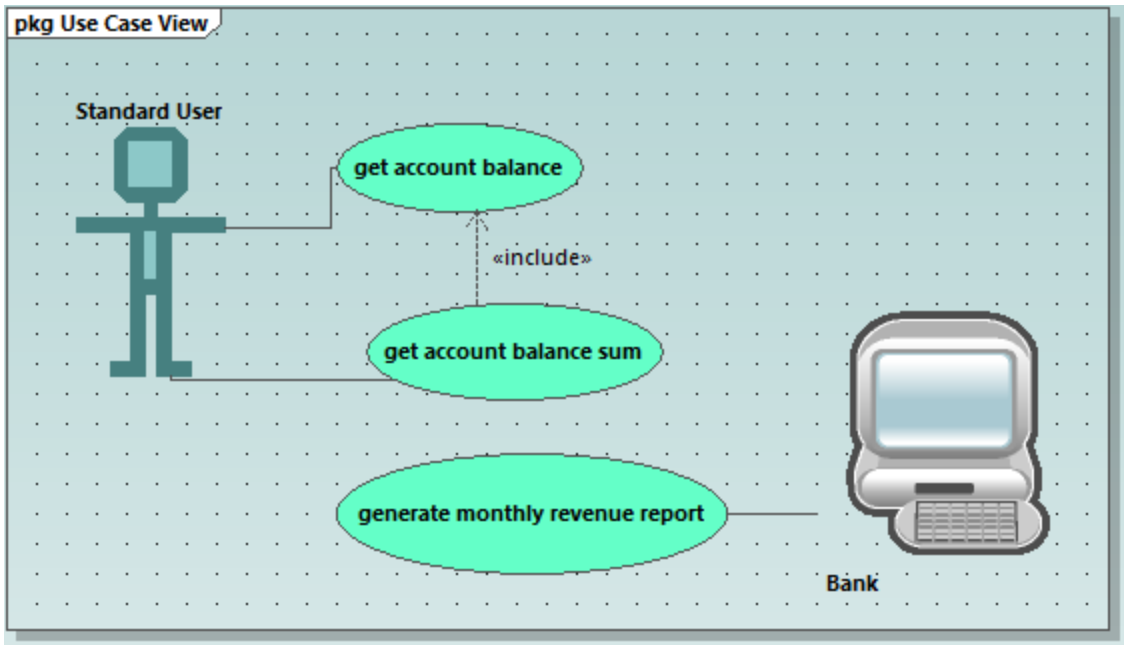
Beim Akteur im Use Case "generate monthly revenue report" handelt es sich nicht um eine Person, sondern um einen automatisierten Batch-Auftrag, der von einem Bankcomputer ausgeführt wird. In der Anleitung unten

wird erläutert, wie Sie einen neuen Akteur zum Diagramm hinzufügen und ein benutzerdefiniertes Bild dafür verwenden.

1. Fügen Sie mit Hilfe des Symbolleistensymbols **Akteur** einen Akteur in das Diagramm ein.
2. Benennen Sie den Akteur in "Bank" um.
3. Klicken Sie im Fenster "Eigenschaften" auf das Durchsuchen-Symbol neben dem Eintrag "Symboldateiname" und navigieren Sie zur **Datei Bank-PC.bmp**, die sich im selben Ordner wie das Projekt befindet.
4. Deaktivieren Sie das Kontrollkästchen **Absoluter Pfad**, um den Pfad relativ zu machen. Bei Auswahl von **Vorschau** wird im Dialogfeld eine Vorschau der ausgewählten Datei angezeigt.



5. Klicken Sie auf OK, um die Einstellungen zu bestätigen und fügen Sie den neuen Akteur ein. Verschieben Sie den neuen Akteur "Bank" und platzieren Sie ihn rechts vom untersten Use Case.
6. Klicken Sie in der Symbolleiste auf das Symbol **Assoziation** und ziehen Sie es vom Akteur "Bank" zum Use Case "generate monthly revenue report".
   Dies ist eine alternative Methode zum Erstellen einer Assoziation.



Anmerkung: Die Hintergrundfarbe, mit der die Bitmap-Grafik transparent gemacht werden kann, hat die RGB-Werte 82.82.82.

© 2016-2022 Altova GmbH

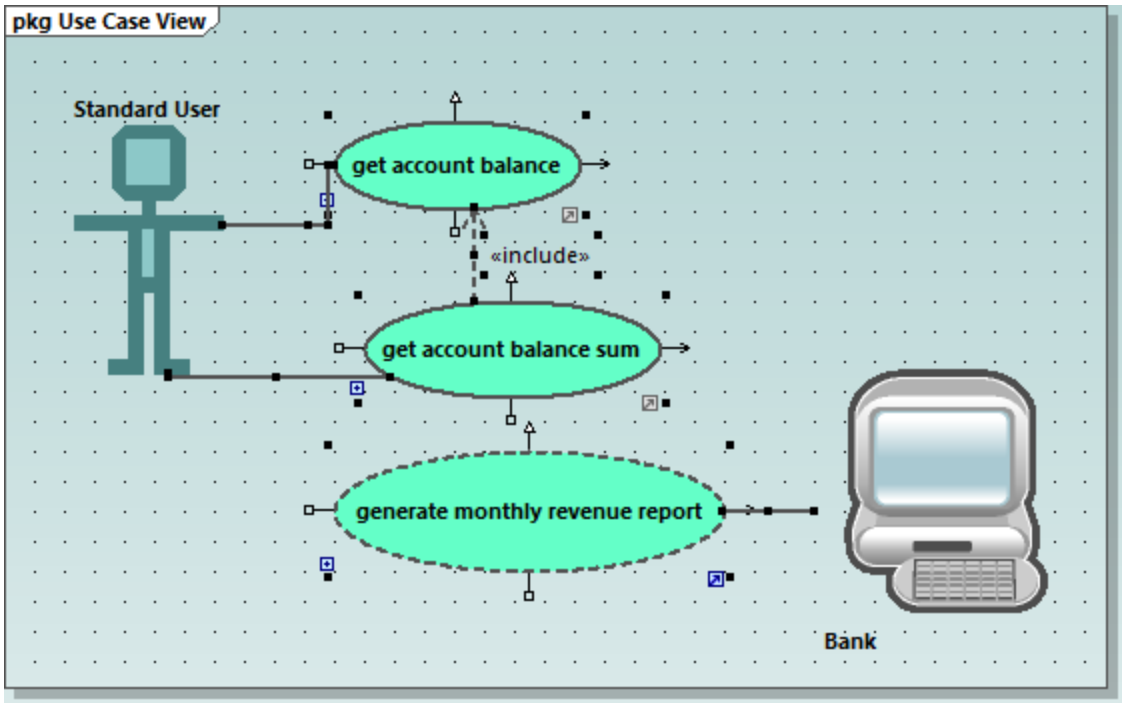
Ausrichten und Anpassen der Größe von Diagrammelementen

Wenn Sie Komponenten in einem Diagramm mit der Maus ziehen, erscheinen Hilfslinien, an denen Sie ein Element an jedem anderen Element im Diagramm ausrichten können. Diese Option kann folgendermaßen aktiviert bzw. deaktiviert werden:

1. Klicken Sie im Menü **Extras** auf **Optionen.**
2. Klicken Sie auf das Register **Ansicht.**
3. Aktivieren Sie in der Gruppe **Ausrichtung** das Kontrollkästchen **Hilfslinien aktivieren**.

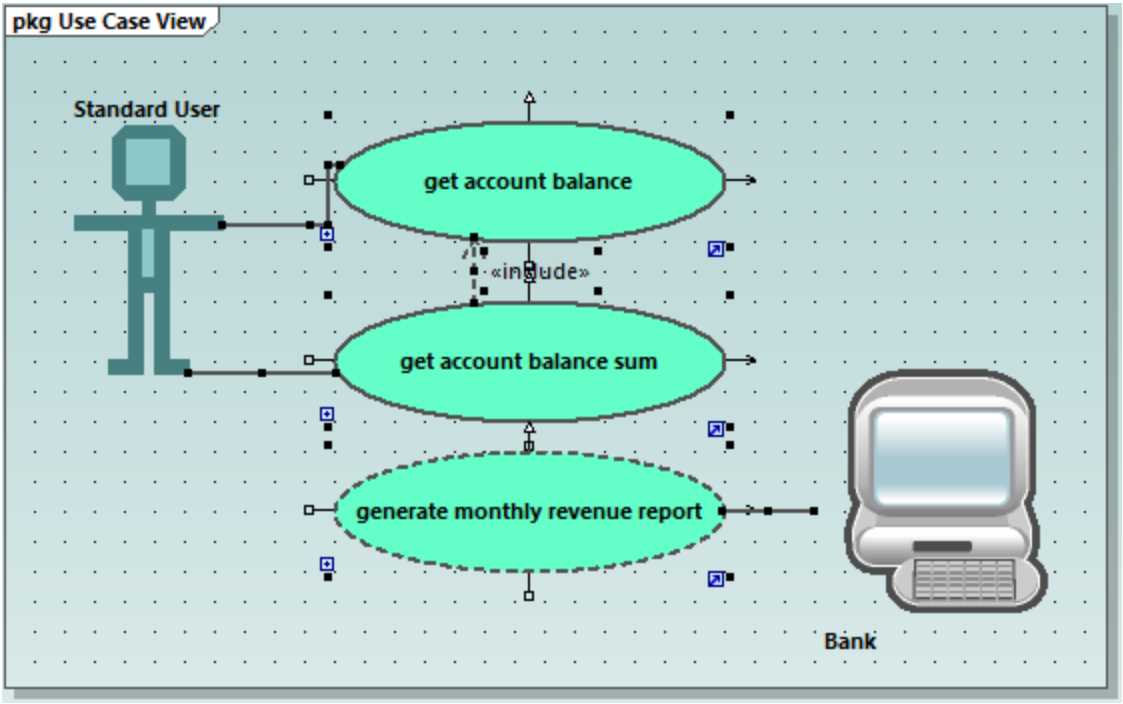
Sie können die Größe mehrerer Elemente auch folgendermaßen ausrichten und anpassen:

1. Ziehen Sie einen Auswahlrahmen auf, indem Sie auf dem Diagrammhintergrund ein Rechteck aufziehen, in dem von oben angefangen alle drei Use Cases enthalten sind. Alternativ dazu können Sie mehrere Elemente auch durch Drücken der **Strg**-Taste beim Anklicken der Elemente auswählen. Beachten Sie, das die Umrandung des zuletzt markierten Use Case im Diagramm und im Übersichtsfenster strichliert angezeigt wird.



   Alle Use Cases sind nun ausgewählt, wobei die unterste Ellipse die Basis für die folgenden Anpassungen bildet.

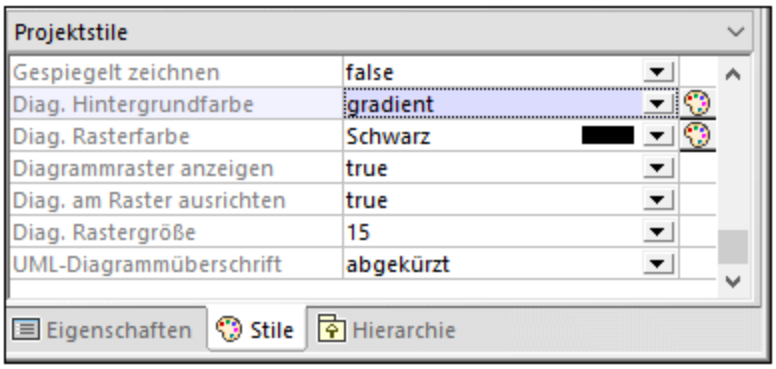
2. Klicken Sie in der Symbolleiste auf das Symbol **Größe angleichen** .
3. Klicken Sie auf das Symbol **Horizontal zentrieren** , um alle Ellipsen parallel auszurichten.



Ändern des Stils von Diagrammen in einem Projekt

Standardmäßig haben alle Diagramme des Tutorial-Projekts eine Hintergrunde mit Farbverlauf und es wird ein Hintergrundraster angezeigt. Das Aussehen von Diagrammen in einem Projekt kann konfiguriert werden. Um z.B. die Hintergrundfarbe aller Diagramme zu ändern, gehen Sie folgendermaßen vor:

1. Klicken Sie im Fenster **Eigenschaften** auf **Stile**.
2. Finden Sie unter **Projekstile** die Einstellung **Diag. Hintergrundfarbe**.



3. Ändern Sie den Wert von "gradient" in eine Farbe Ihrer Wahl.

So aktivieren oder deaktivieren Sie das Diagrammhintergrundraster:

- Ändern Sie die Einstellung **Diagrammraster anzeigen** von "true" in "false". (Wenn ein Diagramm gerade offen ist, können Sie alternativ dazu auch auf die Symbolleisten-Schaltfläche **Raster anzeigen** klicken.)

 © 2016-2022 Altova GmbH

2.3 Klassendiagramme

In diesem Tutorialabschnitt werden die folgenden Aufgaben erläutert:

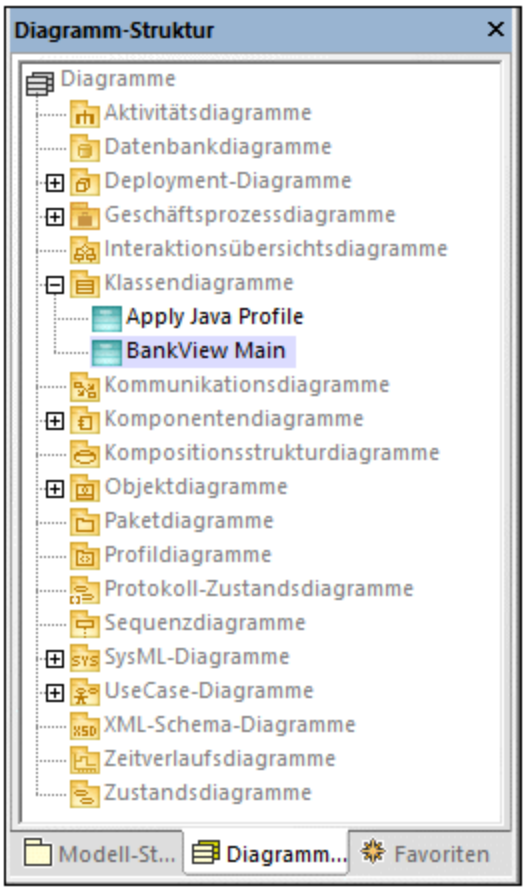
- Hinzufügen einer abstrakten Klasse zu einem vorhandenen Klassendiagramm
- Hinzufügen von Klasseneigenschaften und -operationen und Definieren von Parametern, ihrer Richtung und ihres Typs
- Hinzufügen eines Rückgabetyps zu einer Operation
- Ändern von Symbolen in UML-konforme Symbole
- Löschen und Ausblenden von Klasseneigenschaften und -operationen
- Erstellen einer Kompositions-Assoziation zwischen zwei Klassen

Um fortzufahren, starten Sie UModel und öffnen Sie das Projekt **BankView-start.ump** (siehe auch Öffnen des Tutorial-Projekts 15).

Hinzufügen einer abstrakten Klasse

Das Diagramm, zu dem die abstrakte Klasse hinzugefügt wird, hat den Namen "BankView Main" und kann folgendermaßen geöffnet werden:

1. Erweitern Sie im Fenster **Diagramm-Struktur** das Paket "Klassendiagramme", um alle im Projekt enthaltenen Klassendiagramme zu sehen.



2. Wählen Sie eine der folgenden Methoden:

- Doppelklicken Sie auf das "BankView Main"-Diagrammsymbol.
- Kicken Sie mit der rechten Maustaste auf das Diagramm und wählen Sie im Kontextmenü den Befehl **Diagramm öffnen**.

Anmerkung: Sie können das Diagramm auch über das Fenster **Modell-Struktur** öffnen. Suchen Sie das Diagramm zuerst unter dem Paket "Root | Design-phase | BankView | com | altova | bankview" und verwenden Sie anschließend eine der oben beschriebenen Methoden, um es zu öffnen.

Im Klassendiagramm sehen Sie zwei konkrete Klassen mit einer Kompositions-Assoziation zwischen diesen Klassen.

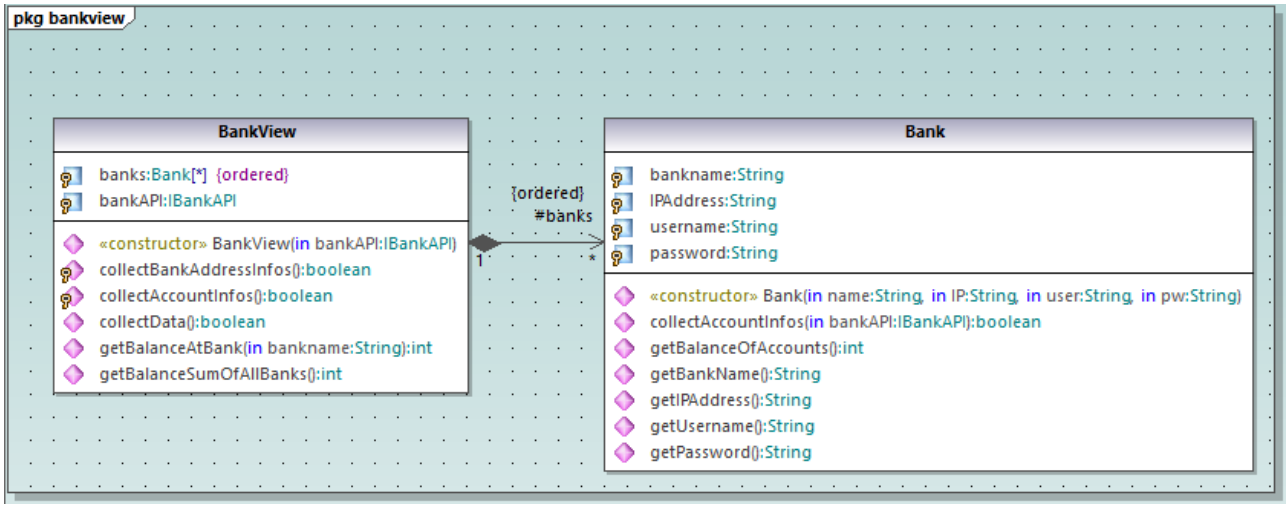
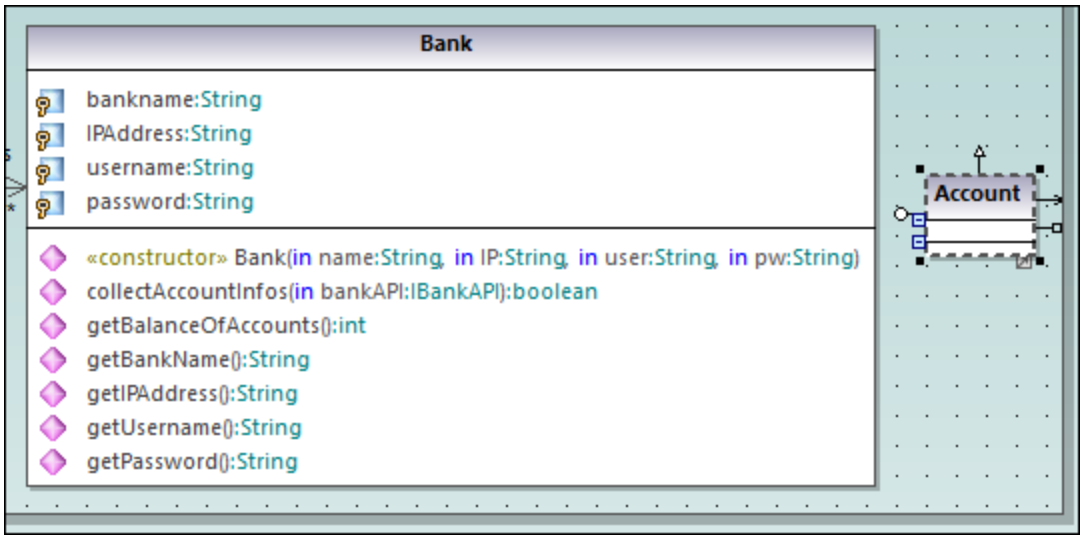


Diagramm "Bank View Main"

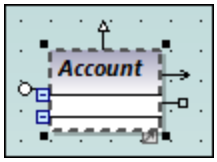
Die neue abstrakte Klasse kann folgendermaßen hinzugefügt werden:

1. Klicken Sie auf die Symbolleisten-Schaltfläche **Klasse** und klicken Sie anschließend rechts von der Klasse `Bank`, um die neue Klasse einzufügen.
2. Doppelklicken Sie auf den Namen der neuen Klasse und ändern Sie ihn in `Account`.

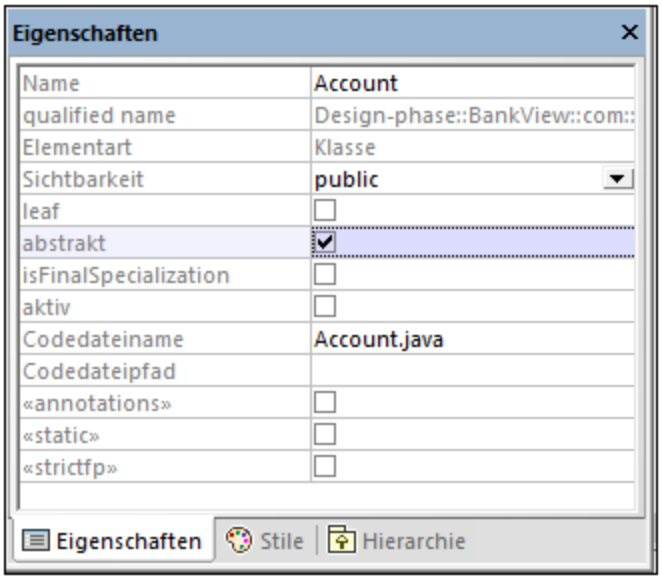
 © 2016-2022 Altova GmbH



3. Aktivieren Sie im Fenster **Eigenschaften** das Kontrollkästchen **abstrakt,** um die Klasse zu einer abstrakten zu machen. Der Klassentitel wird nun kursiv gedruckt angezeigt. Dadurch werden abstrakte Klassen gekennzeichnet.

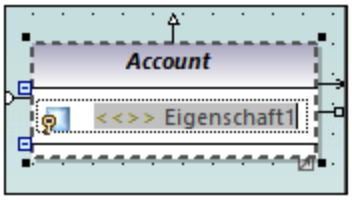


4. Geben Sie in das Textfeld **Codedateiname** "Account.java" ein, um die Java-Klasse zu definieren.

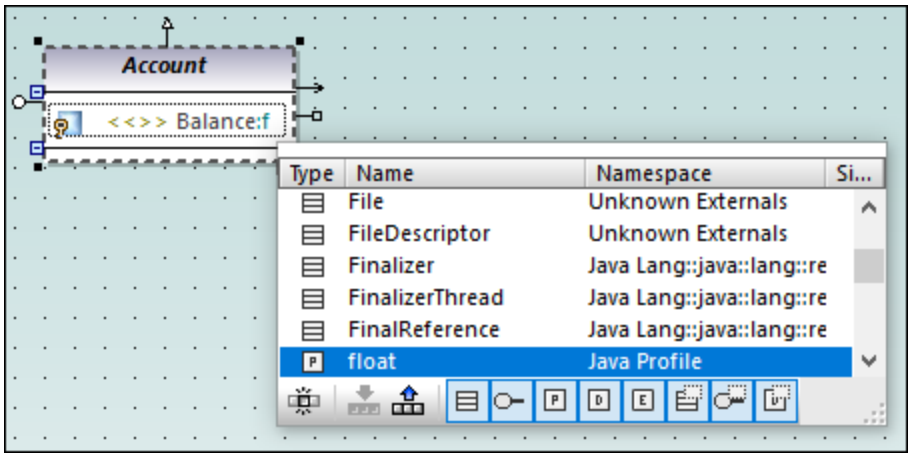


Hinzufügen von Eigenschaften zu einer Klasse

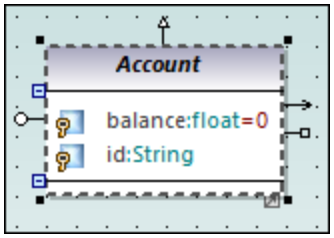
1. Klicken Sie mit der rechten Maustaste auf die Klasse "Account" und wählen Sie **Neu | Eigenschaft** oder drücken Sie **F7**. Daraufhin wird eine Standardeigenschaft `Eigenschaft 1` mit Stereotypmarkierungen << >> eingefügt.



2. Ändern Sie den Eigenschaftsnamen in `balance` gefolgt von einem Doppelpunkt (:) . Daraufhin wird eine Dropdown-Liste mit allen gültigen Typen angezeigt.
3. Geben Sie "f" ein und drücken Sie die Eingabetaste, um den Rückgabetyp "float" einzugeben. Beachten Sie dass, die Groß- und Kleinschreibung in Dropdown-Listen eine Rolle spielt.



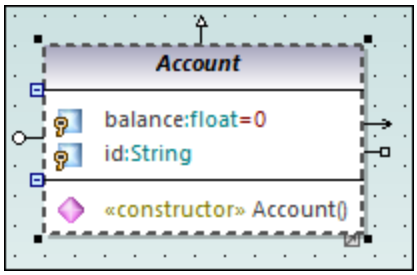
4. Hängen Sie in derselben Zeile "=0" an, um den Standardwert zu definieren.
5. Erstellen Sie auf dieselbe Art, wie beschrieben, eine neue Eigenschaft `id` vom Typ `String`.



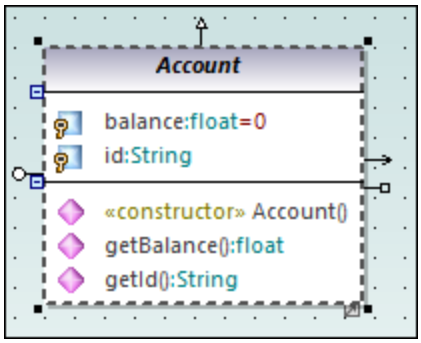
Hinzufügen von Operationen zu einer Klasse

1. Klicken Sie mit der rechten Maustaste auf die Klasse `Account` und wählen Sie **Neu | Operation** oder drücken Sie **F8**.
2. Geben Sie als Operationsnamen "Account()" ein. Beachten Sie, dass sich das Stereotyp in `<<constructor>>`, geändert hat, da der Operationsname mit dem Klassennamen identisch ist.

© 2016-2022 Altova GmbH

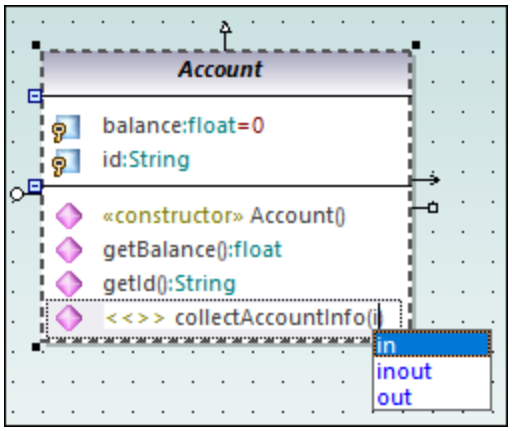


3. Fügen Sie auf dieselbe Art zwei weitere Operationen hinzu und geben Sie ihnen den Namen `getBalance():float` bzw. `getId():String`.

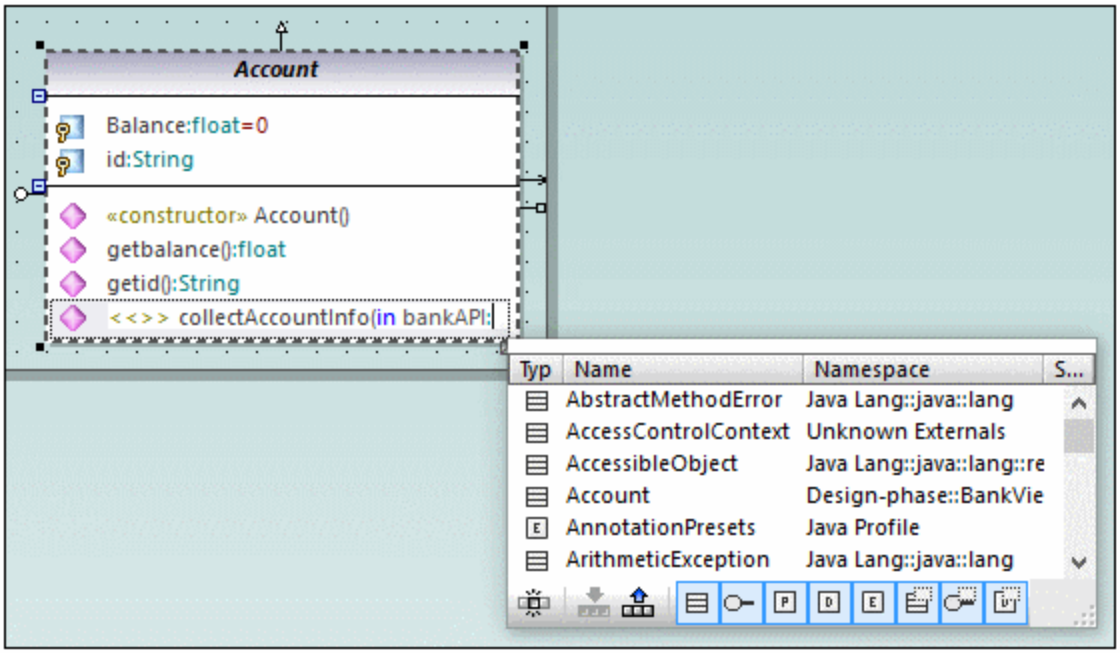


Fügen wir nun einen neue Operation hinzu, die einen Parameter hat. Außerdem werden wir die Richtung und den Typ des Parameters definieren.

1. Drücken Sie **F8,** um eine weitere Operation, `collectAccountInfo()`, zu erstellen.
2. Platzieren sie den Mauscursor innerhalb der Klammern und geben Sie "i" ein. Daraufhin wird eine Dropdown-Liste angezeigt, aus der Sie die Parameterrichtung: `in`, `inout` oder `out` auswählen können.



3. Wählen Sie den Eintrag "in" aus der Dropdown-Liste aus, geben Sie ein Leerzeichen ein und bearbeiten Sie den Eintrag weiter in derselben Zeile.
4. Geben Sie als Parameter "bankAPI", gefolgt von einem Doppelpunkt (:) ein. Daraufhin wird eine Dropdown-Liste angezeigt, aus der Sie den Parametertyp auswählen können.



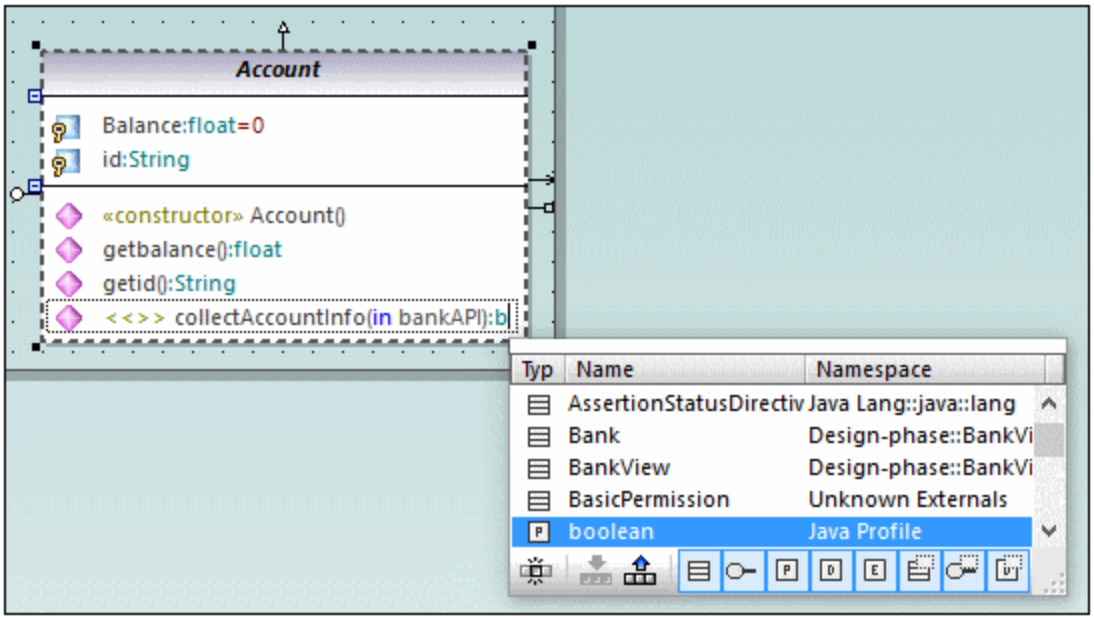
5. Wählen Sie **IBankAPI** aus der Dropdown-Liste aus.

Hinzufügen eines Rückgabetyps zu einer Operation

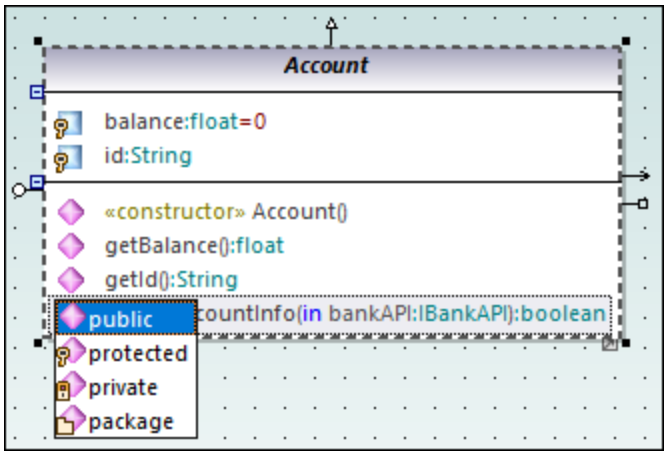
Bisher wurde der Operationsparameter hinzugefügt, hat aber noch keinen Rückgabetyp. So fügen Sie einen Rückgabetyp hinzu:

1. Platzieren Sie den Mauscursor hinter das schließende Klammerzeichen ")" und geben Sie einen Doppelpunkt ein (:). Daraufhin wird eine Dropdown-Liste angezeigt, aus der Sie den Rückgabetyp auswählen können.
2. Drücken Sie die Taste "b" und wählen Sie als Datentyp `boolean` aus.

© 2016-2022 Altova GmbH



Um die Sichtbarkeit einer Operation zu definieren (z.B. "private", "protected", "public"), klicken Sie auf das Symbol vor dem Operationsnamen und wählen Sie den gewünschten Wert aus, z.B.:



Die Sichtbarkeit "package" gilt für Java. In C# können Sie die Sichtbarkeit mit Hilfe von "package" als "internal" definieren. Nähere Informationen zu den UModel-Elemententsprechungen in den einzelnen Sprachen finden Sie unter UModel-Elemententsprechungen 228.

Ändern von Symbolen in UML-konforme Symbole

Die Sichtbarkeitssymbole können gegebenenfalls in UML-konforme Symbole geändert werden:

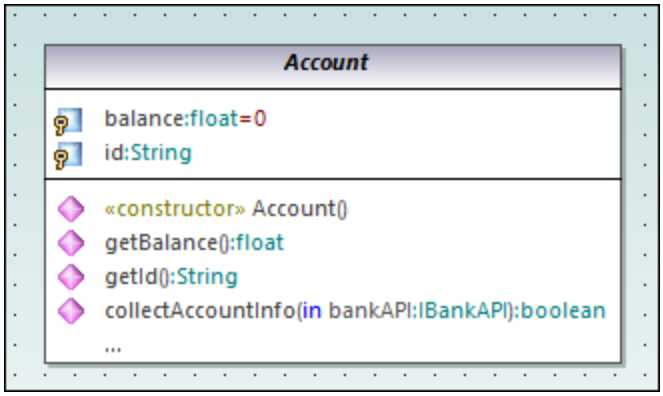
1. Wählen Sie im Fenster **Stile**, aus der Dropdown-Liste ganz oben, den Eintrag **Projektstile**.
2. Scrollen Sie hinunter zur Einstellung **Sichtbarkeit anzeigen** und wählen Sie **UML-Stil** aus.

Löschen und Ausblenden von Klasseneigenschaften und -Operationen aus einem Klassendiagramm

Drücken Sie **F8,** um z.B. eine Operation `Operation1` zur Klasse `Account` hinzuzufügen.

Um diese Operation zu löschen, wählen Sie sie aus und drücken Sie die Taste **Entfernen**. (Alternativ dazu klicken Sie mit der rechten Maustaste darauf und wählen Sie im Kontextmenü die Option **Löschen** aus). Daraufhin erscheint ein Meldungsfeld, in dem Sie gefragt werden, ob Sie das Element aus dem Projekt löschen möchten. Klicken Sie auf **Ja**, um die `Operation1` aus dem Klassendiagramm und aus dem Projekt zu löschen.

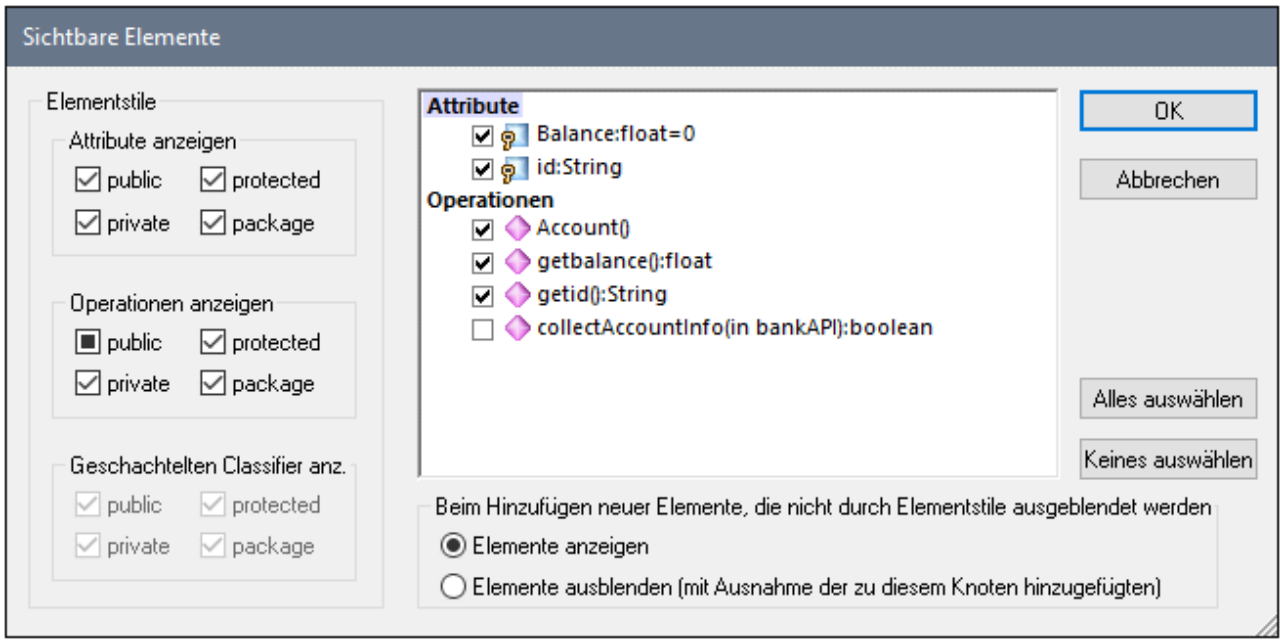
Um die Operation aus der Klasse im Diagramm, nicht aber aus dem Projekt zu löschen, drücken Sie **Strg+Entf**. Dadurch wird die Operation aus dem Diagramm ausgeblendet, obwohl sie im Projekt weiterhin vorhanden ist. Klassen mit ausgeblendeten Mitgliedern werden mit einem Auslassungszeichen (...) angezeigt (siehe Abbildung unten):



Eine Klasse mit ausgeblendeten Operationen

Um die Operation wieder einzublenden, doppelklicken Sie auf die Auslassungspunkte am unteren Rand der Klasse. Daraufhin wird ein Dialogfeld angezeigt, in dem Sie die Elemente, die im Diagramm angezeigt werden sollen, auswählen können, z.B.:

 © 2016-2022 Altova GmbH



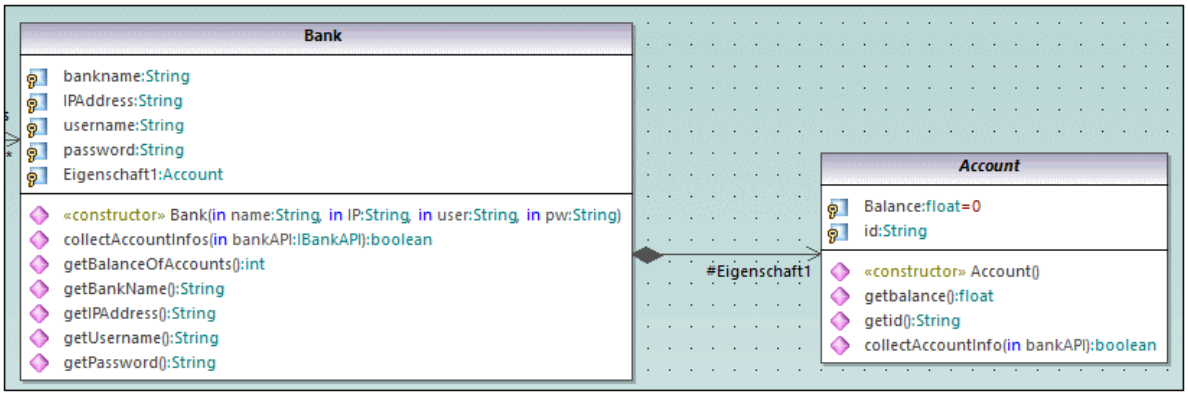
Dialogfeld "Sichtbare Elemente"

Sie können UModel auch so konfigurieren, dass ein Meldungsfeld angezeigt wird, wenn Sie versuchen, ein Objekt aus dem Diagramm zu löschen. Gehen Sie dazu folgendermaßen vor:

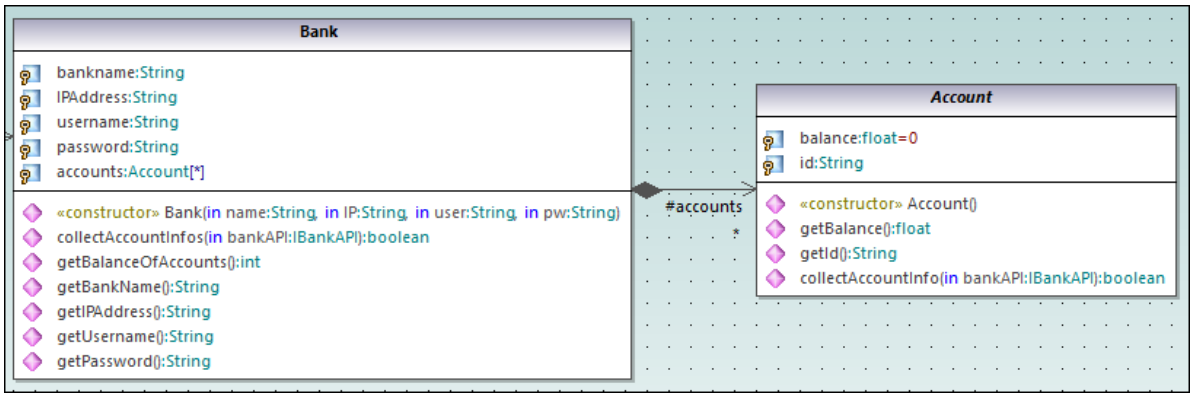
1. Klicken Sie im Menü **Extras** auf **Optionen**.
2. Klicken Sie auf das Register **Bearbeiten**.
3. Deaktivieren Sie unter **Rückfrage vor Löschen aus dem Projekt** das Kontrollkästchen **in Diagrammen**.

Erstellen einer Kompositions-Assoziation zwischen den Klassen "Bank" und "Account"

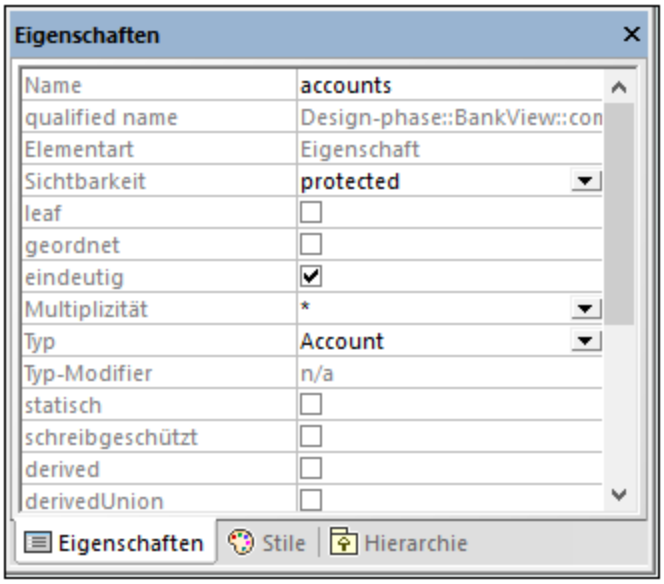
1. Klicken Sie auf die Symbolleisten-Schaltfläche **Komposition** und ziehen Sie diese anschließend von der Klasse `Bank` zur Klasse `Account`. Die Klasse wird markiert, wenn die Assoziation erstellt werden kann. In der Klasse Bank wird eine neue Eigenschaft (`Eigenschaft1:Account`) erstellt und die beiden Klassen werden durch einen Kompositions-Assoziationspfeil verbunden.



2. Doppelklicken Sie auf die neue Eigenschaft `Eigenschaft1` in der Klasse `Bank` und ändern Sie sie in "accounts", ohne dabei die Typdefinition `Account` (in Blaugrün) zu löschen.
3. Drücken Sie die Taste **Ende**, um den Textcursor ans Ende der Zeile zu setzen.
4. Geben Sie die das eckige Klammer auf-Zeichen ([ ) ein und wählen Sie aus der Dropdown-Liste ein Sternchen (*) aus. Damit definieren Sie die *Multiplizität*, d.h. dass eine Bank mehrere Konten (accounts) haben kann.



Beachten Sie, dass der zuvor zum Diagramm hinzugefügte Multipilizitätsbereich auch im Fenster **Eigenschaften** angezeigt wird:



2.3.1 Erstellen von abgeleiteten Klassen

In diesem Tutorialabschnitts werden die folgenden Aufgaben erläutert:

- Hinzufügen eines neuen Klassendiagramms zum Projekt
- Hinzufügen bestehender Klassen zu einem Diagramm
- Hinzufügen einer neuen Klasse zu einem Diagramm
- Erstellen von abgeleiteten Klassen anhand einer abstrakten Klassen mit Hilfe von Generalisierungen

 © 2016-2022 Altova GmbH

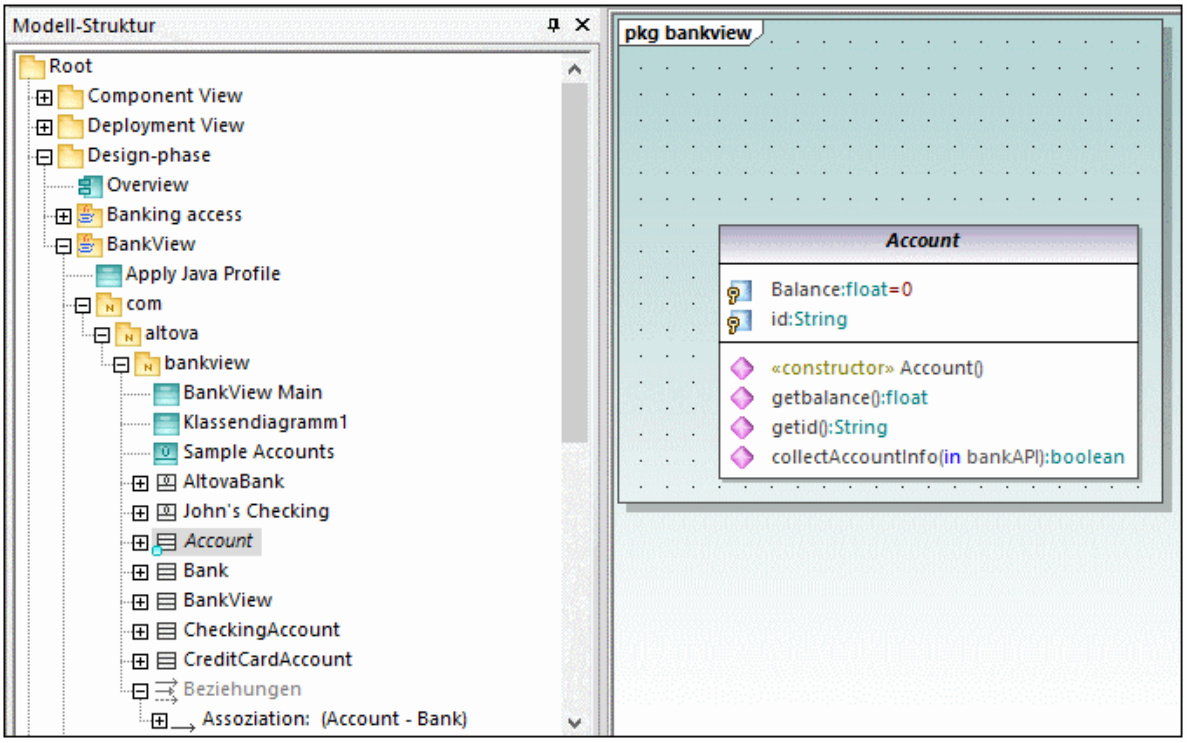
Anmerkung: Es wird davon ausgegangen, dass Sie den vorherigen Tutorial-Abschnitt, Klassendiagramme 25 bereits durchgearbeitet haben, um die abstrakte Klasse `Account` zu erstellen.

Erstellen eines neuen Klassendiagramms

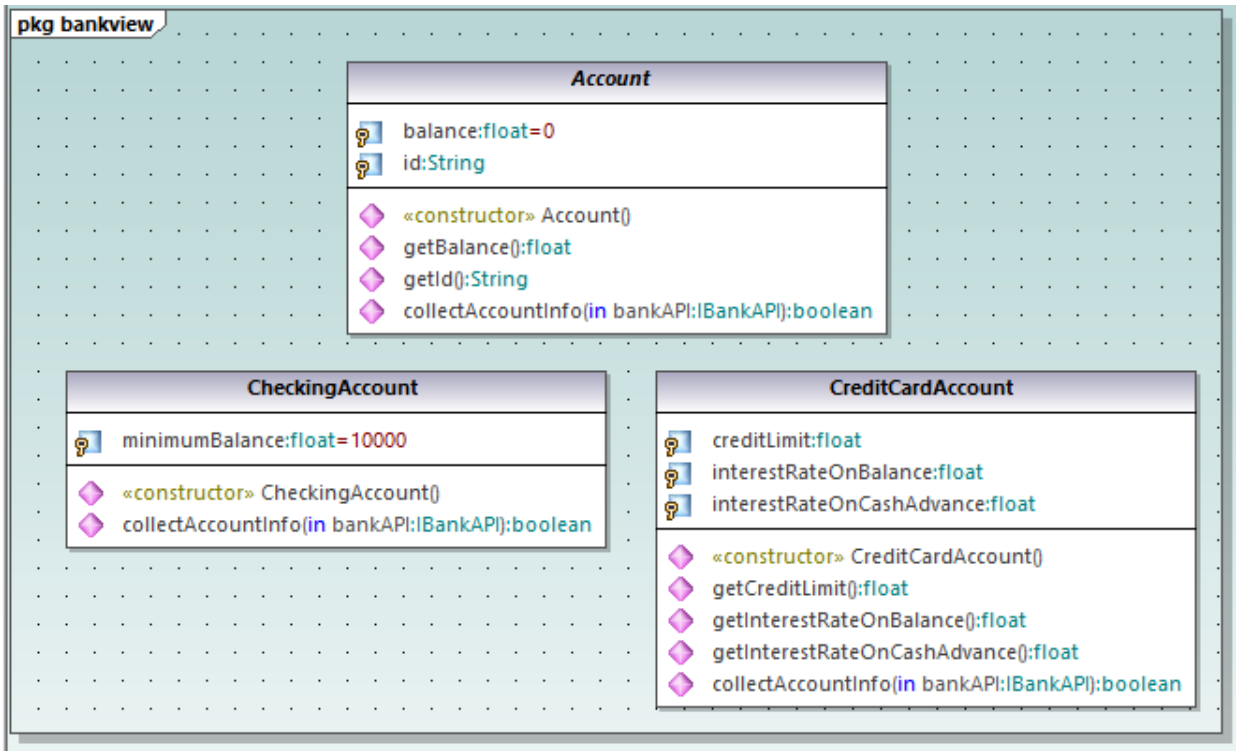
1. Rechtsklicken Sie im Fenster **Modell-Struktur** auf das Paket `bankview` (unter **Root | Design-phase | BankView | com | altova**) und wählen Sie **Neues Diagramm | Klassendiagramm**.
2. Doppelklicken Sie auf den neuen Eintrag "Klassendiagramm1", benennen Sie ihn in "Account Hierarchy" um und drücken Sie die **Eingabetaste**. Im Arbeitsbereich wird nun das neue Diagramm "Account Hierarchy" angezeigt.

Hinzufügen von bestehenden Klassen zu einem Diagramm

1. Klicken Sie im Paket `bankview` auf die Klasse `Account` (unter **com | altova | bankview**) und ziehen Sie sie in das Diagramm.



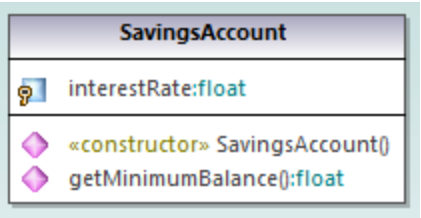
2. Klicken Sie auf die Klasse `CheckingAccount` (im selben Paket) und ziehen Sie diese ebenfalls in das Diagramm. Platzieren Sie die Klasse links unterhalb der Klasse "Account".
3. Fügen Sie auf dieselbe Art die Klasse `CreditCardAccount` ein. Platzieren Sie sie rechts von der Klasse `CheckingAccount`.



Hinzufügen einer neuen Klasse

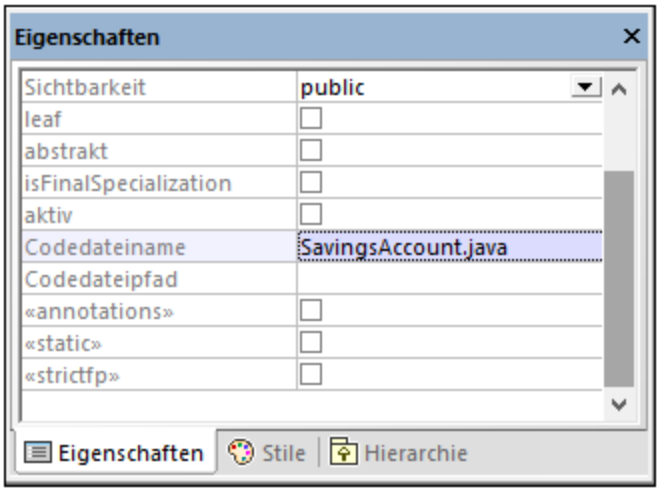
Die dritte abgeleitete Klasse, `SavingsAccount`, wird manuell zum Diagramm hinzugefügt.

1. Rechtsklicken Sie auf das Diagramm und wählen Sie **Neu | Klasse**. Daraufhin wird automatisch eine neue Klasse zum richtigen Paket (`bankview`) hinzugefügt, welches das aktuelle Klassendiagramm "Account Hierarchy" enthält.
2. Doppelklicken Sie auf den Klassennamen und ändern Sie ihn in `SavingsAccount`.
3. Erstellen Sie die Klassenstruktur wie unten gezeigt. Um Eigenschaften und Operationen hinzuzufügen, verwenden Sie die im vorherigen Tutorialabschnitt beschriebenen Methoden (siehe Klassendiagramme) 25.



3. Geben Sie auf dem Register **Eigenschaften** im Textfeld "Codedateiname" "SavingsAccount.java" ein, um die Java Codeklasse zu definieren.

 © 2016-2022 Altova GmbH



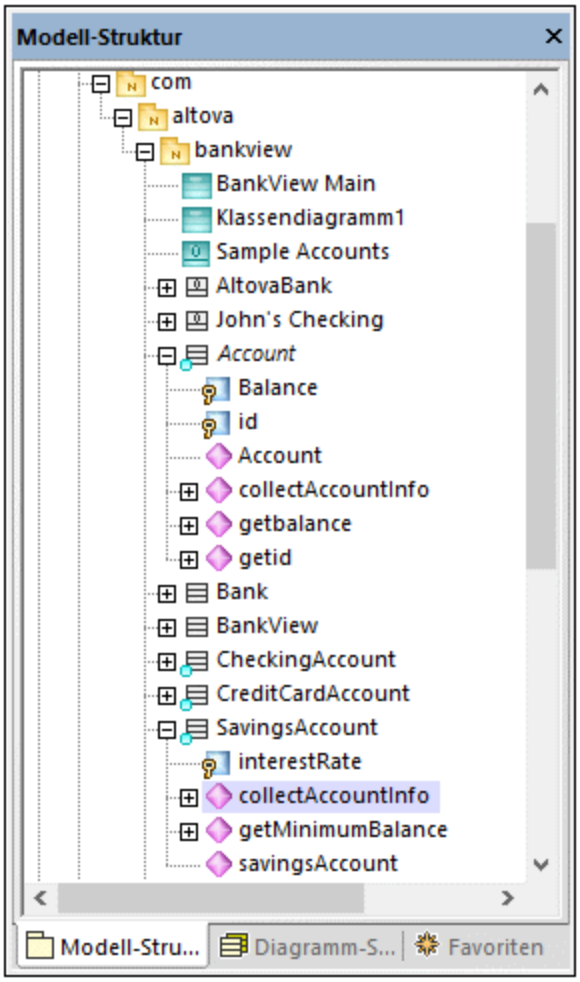
Eigenschaften und Operationen können direkt mit Hilfe von Drag & Drop oder den Standard-Tastenkombinationen von einer Klasse in die andere kopiert oder verschoben werden. Dies funktioniert in folgenden Fällen:

- innerhalb einer Klasse im aktuellen Diagramm
- zwischen verschiedenen Klassen im selben Diagramm
- im Fenster **Modell-Struktur**
- zwischen verschiedenen UML-Diagrammen durch Ziehen der kopierten Daten in ein anderes Diagramm

Dies kann mit Drag-and-Drop sowie den Standardtastaturbefehlen **Kopieren** und **Einfügen** (**Strg+C, Strg+V**) bewerkstelligt werden, siehe auch Umbenennen, Verschieben und Kopieren von Elementen 109. In diesem Beispiel können Sie die Operation `collectAccountInfo()` schnell aus der Klasse `Account` in die neue Klasse `SavingsAccount` kopieren. Gehen Sie dabei folgendermaßen vor:

1. Erweitern Sie die Klasse `Account` in der **Modell-Struktur**.
2. Rechtsklicken Sie auf die Operation `collectAccountInfo` und wählen Sie **Kopieren**.
3. Rechtsklicken Sie auf die Klasse `SavingsAccount` und wählen Sie **Einfügen.**

Die Operation wird in die Klasse `SavingsAccount` kopiert, wobei diese Klasse automatisch erweitert wird, um die neue Operation anzuzeigen.

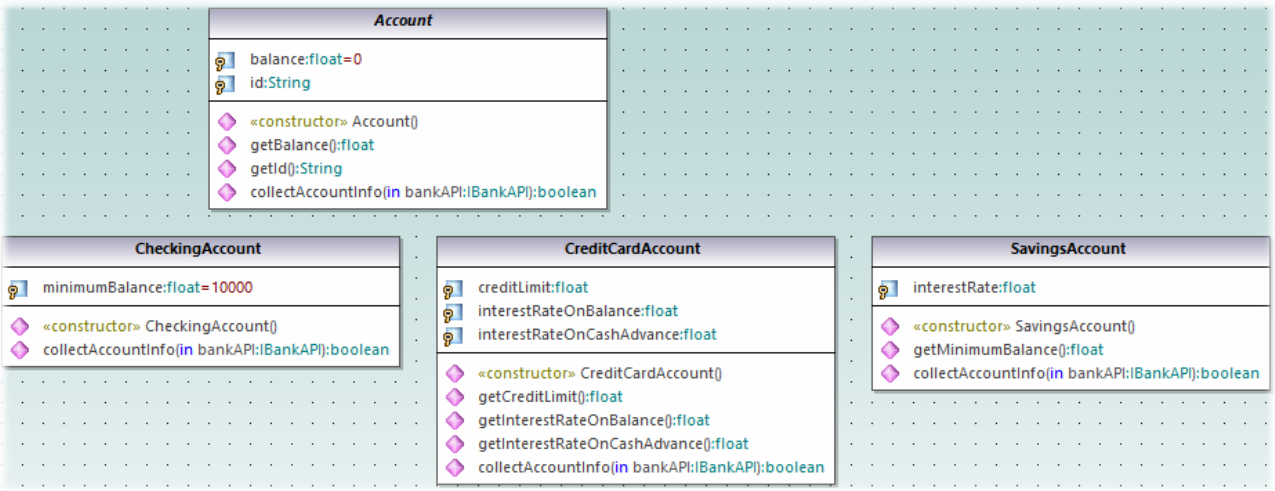


Die neue Operation ist nun auch im Klassendiagramm in der Klasse `SavingsAccount` zu sehen.

Erstellen von abgeleiteten Klassen mittels Generalisierung/Spezialisierung

Derzeit enthält das Klassendiagramm die abstrakte Klasse `Account` sowie drei spezifische Klassen.

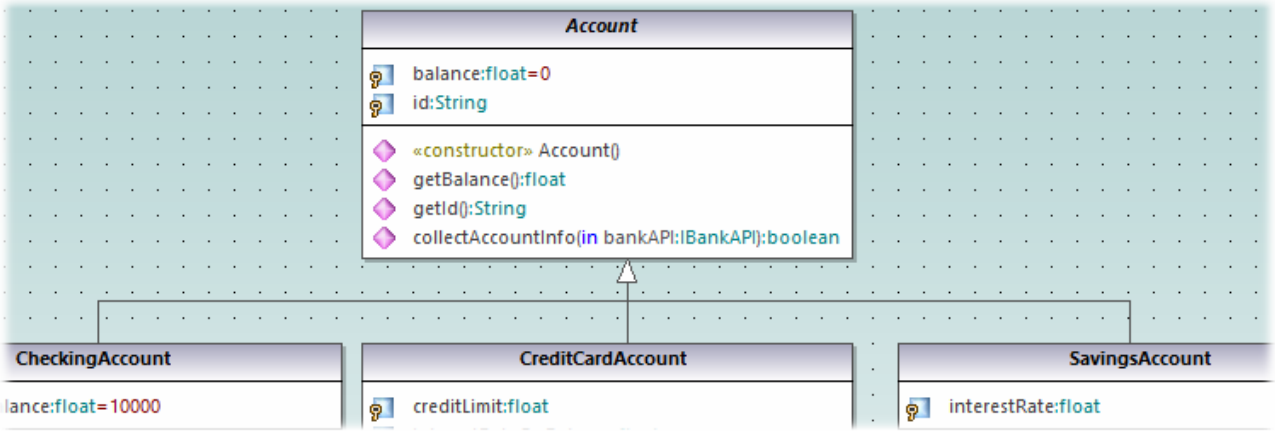
© 2016-2022 Altova GmbH



Wir wollen nun eine Generalisierungs-/Spezialisierungsbeziehung zwischen "`Account`" und den spezifischen Klassen erstellen, d.h. wir wollen drei abgeleitete konkrete Klassen erstellen.

1. Klicken Sie in der Symbolleiste auf das Symbol **Generalisierung** und halten Sie die **Strg**-Taste gedrückt.
2. Ziehen Sie den Cursor von `CreditCardAccount` in die Klasse `Account`.
3. Ziehen Sie von der Klasse `CheckingAccount` auf die *Pfeilspitze* der zuvor erstellten Generalisierung.
4. Ziehen Sie von der Klasse `SavingsAccount` auf die *Pfeilspitze* der zuvor erstellten Generalisierung: lassen Sie die **Strg**-Taste jetzt los.

Zwischen den drei Unterklassen und der Überklasse `Account` werden Generalisierungspfeile erstellt.



2.4 Objektdiagramme

In diesem Tutorialabschnitt werden die folgenden Aufgaben erläutert:

- Kombinieren von Klassen- und Objektdiagrammen zu einem Diagramm
- Erstellen von Objekten/Instanzen und Definieren von Beziehungen zwischen diesen
- Formatieren von Assoziationen/Objektbeziehungen
- Eingabe von realen Daten in Objekte/Instanzen

Um fortzufahren, starten Sie UModel und öffnen Sie das Projekt **BankView-start.ump** (siehe auch Öffnen des Tutorial-Projekts 15). Das Projekt enthält ein vordefiniertes Objektdiagramm "Sample Accounts", anhand dessen die oben beschriebenen Aufgaben demonstriert werden.

Kombinieren von Objekten und Klassen zu einem Diagramm

Navigieren Sie im Fenster **Modell-Struktur** zu folgendem Pfad: **Root | Design-phase | BankView | com | altova | bankview** und doppelklicken Sie auf das Symbol neben dem Diagramm "Sample Accounts".

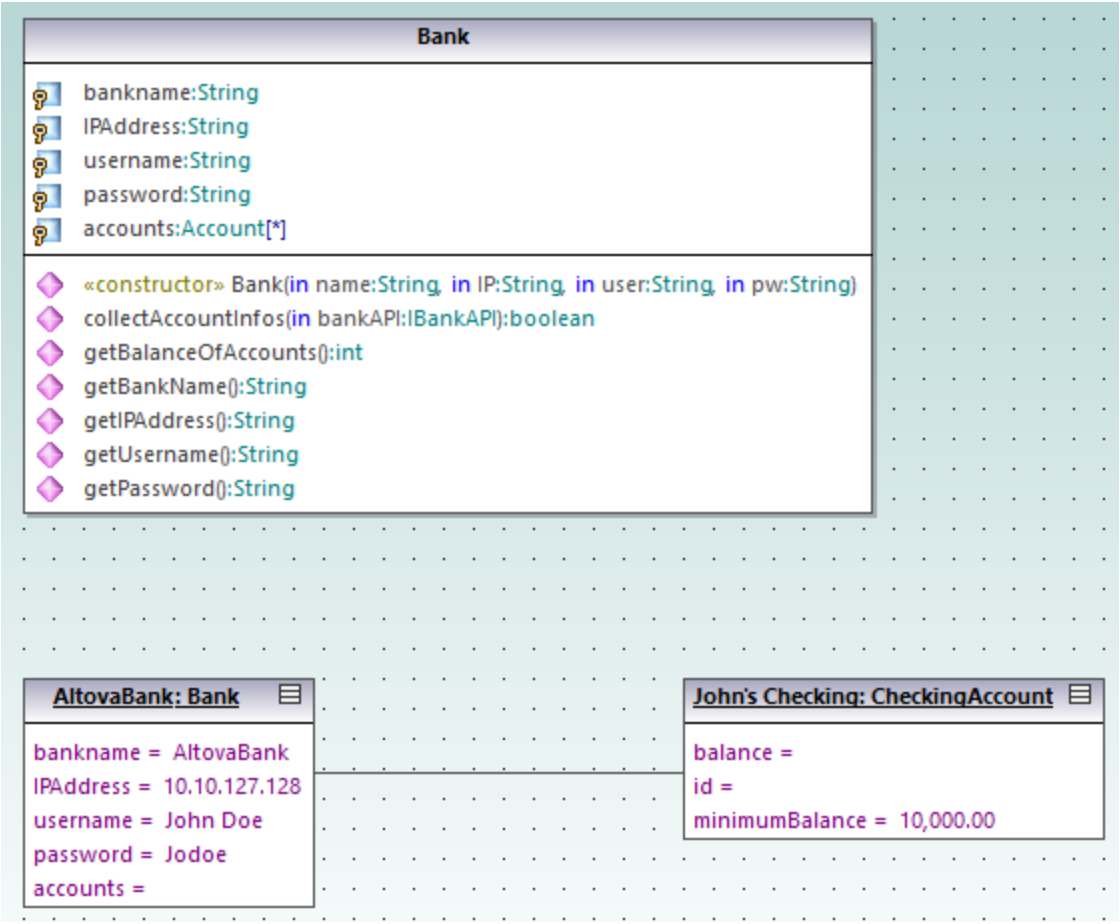
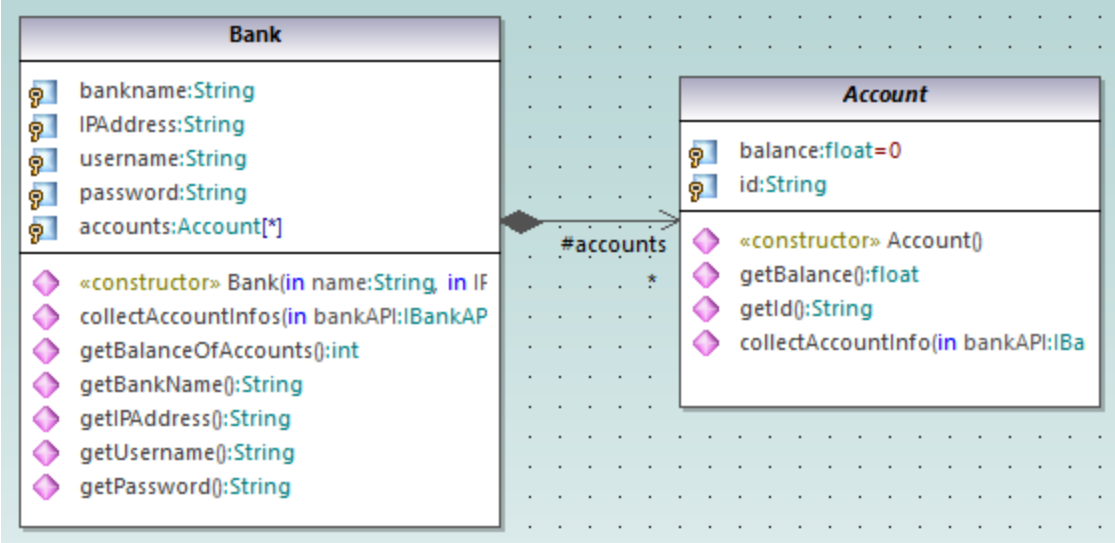


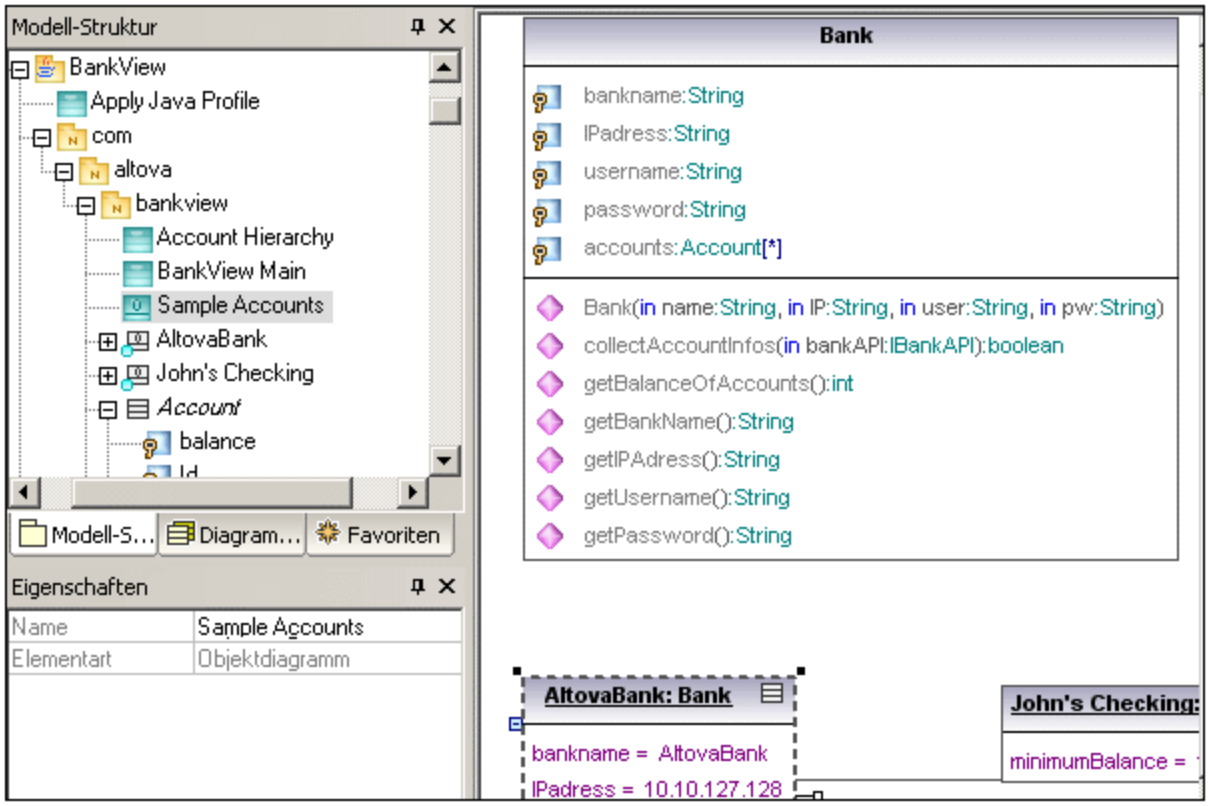
Diagramm "Sample Accounts"

 © 2016-2022 Altova GmbH

In diesem Objektdiagramm sind sowohl Klassen als auch Instanzen davon (Objekte) kombiniert. `AltovaBank:Bank` ist das Objekt/die Instanz der Klasse `Bank`, während `John's checking: CheckingAccount` eine Instanz der (noch nicht zum Diagramm hinzugefügten) Klasse `CheckingAccount` ist.

Fügen wir nun die fehlende Klasse `Account` zum Diagramm hinzu, indem wir sie aus der **Modell-Struktur** in das Diagramm ziehen. Beachten Sie, dass die Kompositions-Assoziation zwischen `Bank` und `Account` automatisch angezeigt wird (diese Assoziation wurde in einem der vorhergehenden Tutorial-Abschnitte definiert, siehe Klassendiagramme 25).

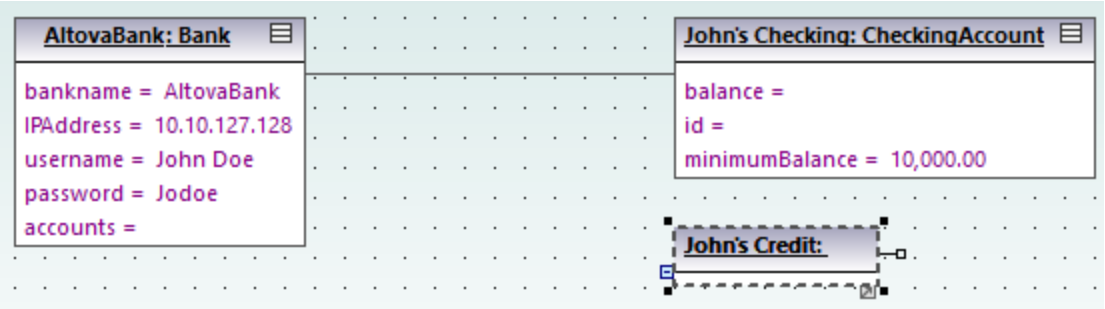




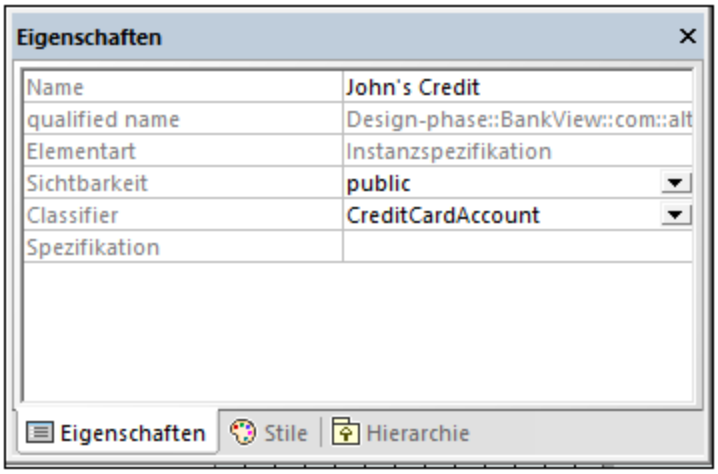
Hinzufügen eines neuen Objekts/einer neuen Instanz (Methode 1)

Wir werden nun ein neues Objekt zum Diagramm namens `John's Credit` hinzufügen. Dieses Objekt instantiiert die Klasse `CreditCardAccount`.

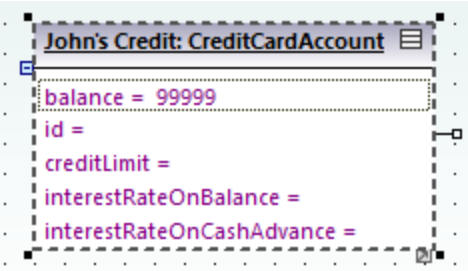
1. Klicken Sie in der Symbolleiste auf das **Instanzspezifikation**-Symbol und klicken Sie anschließend in das Diagramm unterhalb des Objekts `John's Checking: Checking Account`.
2. Ändern Sie den Namen der Instanz in `John's Credit` und drücken Sie die **Eingabetaste**.



3. Wählen Sie die Instanz aus, damit ihre Eigenschaften im Fenster **Eigenschaften** angezeigt werden.
4. Wählen Sie im Fenster Eigenschaften neben "Classifier" den Eintrag **CreditCardAccount** aus der Dropdown-Liste aus.



Die Instanz hat sich nun geändert und zeigt alle Eigenschaften der Klasse an. Doppelklicken Sie auf eine beliebige Eigenschaft, um einen Wert einzugeben, z.B.:



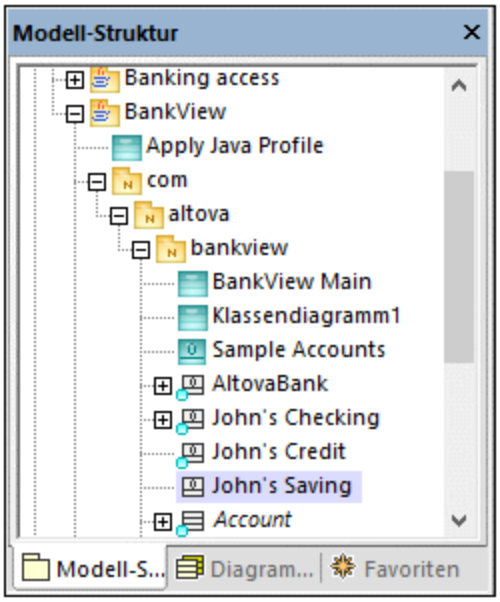
 © 2016-2022 Altova GmbH

Um bestimmte Knoten ein- oder auszublenden, klicken Sie mit der rechten Maustaste auf die Instanz und wählen Sie im Kontextmenü den Befehl **Knoteninhalt ein-/ausblenden (Strg+Umschalt+H)**.

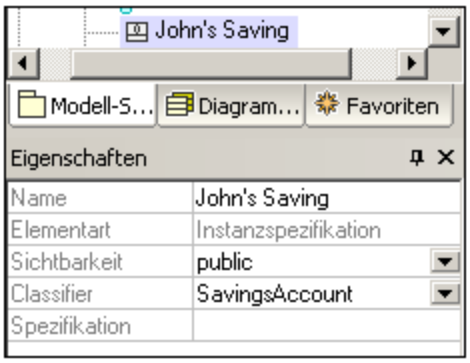
Hinzufügen eines neuen Objekts/einer neuen Instanz (Methode 2)

Wir werden nun eine neue Instanz der Klasse `SavingsAccount` hinzufügen, diesmal mit einer anderen Methode:

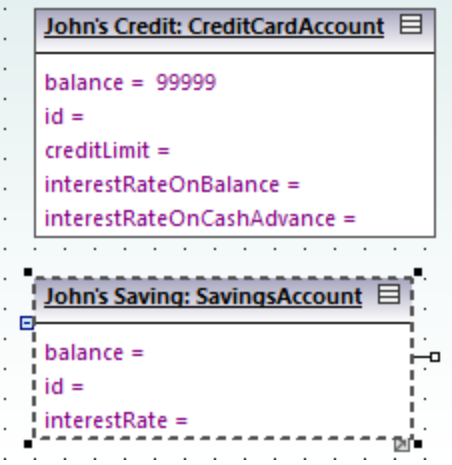
1. Rechtsklicken Sie im Fenster **Modell-Struktur** auf das `bankview`-Paket und wählen Sie **Neues Element | Instanzspezifikation**.
2. Benennen Sie die neue Instanz in `John's Saving` um und drücken Sie zum Bestätigen die Eingabetaste. Das neue Objekt wird zum Paket hinzugefügt und entsprechend gereiht.



3. Wählen, Sie während das Objekt weiterhin im Fenster **Modell-Struktur** ausgewählt ist, im Fenster **Eigenschaften** neben "Classifier" **SavingsAccount** aus.



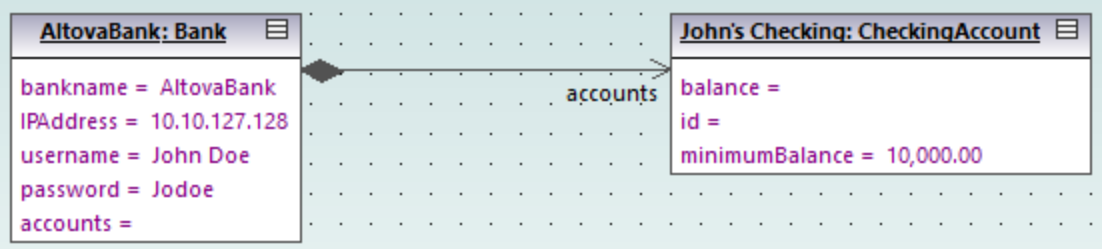
4. Ziehen Sie das Objekt `John's Saving` aus dem Fenster **Modell-Struktur** in das Diagramm und platzieren Sie es unterhalb von `John's credit`.



Erstellen von "Objektbeziehungen" zwischen Objekten

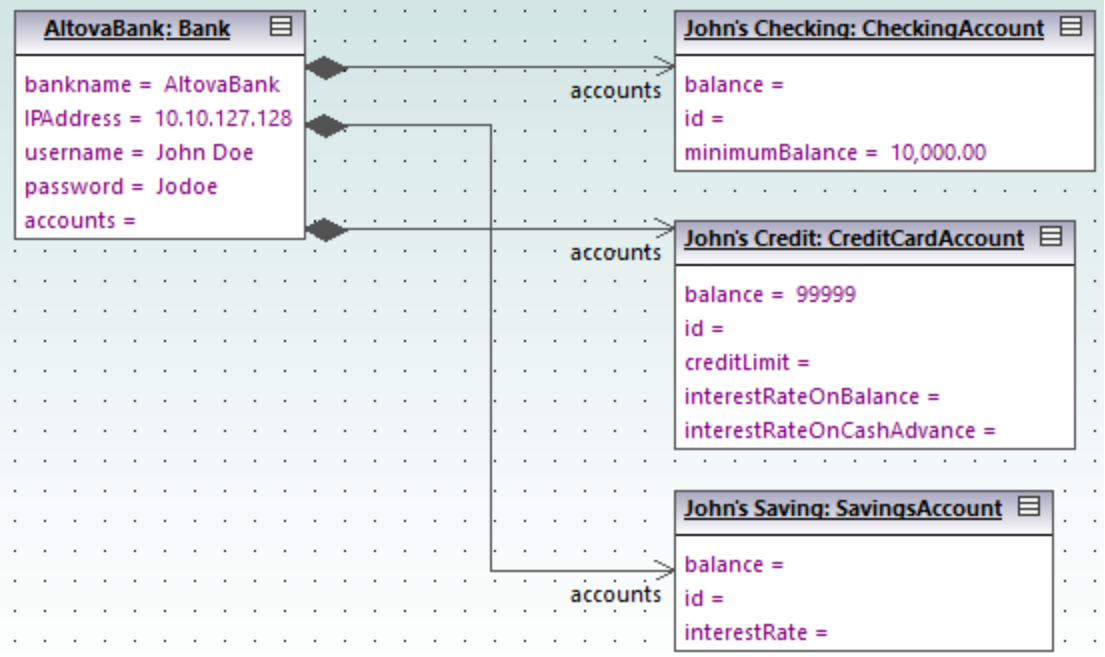
Objektbeziehungen sind die Instanzen von Klassenassoziationen und beschreiben die Assoziationen zwischen Objekten/Instanzen zu einem bestimmten Zeitpunkt.

1. Klicken Sie auf die bestehende Objektbeziehung (Assoziation) zwischen dem Objekt `AltovaBank` und dem Objekt `John's Checking: Checking Account`.
2. Wählen Sie im Fenster **Eigenschaften** neben "Classifier" den Eintrag **Account - Bank** aus. Die Objektbeziehung ändert sich nun gemäß den Klassendefinitionen in eine Kompositions-Assoziation.



3. Klicken Sie in der Symbolleiste auf das **Instanzspezifikation**-Symbol und positionieren Sie den Cursor über das Objekt `John's Credit: CreditAccount`. Der Cursor wird nun als +-Symbol angezeigt.
4. Ziehen Sie den Cursor vom Objekt `John's Credit: CreditAccount` zu `AltovaBank: Bank`, um die beiden Objekte zu verknüpfen.
5. Wählen Sie im Fenster **Eigenschaften** neben "Classifier" den Eintrag **Account - Bank** aus.
5. Erstellen Sie, wie oben beschrieben, eine Objektbeziehung zwischen dem Objekt `AltovaBank: Bank` und dem Objekt `John's Saving: SavingsAccount`.

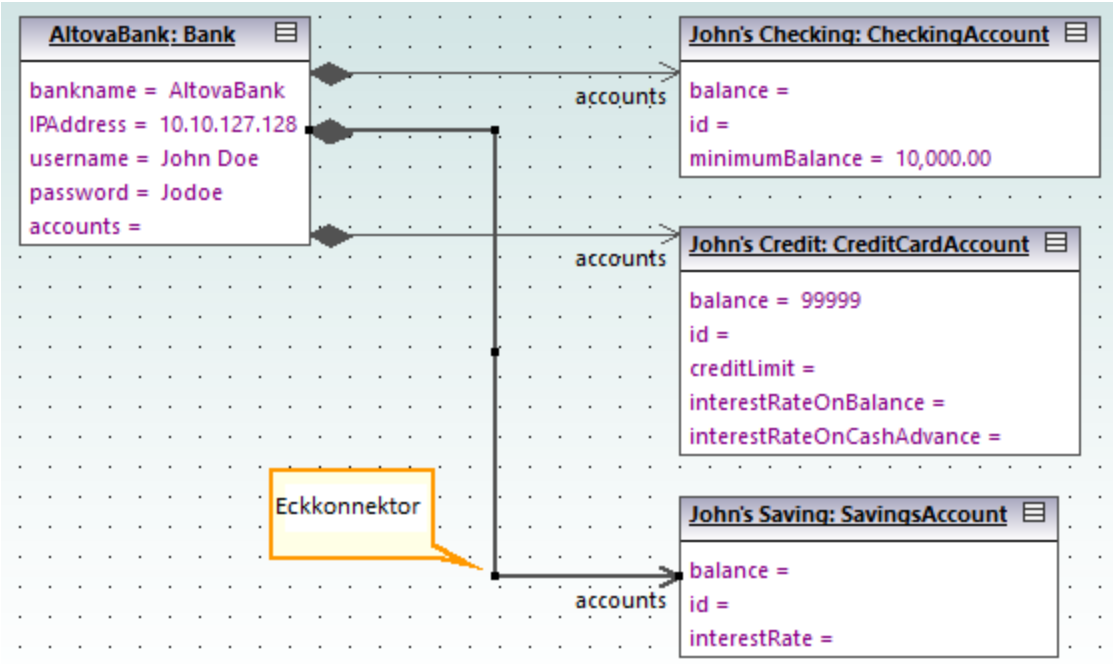
© 2016-2022 Altova GmbH



Beachten Sie, dass Änderungen, die in einem Klassendiagramm am Assoziationstyp vorgenommen werden, automatisch im Objektdiagramm aktualisiert werden.

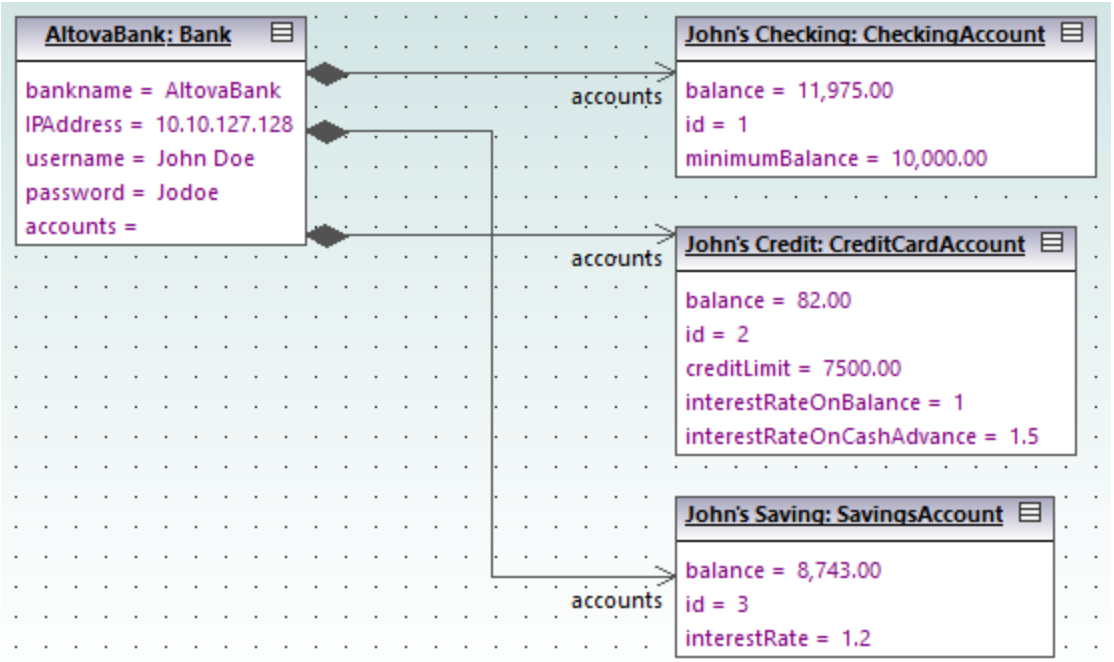
Formatieren von Assoziations-/Objektbeziehungslinien in einem Diagramm

Um Objektbeziehungen zwischen Objekten zu formatieren, platzieren Sie den Cursor auf die Linie und ziehen Sie sie an die gewünschte Position. Um die Linie horizontal und vertikal zu verschieben, ziehen Sie den Eckkonnektor, wie unten gezeigt.



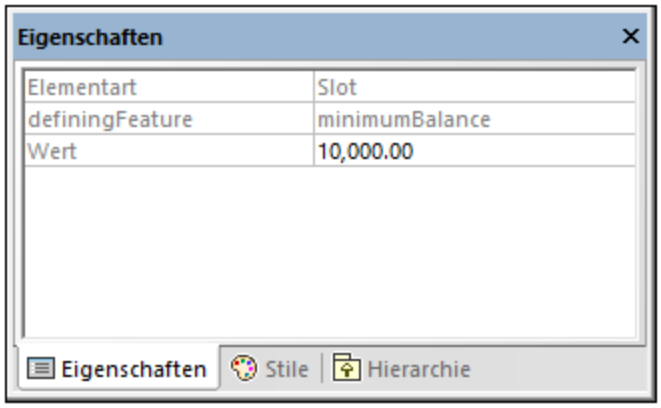
Eingabe von Beispieldaten in Objekte

Der Instanzwert eines Attributs/einer Eigenschaft in einem Objekt wird *Slot* genannt. Um den Zustand eines Objekts zu beschreiben, doppelklicken Sie auf die Slots und geben Sie nach dem Zeichen "=" Beispielinstanzdaten ein, z.B.:



 © 2016-2022 Altova GmbH

Objekt-Slots können auch über das Fenster Eigenschaften befüllt werden, indem Sie das Objekt auswählen und den entsprechenden Text neben "Wert" eingeben, z.B.:



2.5 Komponentendiagramme

In diesem Tutorialabschnitt werden die folgenden Aufgaben erläutert:

- Erstellen von Realisierungsbeziehungen zwischen Klassen und Komponenten
- Ändern der Darstellung von im Diagramm verwendeten Linien
- Hinzufügen von Verwendungsbeziehungen zu einer Schnittstelle
- Verwendung der "Ball-and-Socket"-Schnittstellen-Notation

Um fortzufahren, starten Sie UModel und öffnen Sie das Projekt **BankView-start.ump** (siehe auch Öffnen des Tutorial-Projekts 15). Das Projekt enthält eine Reihe von vordefinierten Objektdiagrammen, anhand derer die oben beschriebenen Aufgaben demonstriert werden. Es wird davon ausgegangen, dass Sie den Tutorial-Abschnitt Erstellen von abgeleiteten Klassen 34 bereits durchgearbeitet haben und die Klasse `SavingsAccount` bereits erstellt haben.

Erstellen von Realisierungsbeziehungen zwischen Klassen und Komponenten

Erweitern Sie im Fenster **Diagramm-Struktur** den Eintrag "Komponentendiagramme" und doppelklicken Sie auf das "BankView realization"-Diagrammsymbol. Dieses Diagramm enthält bereits die Komponente `BankView` sowie eine Reihe von Klassen, die über Abhängigkeiten vom Typ "Komponentenrealisierung" damit verbunden sind. Der Text "von bankview" innerhalb der einzelnen Klassen gibt den Namen des Pakets an, zu dem die Klasse gehört.

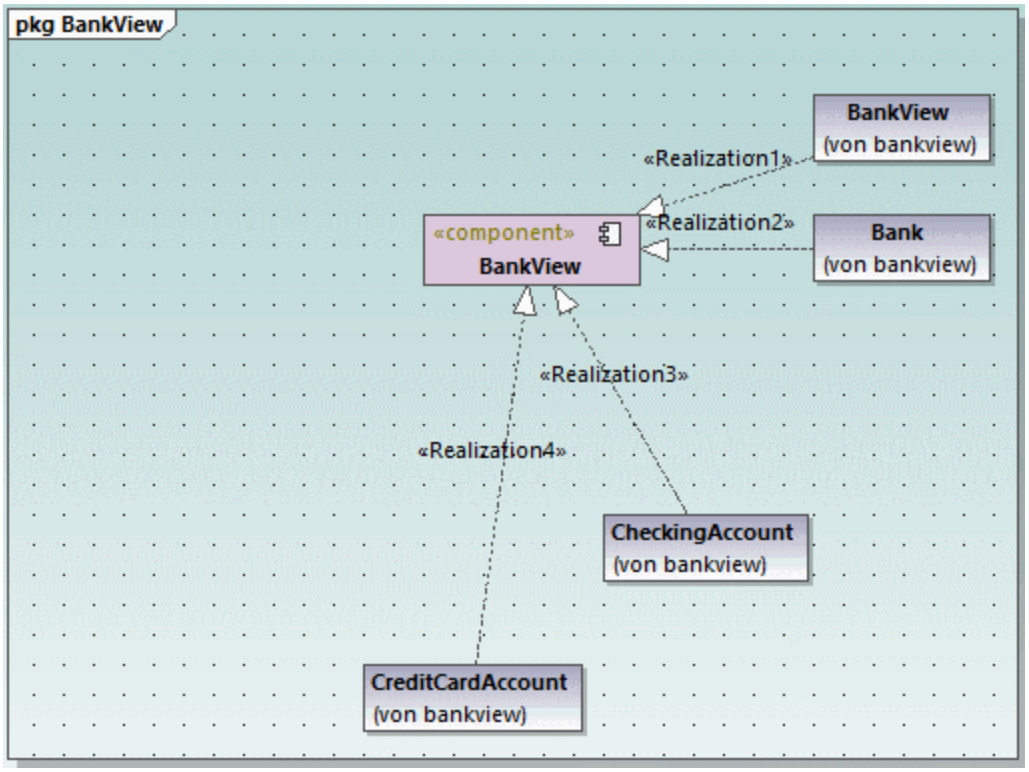


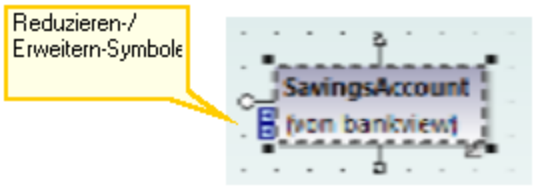
Diagramm "Bank View realization"

Fügen wir nun eine neue Klasse zum Diagramm sowie eine Realisierungsbeziehung zwischen der neuen Klasse und der Komponente `BankView` hinzu.

 © 2016-2022 Altova GmbH

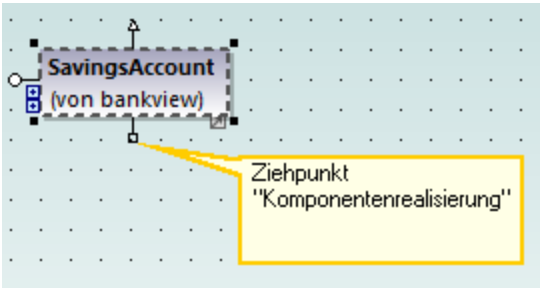
1. Gehen Sie im Fenster **Modell-Struktur** zur Klasse `SavingsAccount` im Paket `bankview`. Falls diese Klasse fehlt, gehen Sie vor, wie im Tutorial-Abschnitt Erstellen von abgeleiteten Klassen 34 beschrieben, um diese zuerst zu erstellen.
2. Ziehen Sie die Klasse `SavingsAccount` aus der **Modell-Struktur** in das Diagramm.

Standardmäßig wird die Klasse so angezeigt, dass alle Bereiche erweitert sind. Klicken Sie auf die Erweitern/Reduzieren-Symbole links von der Klasse, um Eigenschaften und Operationen ein- oder auszublenden.

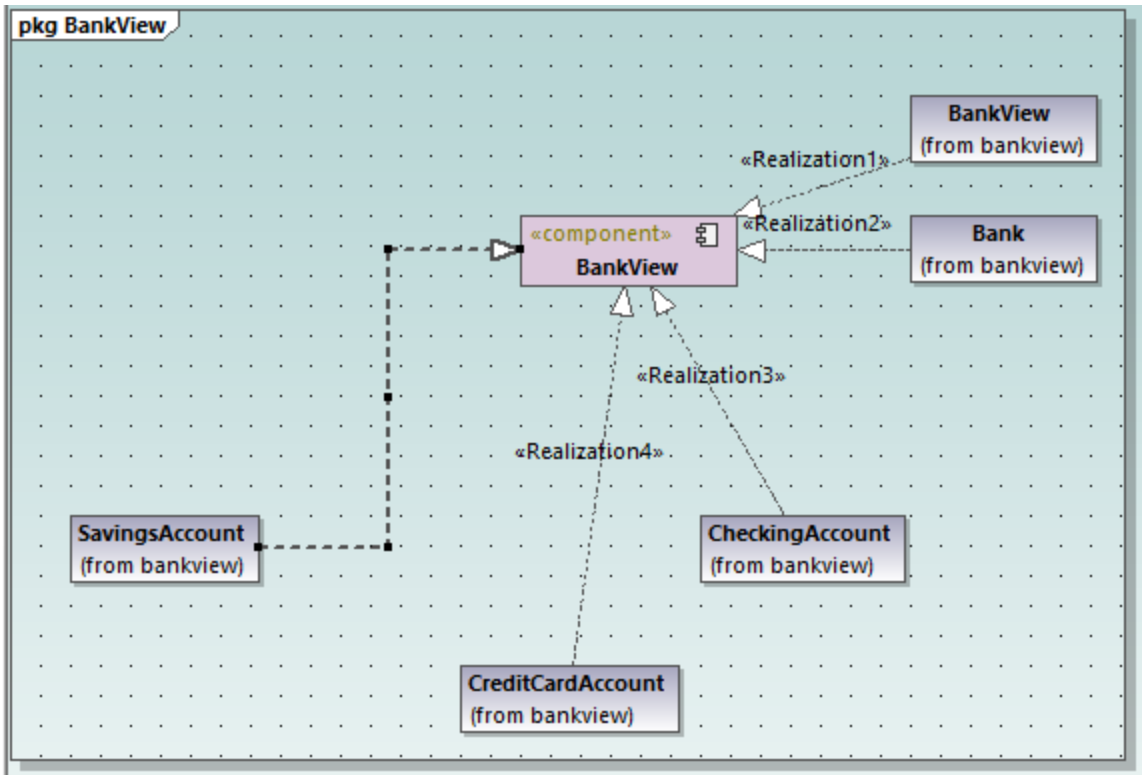


Wählen Sie eine der folgenden Methoden, um eine Realisierungsbeziehung zwischen der Klasse und der Komponente zu erstellen:

- Klicken Sie auf die Symbolleisten-Schaltfläche **Komponentenrealisierung** und ziehen Sie die Maus von der Klasse `SavingsAccount` zur Komponente `BankView`.
- Bewegen Sie den Cursor über den Ziehpunkt "Komponentenrealisierung" der Klasse und ziehen Sie ihn auf die Komponente **BankView**.



Die Realisierungsbeziehung zwischen `SavingsAccount` und `BankView` wurde nun erstellt.



Um der neuen Abhängigkeitslinie einen Namen zu geben (z.B., "Realization5"), wählen Sie zuerst die Linie aus und geben Sie dann den Namen direkt ein. Alternativ dazu können Sie die Linie auswählen und dann die Eigenschaft **Name** im Fenster **Eigenschaften** bearbeiten.

Ändern der Darstellung von Diagrammlinien

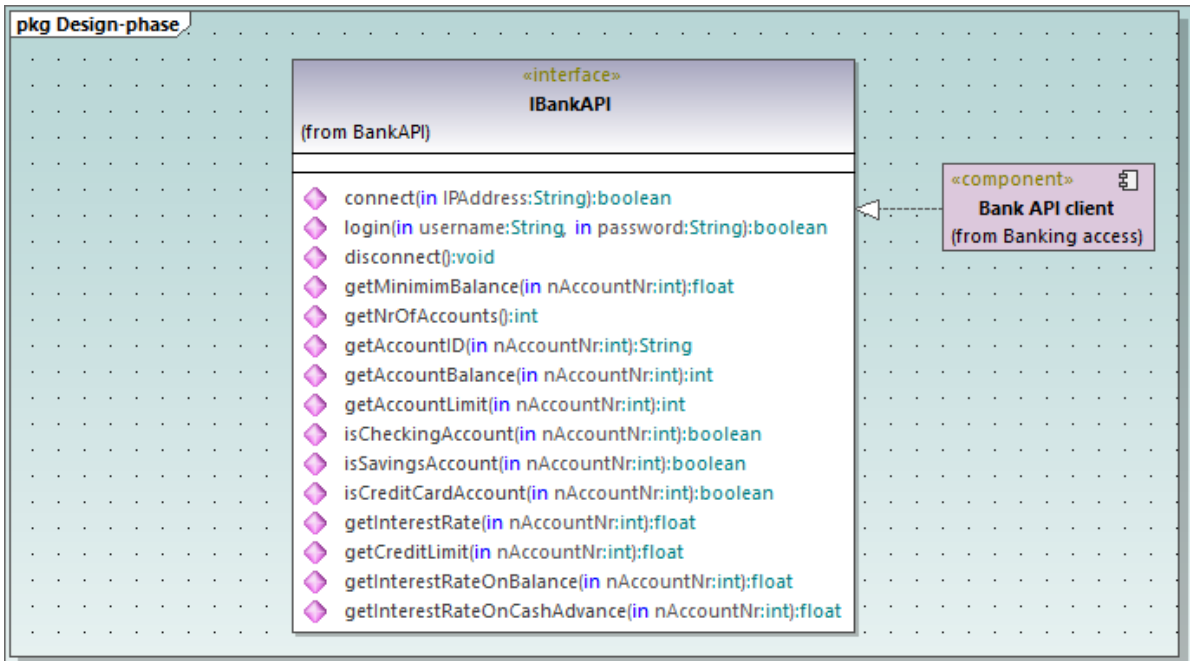
Wir werden nun die Liniendarstellung folgendermaßen von "gebogen" in "direkte Linie" ändern:

1. Wählen Sie die zuvor erstellte Linie aus (d.h. die Linie zwischen `SavingsAccount` und `BankView`).
2. Klicken Sie auf die Symbolleisten-Schaltfläche **Gerade Linie** .

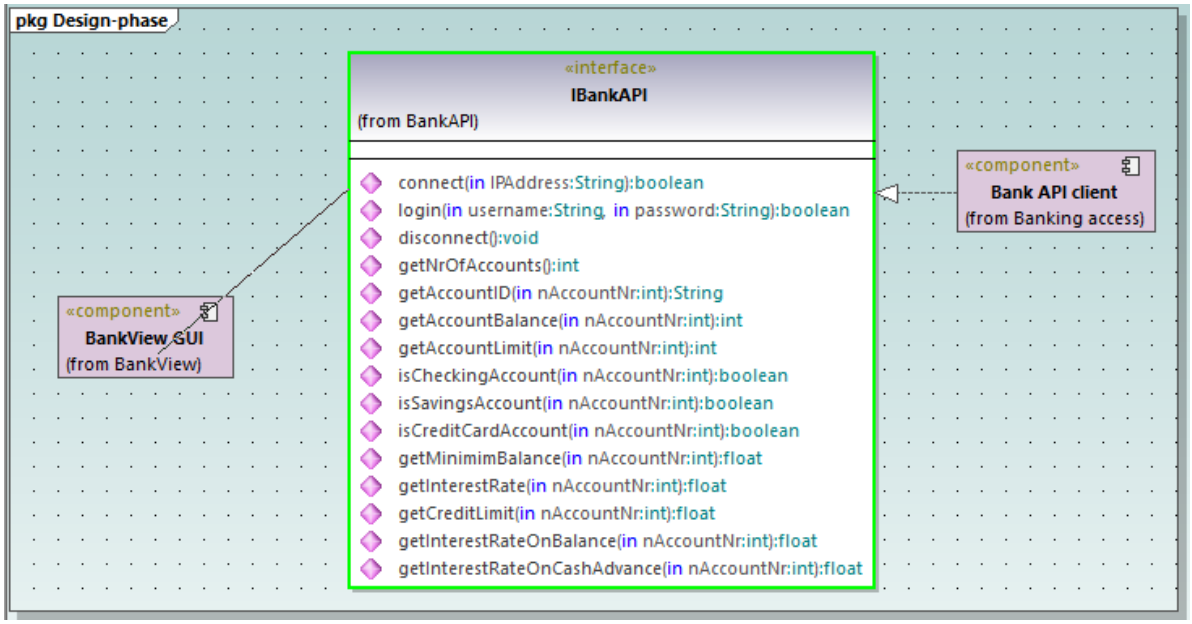
Hinzufügen von Verwendungsbeziehungen zu einer Schnittstelle

1. Navigieren Sie im Fenster **Modell-Struktur** zu **Root | Design-phase** und doppelklicken Sie auf das Symbol neben dem Diagramm "Overview". Daraufhin wird das Komponentendiagramm "Overview" geöffnet. Die gerade definierten Systemabhängigkeiten zwischen Komponenten und Schnittstellen werden darin angezeigt.

 © 2016-2022 Altova GmbH

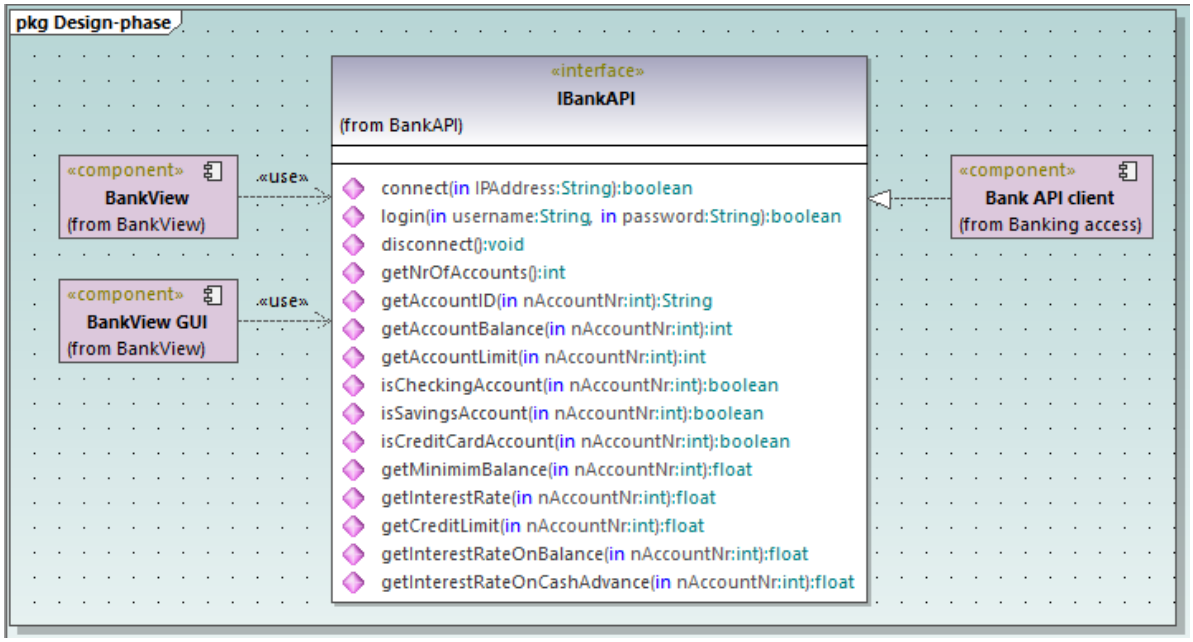


2. Navigieren Sie im Fenster **Modell-Struktur** zu **Root | Component View | BankView** und ziehen Sie das Paket `BankView GUI` in das Diagramm.
3. Ziehen Sie auch das Paket `BankView` in das Diagramm.
4. Klicken Sie auf die Symbolleisten-Schaltfläche **Verwendung** und ziehen Sie die Maus vom Paket `BankView GUI` zur Schnittstelle `IBankAPI`.



5. Wiederholen Sie den vorhergehenden Schritt für das Paket `BankView`.

Wie unten gezeigt, haben beide Pakete nun eine Verwendungsbeziehung zur Schnittstelle. Für die Pakete `BankView` und `BankView GUI` wird die Schnittstelle `IBankAPI` benötigt. Sie stellt wie für das Paket `Bank API Client` die Schnittstelle bereit.

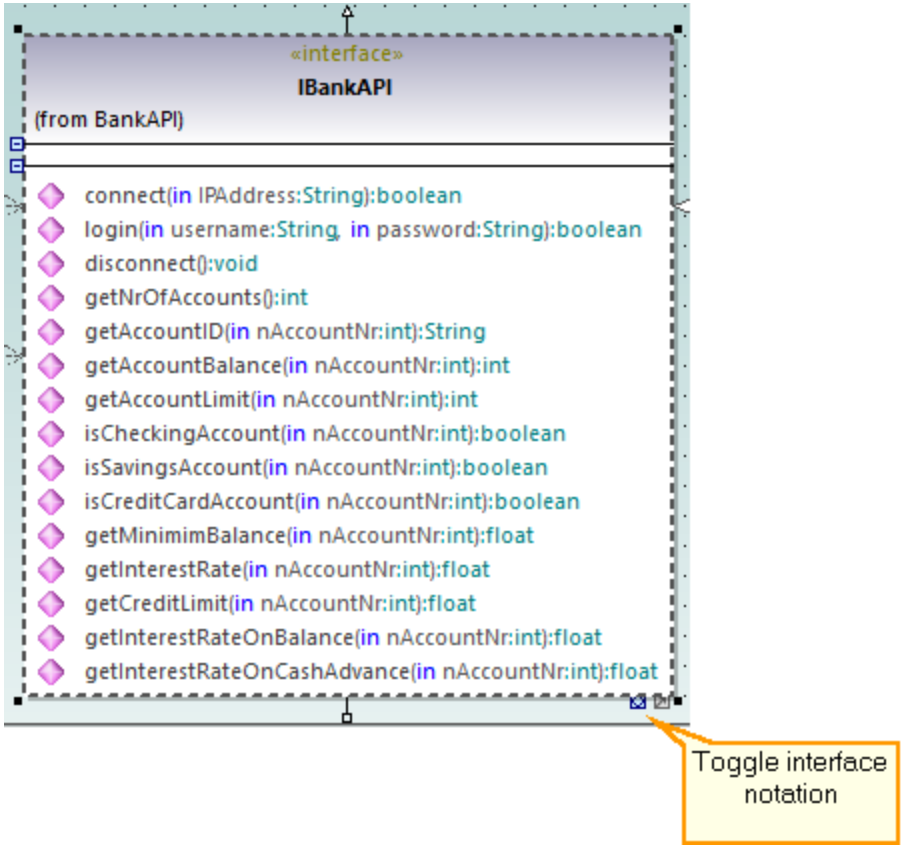


Verwendung der "Ball-and-Socket"-Notation

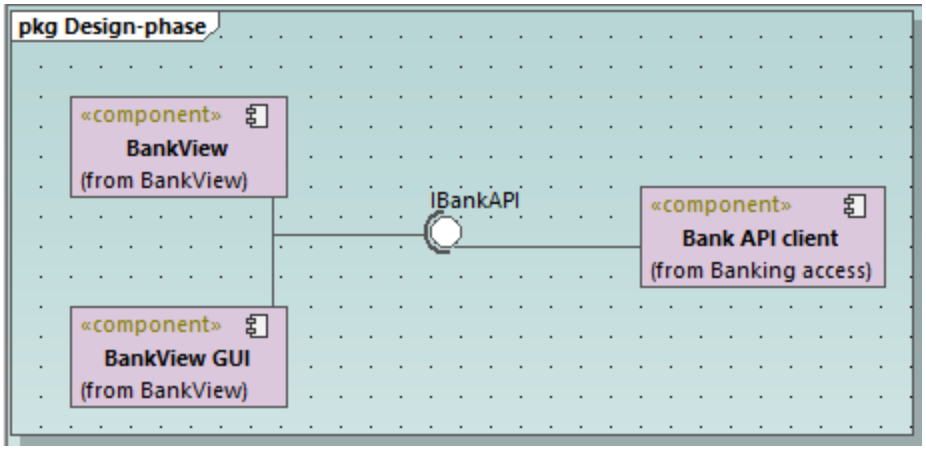
Sie haben optional die Möglichkeit, die aktuelle Diagrammnotation in "Ball-and-Socket"-Notation zu konvertieren. Gehen Sie dazu folgendermaßen vor:

- Wählen Sie die Schnittstelle aus und klicken Sie anschließend in ihrer rechten unteren Ecke auf die Schaltfläche **Darstellung wechseln**.

© 2016-2022 Altova GmbH



Das Diagramm wird nun in der "Ball-and-Socket"-Notation dargestellt.



Um wieder zurück zum vorherigen Notationsstil zu wechseln, wählen Sie die Schnittstelle aus und klicken Sie anschließend wieder auf die Schaltfläche **Darstellung wechseln**.

2.6 Deployment-Diagramme

In diesem Tutorialabschnitt werden die folgenden Aufgaben erläutert:

- Hinzufügen einer Abhängigkeit zwischen zwei Artefakten in einem Deployment-Diagramm
- Hinzufügen von Elementen zu einem Deployment-Diagramm
- Einbetten von Artefakten in einen Knoten in einem Deployment-Diagramm
- Erstellen von Artefakt-Elementen (z.B. Eigenschaften, Operationen, geschachtelten Artefakten)

Um fortzufahren, starten Sie UModel und öffnen Sie das Projekt **BankView-start.ump** (siehe auch Öffnen des Tutorial-Projekts 15).

Hinzufügen einer Abhängigkeit zwischen zwei Artefakten in einem Deployment-Diagramm

Doppelklicken Sie im Fenster Diagramm-Struktur unter "Deployment-Diagramme" auf das Symbol neben dem Diagramm "Artifacts", um es zu öffnen. Wie unten gezeigt, sehen Sie in diesem Diagramm die Manifestation der Komponenten `Bank API client` und `BankView` zu ihren entsprechenden kompilierten Java .jar-Dateien.

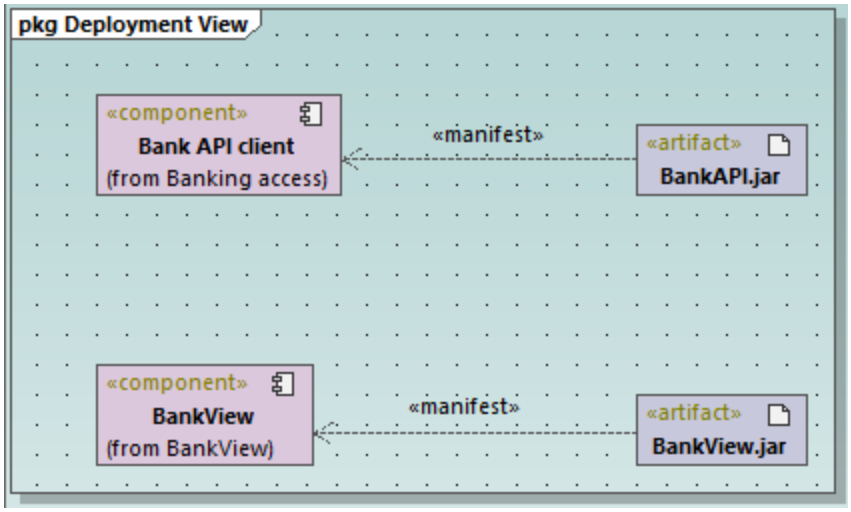


Diagramm "Artifacts"

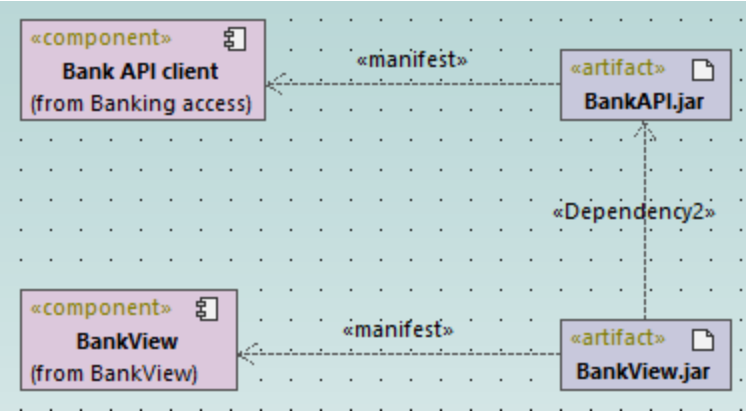
Diese Manifestationen wurden auf ähnliche Weise wie die anderen, zuvor in diesem Tutorial beschriebenen Beziehungen erstellt:

1. Klicken Sie auf die Symbolleisten-Schaltfläche **Manifestation** .
2. Bewegen Sie den Mauszeiger über das Artefakt und ziehen Sie ihn in die Komponente.

Fügen wir nun auf dieselbe Art eine Abhängigkeit zwischen den beiden .jar-Dateien hinzu:

1. Klicken Sie auf die Symbolleisten-Schaltfläche **Abhängigkeit** .
2. Bewegen Sie den Mauszeiger über das Artefakt `BankView.jar` und ziehen Sie ihn in das Artefakt `BankAPI.jar`.
3. Wählen Sie die Abhängigkeitslinie aus und geben Sie "Dependency2" ein.

 © 2016-2022 Altova GmbH



Hinzufügen von Elementen zu einem Deployment-Diagramm

Doppelklicken Sie im Fenster **Diagramm-Struktur** unter "Deployment-Diagramme" auf das Symbol neben dem Diagramm "Deployment", um es zu öffnen. Dieses Diagramm wurde absichtlich unfertig gelassen und besteht aus einem einzigen Knoten, der einen Heim-PC darstellt. In den folgenden Schritten werden wir weitere Elemente zu diesem Diagramm hinzufügen.

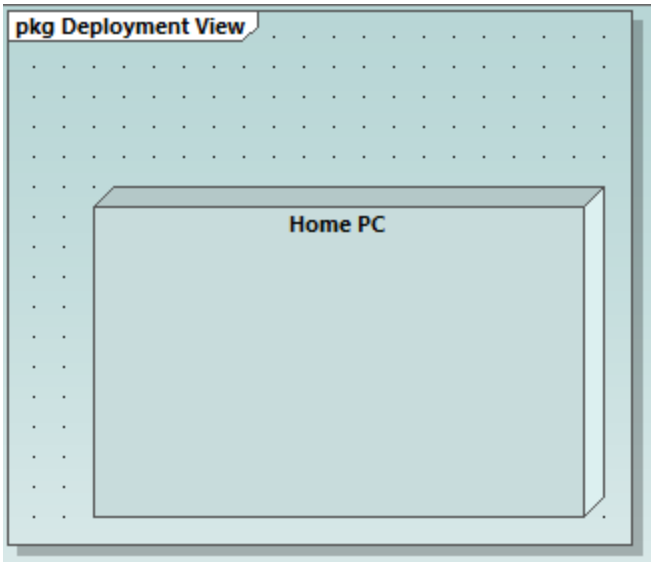
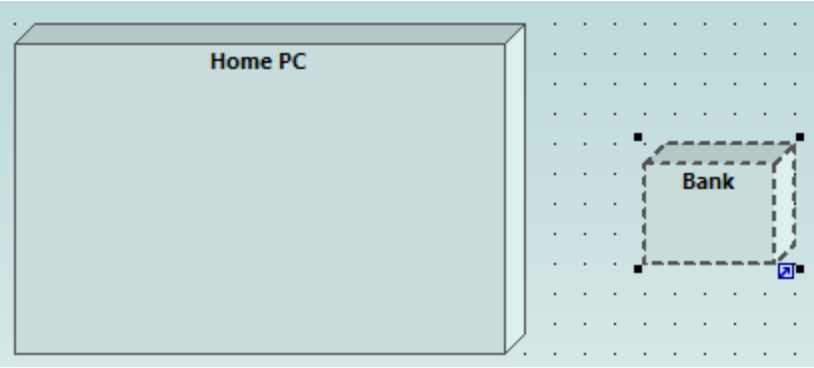


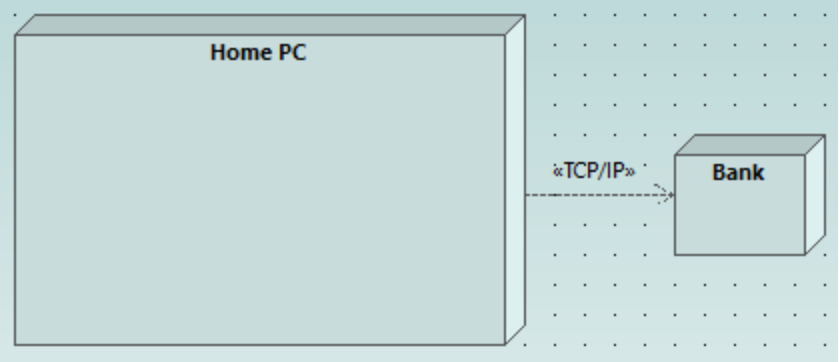
Diagramm "Deployment"

Angenommen, Sie möchten eine TCP/IP-Verbindung zwischen dem Heim-PC und einer Bank darstellen. Fügen wir also nun die erforderlichen Elemente hinzu:

1. Klicken Sie auf die Symbolleisten-Schaltfläche **Knoten** und klicken Sie an eine Stelle rechts vom Knoten "Home PC", um den Knoten einzufügen.
2. Benennen Sie den Knoten um in "Bank" und vergrößern Sie ihn durch Ziehen einer seiner Kanten.



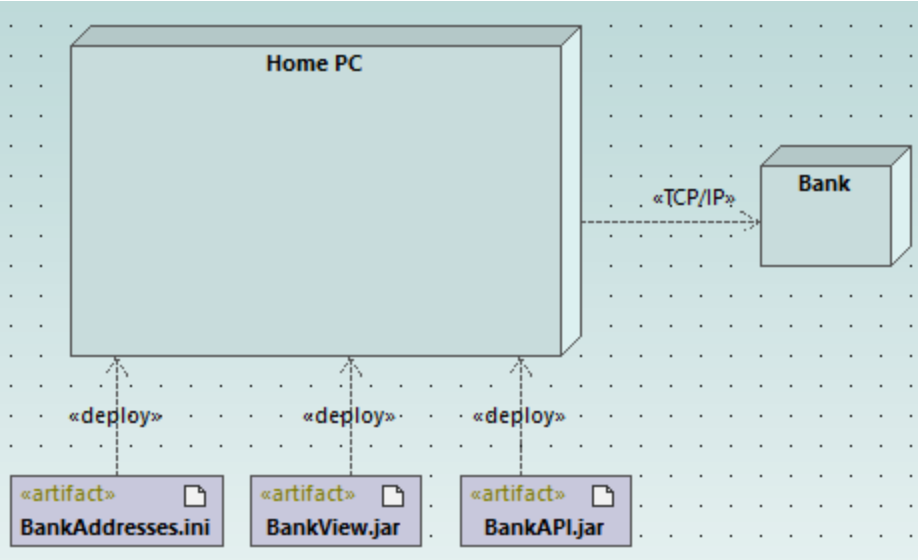
3. Klicken Sie auf die Symbolleisten-Schaltfläche **Abhängigkeit** und ziehen Sie die Abhängigkeit vom Knoten "Home PC" zum Knoten "Bank". Dadurch wird eine Abhängigkeit zwischen den beiden Knoten erstellt.
4. Wählen Sie die Abhängigkeitslinie aus und geben Sie als Namen der neuen Abhängigkeit "TCP/IP" ein. (Bearbeiten Sie alternativ dazu die Eigenschaft **Name** im Fenster **Eigenschaften**).



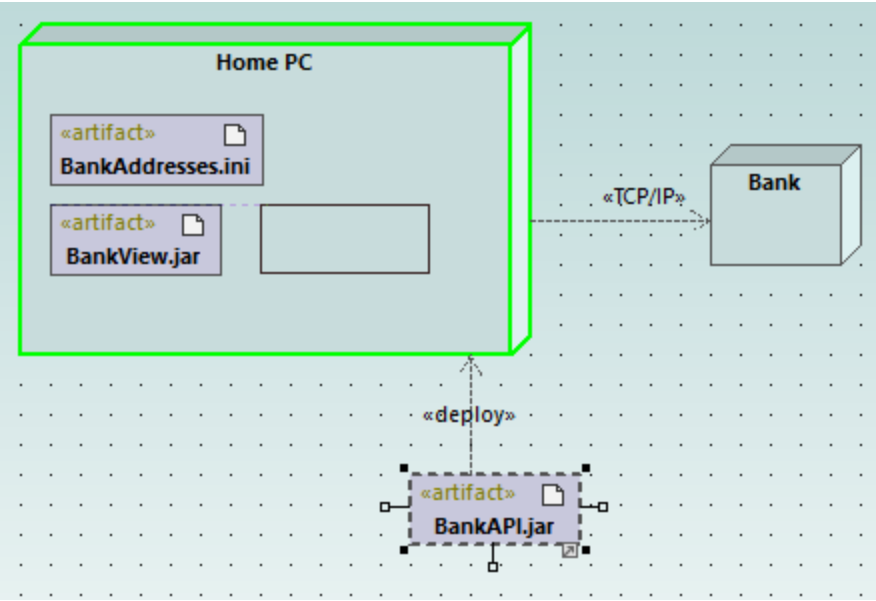
Einbetten von Artefakten

Erweitern Sie im Fenster **Modell-Struktur** das Paket "Deployment View" und ziehen Sie anschließend alle der folgenden Artefakte in das Diagramm: **BankAddresses.ini**, **BankAPI.jar** und **BankView.jar**. Das Projekt wurde so vorkonfiguriert, dass es Deploy-Abhängigkeiten zwischen diesen Artefakten und dem Knoten "HomePC" enthält, daher werden alle diese Abhängigkeiten im Diagramm nun angezeigt:

 © 2016-2022 Altova GmbH

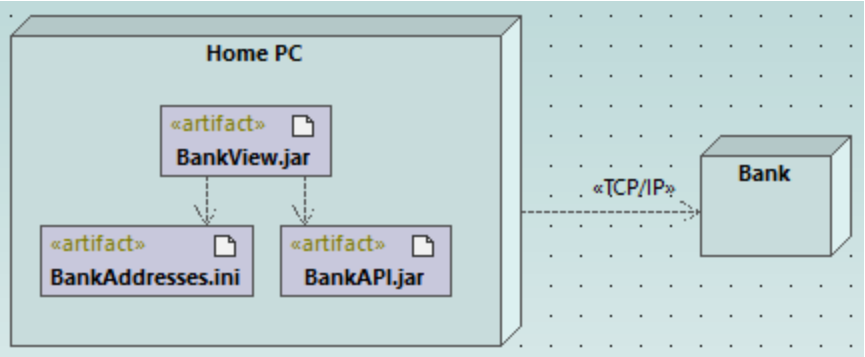


Sie können die Artefakte auch durch Ziehen der einzelnen Artefakte in den Knoten "Home PC" einbetten. Beachten Sie, dass die Deploy-Abhängigkeiten im Diagramm nicht mehr sichtbar sind, obwohl sie logisch weiterhin bestehen.



In den Knote eingebettete Artefakte können auch untereinander Abhängigkeiten aufweisen. Tun Sie zur Veranschaulichung Folgendes:

1. Klicken Sie auf die Symbolleisten-Schaltfläche **Abhängigkeit** und ziehen Sie die Maus bei gedrückter Strg-Taste vom Artefakt "BankView.jar" in das Artefakt "BankAddresses.ini".
2. Ziehen Sie die Maus vom Artefakt "BankView.jar" in das Artefakt "BankAPI.jar", während Sie die Strg-Taste gedrückt halten.



Anmerkung: Wenn Sie ein Artefakt aus einem Knoten in ein Diagramm ziehen, wird immer automatisch eine Deployment-Abhängigkeit erstellt.

Erstellen von Artefaktelementen (Eigenschaften, Operationen, geschachtelten Artefakten)

Artefakte können in UML aus Eigenschaften, Operationen und anderen Elementen, wie z.B. geschachtelten Artefakten, bestehen. Um solche geschachtelten Elemente zu erstellen, klicken Sie mit der rechten Maustaste im Fenster **Modell-Struktur** auf das Artefakt und wählen Sie im Kontextmenü die entsprechende Aktion aus (z.B. **Neues Element | Operation** oder **Neues Element | Eigenschaft**). Das neue Element erscheint im Fenster **Modell-Struktur** geschachtelt unterhalb des ausgewählten Artefakts.

 © 2016-2022 Altova GmbH

2.7 Forward Engineering (Modell zu Code)

In diesem Beispiel wird gezeigt, wie Sie ein neues UModel-Projekt erstellen und Programmcode anhand dieses Projekts generieren (ein Prozess, der als "Forward Engineering" bezeichnet wird). Der Einfachheit halber besteht das Projekt aus nur einer Klasse. Außerdem lernen Sie, wie Sie das Projekt für die Codegenerierung vorbereiten und sicher stellen, dass im Projekt die richtige Syntax verwendet wird. Nachdem Sie Programmcode generiert haben, werden Sie diesen außerhalb von UModel ändern, indem Sie eine neue Methode zur Klasse hinzufügen. Zum Abschluss werden Sie die Codeänderungen wieder im ursprünglichen UModel-Projekt zusammenführen (ein als "Reverse Engineering" bekannter Prozess).

In diesem Tutorial wird als Codegenerierungssprache Java verwendet. Die Vorgangsweise ist aber auch bei anderen Codegenerierungssprachen ähnlich.

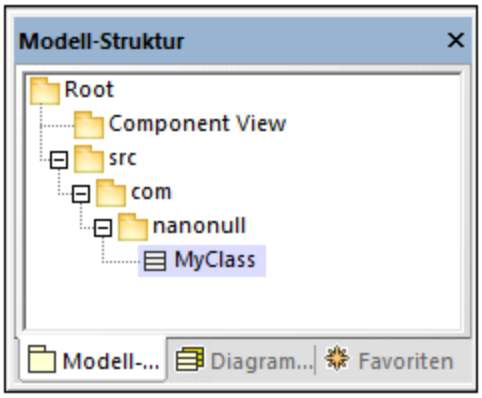
Erstellen eines neue UModel-Projekts

Sie können folgendermaßen ein neues UModel-Projekt erstellen:

- Klicken Sie im Menü **Datei** auf **Neu**. (Oder drücken Sie alternativ dazu **Strg+N** oder klicken Sie auf die Symbolleisten-Schaltfläche "Neu" .)

Das Projekt enthält zu diesem Zeitpunkt nur die Standardpakete "Root" und "Component View". Diese beiden Pakete können nicht gelöscht oder umbenannt werden. "Root" bildet die oberste Hierarchieebene für alle anderen Pakete und Elemente im Projekt. "Component View" wird für das Code Engineering benötigt; normalerweise enthält es eine oder mehrere UML-Komponenten, die von den Klassen oder Schnittstellen Ihres Projekts realisiert werden; wir haben zu diesem Zeitpunkt jedoch noch keine Klassen erstellt. Erstellen wir also zuerst die Struktur unseres Programms. Gehen Sie dazu folgendermaßen vor:

1. Klicken Sie mit der rechten Maustaste im Fenster "Modell-Struktur" auf das Paket "Root" und wählen Sie im Kontextmenü den Befehl **Neues Element | Paket**. Benennen Sie das neue Paket in "src" um.
2. Klicken Sie mit der rechten Maustaste auf "src" und wählen Sie im Kontextmenü den Befehl **Neues Element | Paket.** Benennen Sie das neue Paket in "com" um.
3. Klicken Sie mit der rechten Maustaste auf "com" wählen Sie im Kontextmenü den Befehl **Neues Element | Paket**. Benennen Sie das neue Paket in "nanonull" um.
4. Klicken Sie mit der rechten Maustaste auf "nanonull" wählen Sie im Kontextmenü den Befehl **Neues Element | Klasse**. Benennen Sie die neue Klasse in "MyClass" um.



Vorbereiten des Projekts für die Codegenerierung

Um Code anhand eines UModel-Modells zu generieren, müssen die folgenden Voraussetzungen gegeben sein:

- Es muss ein C#- oder VB.NET-Namespace-Root-Pakte definiert sein.
- Es muss eine Komponente geben, die von allen Klassen oder Schnittstellen, für die Code generiert werden muss, realisiert wird.
- Der Komponente muss ein physischer Pfad (Verzeichnis) zugewiesen worden sein. Der Code wird in diesem Verzeichnis generiert.
- Für die Komponente muss die Eigenschaft **für Code Engineering verwenden** aktiviert sein.

Alle diese Anforderungen werden weiter unten ausführlicher beschrieben. Beachten Sie, dass Sie jederzeit überprüfen können, ob das Projekt alle Voraussetzungen für die Codegenerierung erfüllt, indem Sie es validieren:

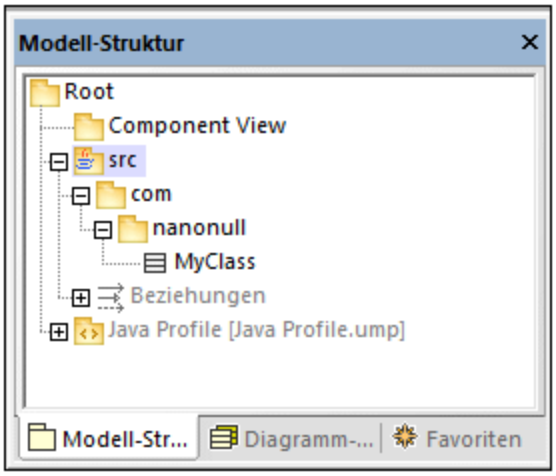
- Klicken Sie im Menü **Projekt** auf **Projektsyntax überprüfen**. (Drücken Sie alternativ dazu **F11**.)

Wenn Sie das Projekt in dieser Phase validieren, wird im Fenster "Meldungen" ein Validierungsfehler angezeigt ("*Es wurde keine Namespace Root gefunden! Verwenden Sie bitte das Kontextmenü in der Modell-Struktur, um ein Paket als Namespace Root zu definieren*"). Um diesen Fehler zu beheben, weisen Sie das Paket "src" als Namespace Root zu:

- Klicken Sie mit der rechten Maustaste auf das Paket "src" und wählen Sie im Kontextmenü den Befehl **Code Engineering | Als Java Namespace Root definieren**.
- Wenn Sie informiert werden, dass das UModel Java-Profil inkludiert wird, klicken Sie auf **OK**.

Beachten Sie, dass sich das Paketsymbol nun in das Symbol geändert hat, was bedeutet, dass es sich beim Paket um eine Java Namespace Root handelt. Zusätzlich dazu wurde ein Java-Profil zum Projekt hinzugefügt.

 © 2016-2022 Altova GmbH



Der eigentliche Namespace kann folgendermaßen definiert werden:

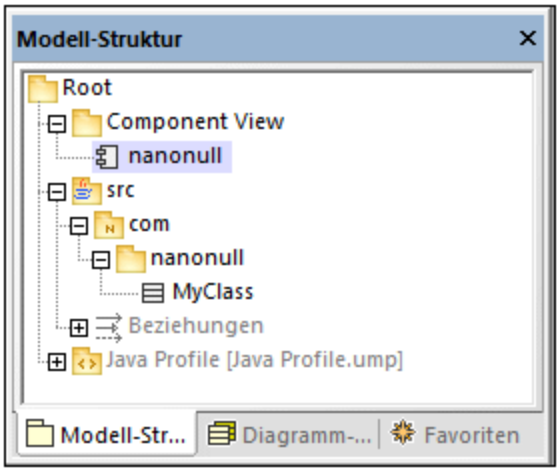
1. Wählen Sie das Paket "com" im Fenster **Modell-Struktur** aus.
2. Aktivieren Sie im Fenster **Eigenschaften** die Eigenschaft **<<namespace>>**.

3. Wiederholen Sie die obigen Schritte für das Paket "nanonull".

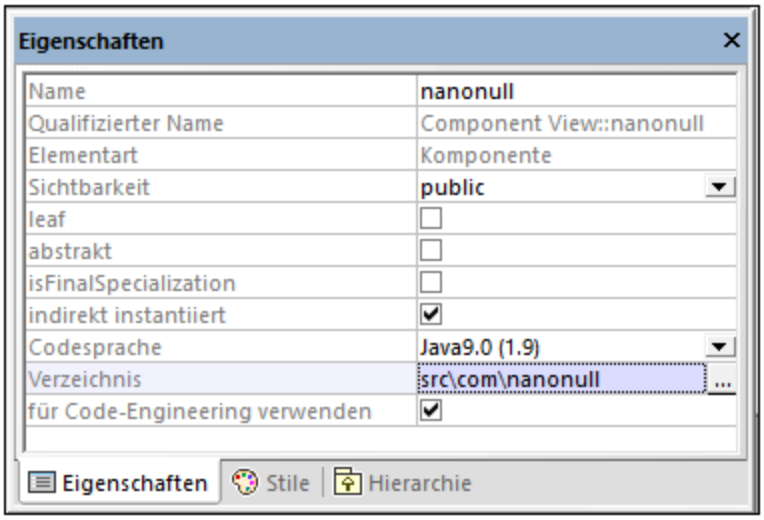
Beachten Sie, dass sich das Symbol der beiden Pakete "com" und "nanonull" nun in geändert hat, was bedeutet, dass es sich nun um einen Namespace handelt.

Eine weitere Voraussetzung für die Codegenerierung ist, dass eine Komponente von mindestens einer Klasse oder Schnittstelle realisiert werden muss. Eine Komponente ist in UML ein Teilstück des Systems. In UModel können Sie über die Komponente das Verzeichnis für die Codegenerierung und andere Einstellungen definieren. Andernfalls wäre die Codegenerierung nicht möglich. Wenn Sie das Projekt zu diesem Zeitpunkt validieren, wird im Fenster **Meldungen** eine Warnung angezeigt: "*MyClass hat keine Komponentenrealisierung zu einer Komponente - es wird kein Code generiert*". Um diesen Fehler zu beheben, muss eine Komponente zum Projekt hinzugefügt werden. Gehen Sie folgendermaßen vor:

1. Klicken Sie mit der rechten Maustaste im Fenster "Modell-Struktur" auf "Component View" und wählen Sie im Kontextmenü den Befehl **Neues Element | Komponente**.
2. Benennen Sie die neue Komponente in "nanonull" um.



3. Ändern Sie im Fenster **Eigenschaften** die Eigenschaft **Verzeichnis** in ein Verzeichnis, in dem der Code generiert werden soll (in diesem Beispiel, "src\com\nanonull"). Beachten Sie dass die Eigenschaft **für Code Engineering verwenden** aktiviert ist, was eine weitere Vorbedingung für die Codegenerierung ist.



4. Speichern Sie das UModel-Projekt in einem Verzeichnis und geben Sie ihm einen beschreibenden Namen (in diesem Beispiel **C:\UModelDemo\Tutorial.ump**).

Anmerkung: Der Pfad für die Codegenerierung kann absolut oder relativ zum .ump-Projekt sein. Wenn er, wie in diesem Beispiel, relativ ist, würden mit dem Pfad **src\com\nanonull** alle Verzeichnisse im selben Verzeichnis, in dem auch das UModel-Projekt gespeichert ist, erstellt werden.

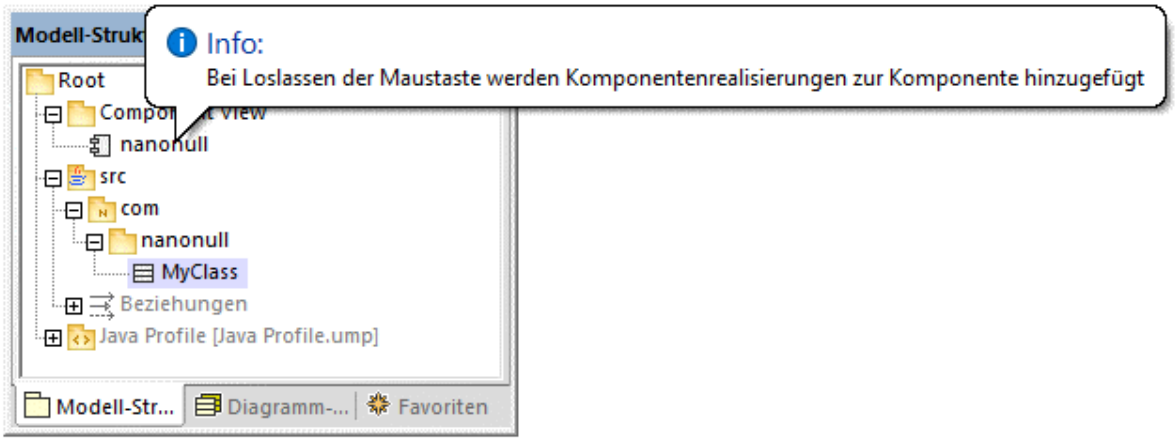
Wir haben uns absichtlich entschieden, Code in einem Verzeichnis zu generieren, das den Namespace-Namen enthält; andernfalls würde es zu Warnungen kommen. UModel zeigt standardmäßig Projektvalidierungswarnungen an, wenn die Komponente so konfiguriert ist, das Java-Code in einem Verzeichnis generiert wird, das nicht denselben Namen wie der Namespace hat. Die Komponente "nanonull" in diesem Beispiel hat den Pfad "C:\UModelDemo\src\com\nanonull", sodass es zu keinen Validierungswarnungen kommt. Wenn Sie eine ähnliche Überprüfung für C# oder VB.NET implementieren möchten oder wenn Sie die Namespace-Validierung für Java deaktivieren möchten, gehen Sie folgendermaßen vor:

 © 2016-2022 Altova GmbH

1. Klicken Sie im Menü **Extras** auf **Optionen**.
2. Klicken Sie auf das Register **Code Engineering**.
3. Aktivieren Sie unter **Namespace für Codedateipfad verwenden** das entsprechende Kontrollkästchen.

Die Komponentenrealisierungsbeziehung kann folgendermaßen erstellt werden:

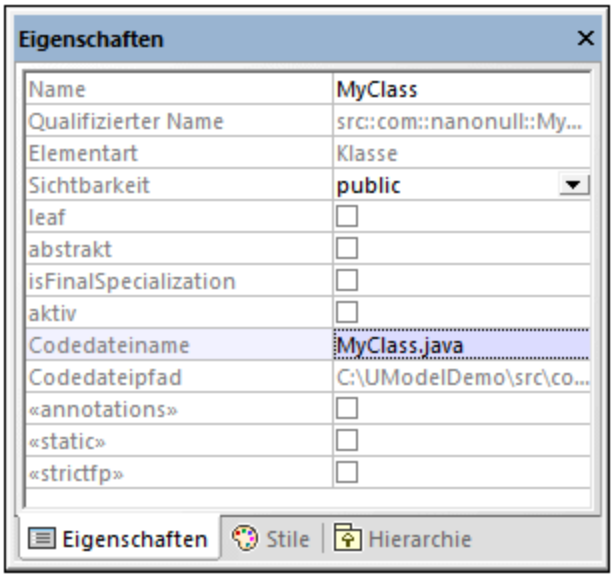
- Ziehen Sie die Maus im Fenster Modell-Struktur bei gedrückter Maustaste von der zuvor erstellten Klasse `MyClass` auf die Komponente `nanonull`.



Die Komponente wird nun von der einzigen Klasse des Projekts, nämlich `MyClass` realisiert. Beachten Sie, dass die oben beschriebene Methode nun eine von mehreren Möglichkeiten ist, die Komponentenrealisierung zu erstellen. Eine weitere Methode ist, sie, wie im Tutorialabschnitt Komponentendiagramme 48 beschrieben, von einem Komponentendiagramm aus zu erstellen.

Als nächstes wird empfohlen, den an der Codegenerierung beteiligten Klassen oder Schnittstellen einen Dateinamen zu geben. Andernfalls generiert UModel die entsprechende Datei mit einem Standarddateinamen und im Fenster **Meldungen** wird eine Warnung angezeigt ("*'Codedateiname wurde nicht definiert - es wird ein Standardname generiert.*"). Um diese Warnung zu entfernen, gehen Sie folgendermaßen vor:

1. Wählen Sie die Klasse `MyClass` im Fenster **Modell-Struktur** aus.
2. Ändern Sie die Eigenschaft **Codedateiname** im Fenster **Eigenschaften** in den gewünschten Datennamen (in diesem Beispiel, `MyClass.java`).

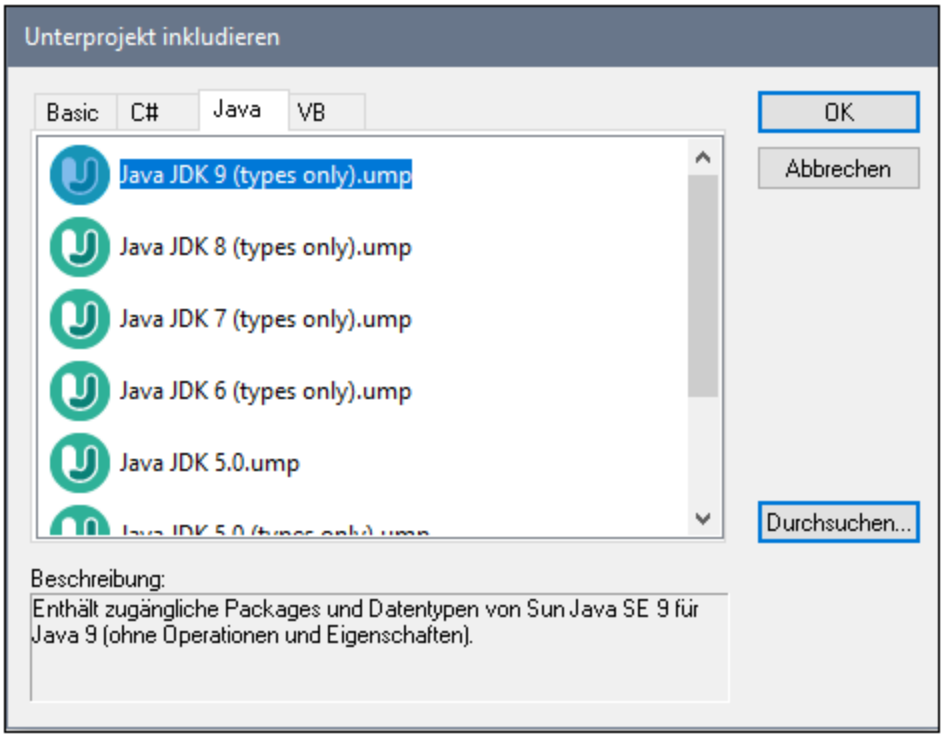


Inkludieren der JDK-Typen

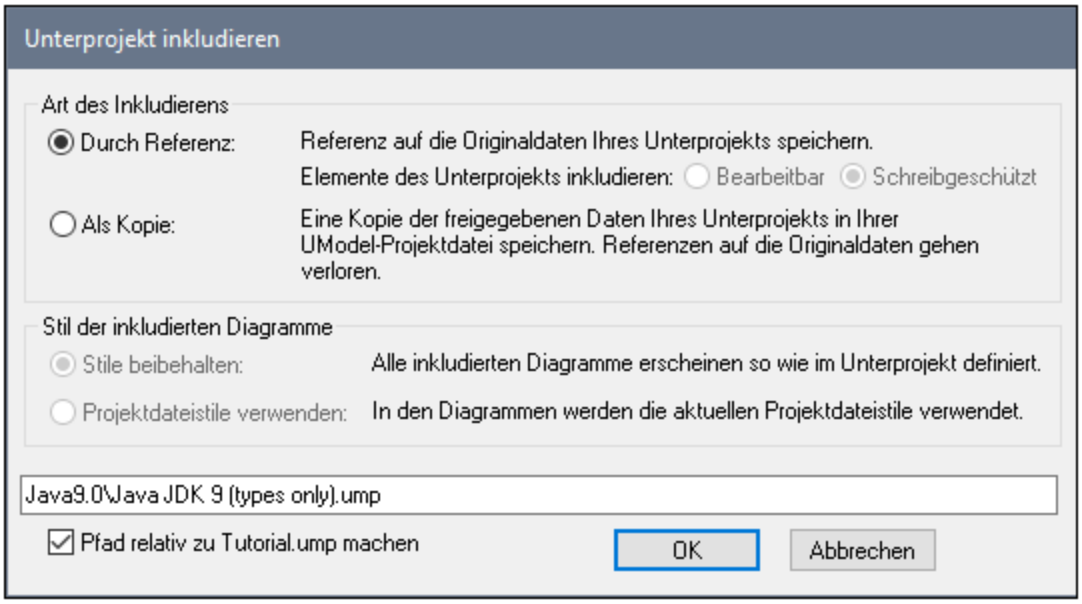
Dieser Schritt ist zwar optional, dennoch wird empfohlen, die Java Development Kit (JDK)-Sprachentypen als Unterprojekt Ihres aktuellen UModel-Projekts zu inkludieren. Andernfalls stehen die JDK-Typen beim Erstellen von Klassen oder Schnittstellen nicht zur Verfügung. Gehen Sie dazu folgendermaßen vor (die Vorgangsweise ist für C# und VB.NET ähnlich):

1. Klicken Sie im Menü **Projekt** auf **Unterprojekt inkludieren**.
2. Klicken Sie auf das Register **Java** und wählen Sie das Projekt **Java JDK 9 (types only)** aus.

© 2016-2022 Altova GmbH



3. Wenn Sie gefragt werden, ob das Projekt über eine Referenz oder als Kopie inkludiert werden soll, wählen Sie **Durch Referenz**.

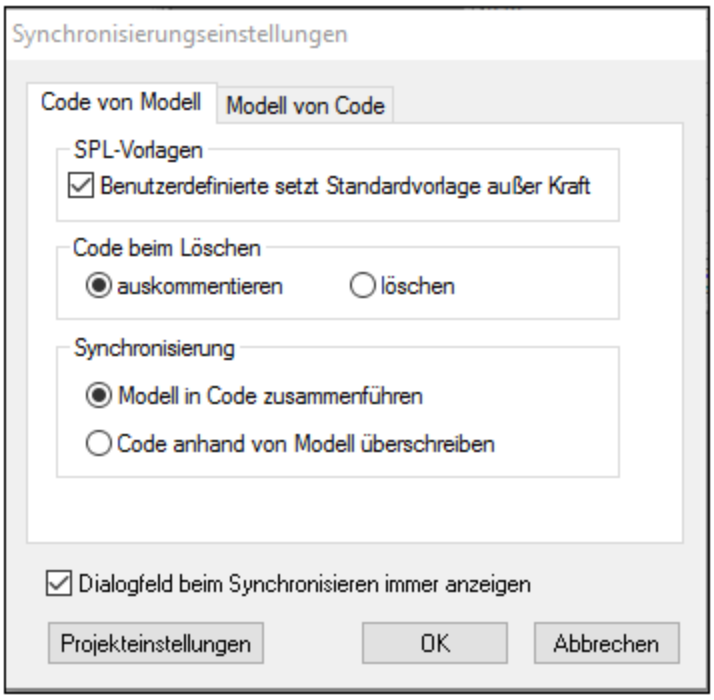


Generieren von Code

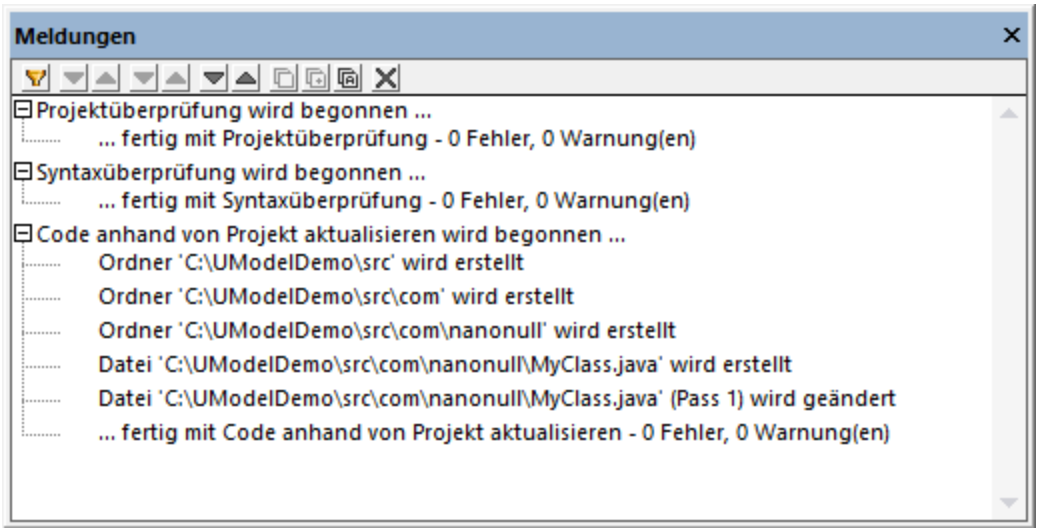
Nachdem nun alle Vorbedingungen erfüllt werden, kann Code folgendermaßen generiert werden:

1. Klicken Sie im Menü **Projekt** auf **Merge Programmcode aus UModel-Projekt**. (Drücken Sie

alternativ dazu **F12**). Beachten Sie, dass dieser Befehl den Namen **Überschreibe Programmcode aus UModel-Projekt** hat, wenn zuvor im unten gezeigten Dialogfeld "Synchronisierungseinstellungen" die Option **Code anhand von Modell überschreiben** aktiviert wurde.



2. Lassen Sie die Standardsynchronisierungseinstellungen unverändert und klicken Sie auf **OK**. Es wird automatisch eine Projektsyntaxüberprüfung durchgeführt, deren Ergebnis im Fenster **Meldungen** angezeigt wird:



Ändern von Code außerhalb von UModel

Die Generierung von Programmcode ist erst der erste Schritt bei der Entwicklung Ihrer Software-Applikation oder Ihres Systems. In einem realen Szenario würde der Code mehrmals verändert, bevor das ein Programm

 © 2016-2022 Altova GmbH

mit allen seinen Features fertig entwickelt ist. Öffnen Sie die generierte Datei **MyClass.java** zu diesem Zweck in einem Text-Editor und fügen Sie eine neue Methode zur Klasse hinzu, wie unten gezeigt. Die Datei **MyClass.java** sollte nun folgendermaßen aussehen:

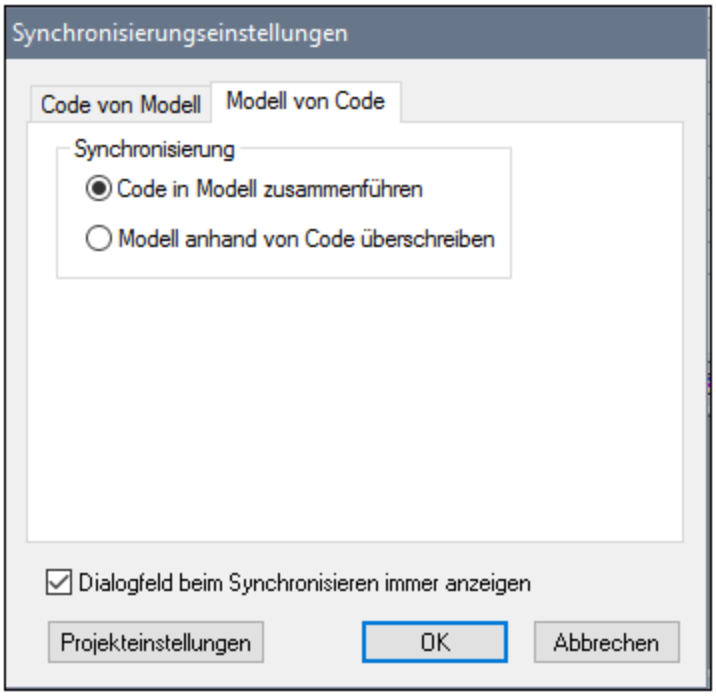
```
package com.nanonull;
public class MyClass{
   public float sum(float num1, float num2){
    return num1 + num2;
  }
}
```

MyClass.java

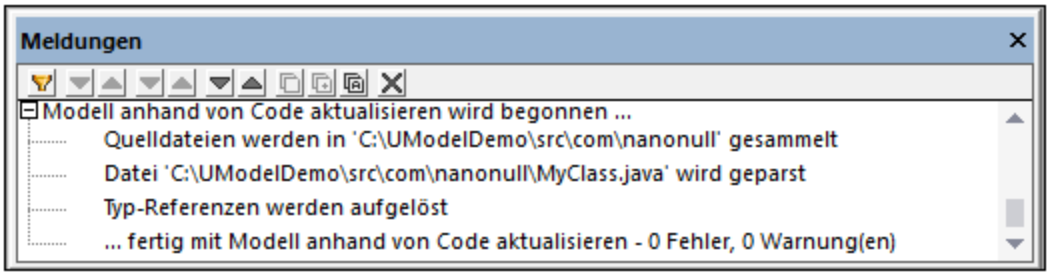
Zusammenführen der Codeänderungen im Modell

Sie können den geänderten Code nun wieder im Modell zusammenführen. Gehen Sie folgendermaßen vor:

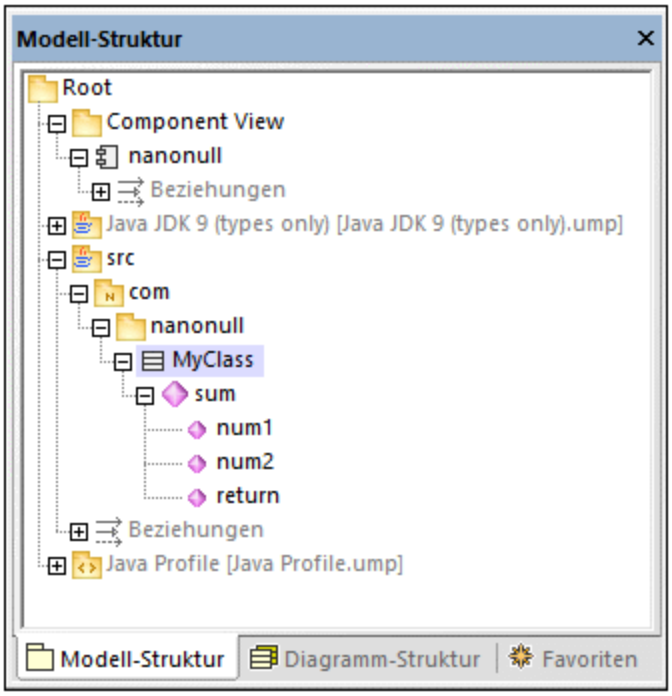
1. Klicken Sie im Menü **Projekt** auf **Merge UModel-Projekt aus Programmcode** (Drücken Sie alternativ dazu **Ctrl + F12**).



2. Lassen Sie die Standardsynchronisierungseinstellungen unverändert und klicken Sie auf **OK**. Es wird automatisch eine Codesyntaxüberprüfung durchgeführt, deren Ergebnis im Fenster **Meldungen** angezeigt wird:



Die Operation `sum` (die mit Reverse Engineering anhand des Codes generiert wurde) wird nun im Fenster "Modell-Struktur" angezeigt.



 © 2016-2022 Altova GmbH

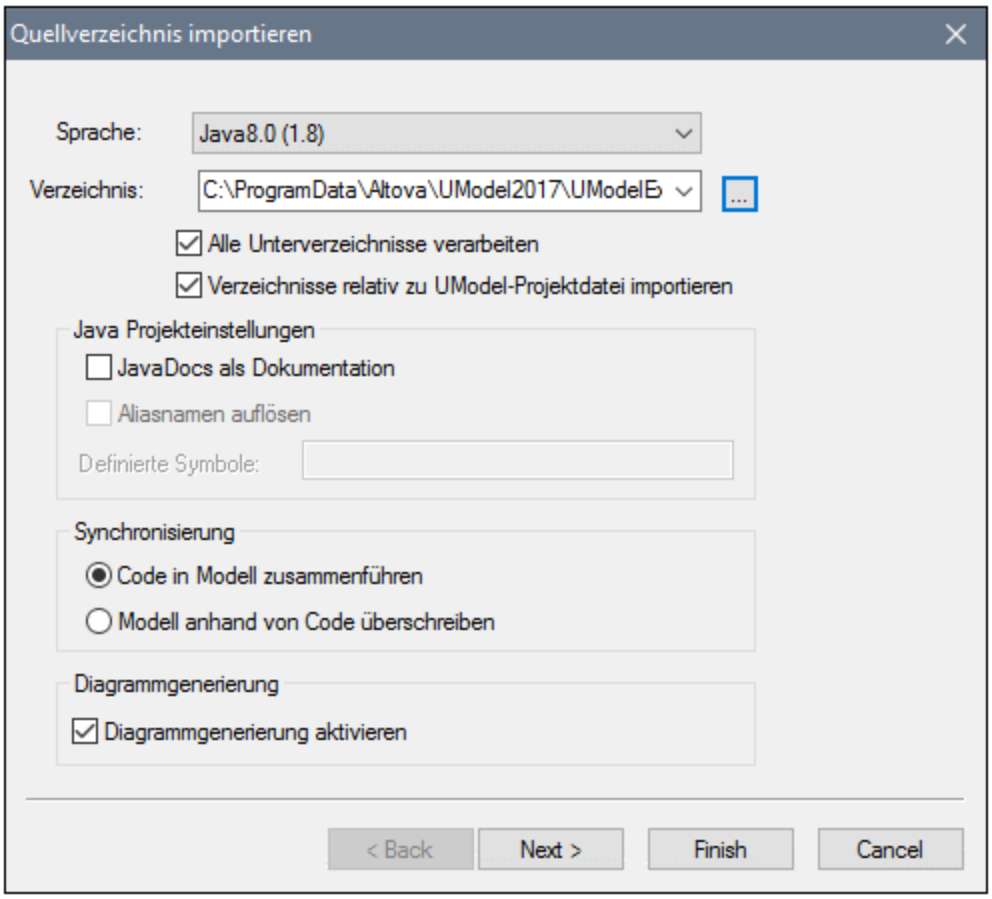
2.8 Reverse Engineering (Code zu Modell)

In diesem Tutorialabschnitt wird gezeigt, wie Sie vorhandenen Programmcode aus einem Verzeichnis in ein neues UModel-Projekt importieren (Reverse Engineering). Außerdem werden Sie im Modell eine neue Klasse hinzufügen, es für die Codegenerierung vorbereiten und die Änderungen anschließend mittels Forward Engineering im Java-Code wieder zusammenführen. Zwar wird in diesem Tutorial der Import von Java-Code beschrieben, doch ist das Verfahren für den Import von C# oder VB.NET-Code ähnlich.

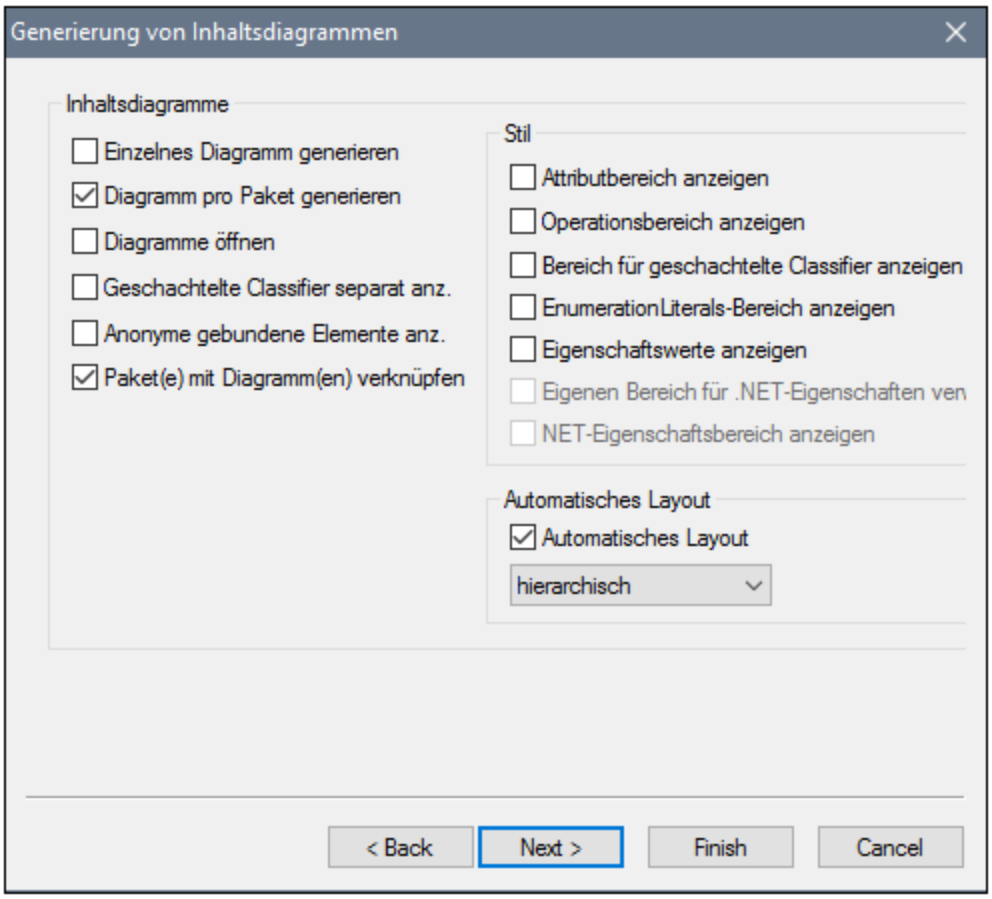
Anmerkung: Der in diesem Tutorial verwendete Java-Beispielcode steht in einem ZIP-Archiv unter dem folgenden Pfad zur Verfügung: **C:\Benutzer\<Benutzername>\Dokumente\Altova\UModel2023\UModelExamples\OrgChart.zip**. Entpacken Sie das Archiv bitte in dasselbe Verzeichnis, bevor Sie mit dem Tutorial beginnen.

Importieren von bestehendem Code aus einem Verzeichnis

1. Klicken Sie im Menü **Datei** auf **Neu**.
2. Klicken Sie im Menü **Projekt** auf **Quellverzeichnis importieren**.
3. Wählen Sie die Quellcodesprache aus (in diesem Beispiel Java).
4. Klicken Sie auf die Durchsuchen-Schaltfläche ... und wählen Sie das zuvor entpackte Verzeichnis **OrgChart** und klicken Sie auf **Weiter**. Beachten Sie, dass das Kontrollkästchen Diagrammgenerierung aktivieren aktiviert ist, damit UModel Klassen 394 - und Paketdiagramme 414 anhand des Quellcodes generiert.

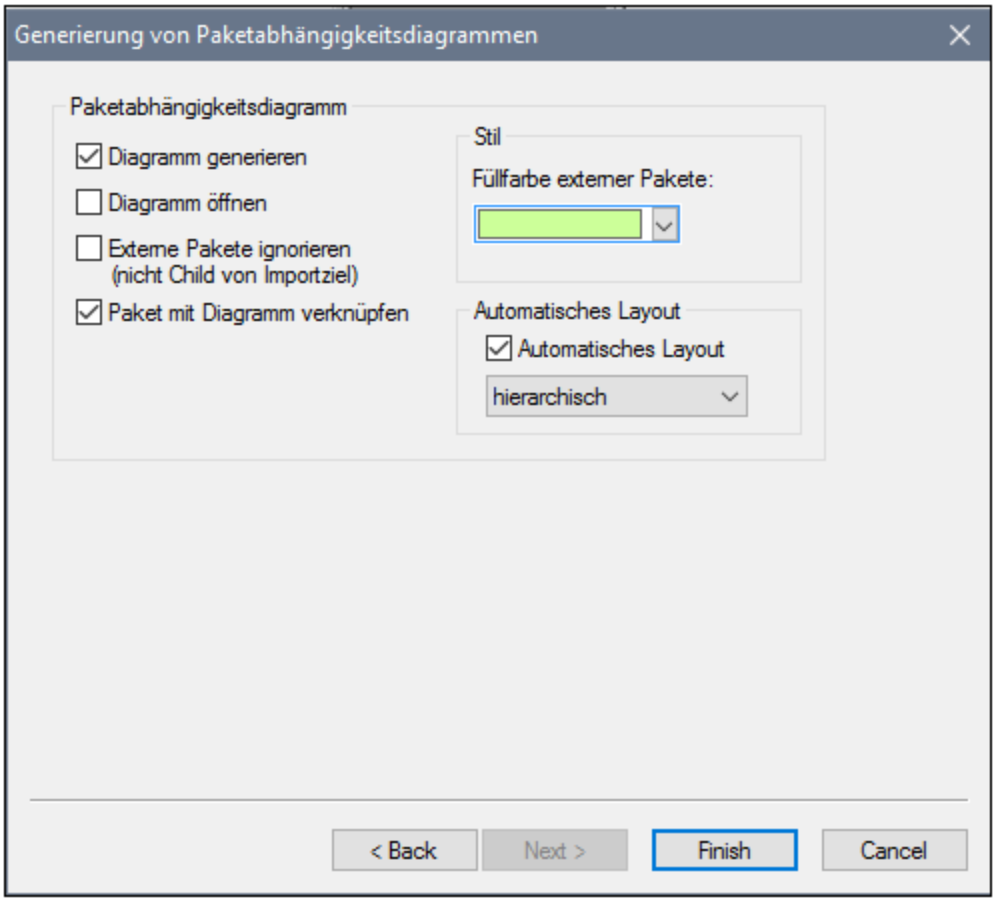


5. Wählen Sie die Option **Diagramm pro Paket generieren,** damit UModel für jedes Paket ein neues Diagramm generiert. Sie können die Diagrammstile später gegebenenfalls ändern.

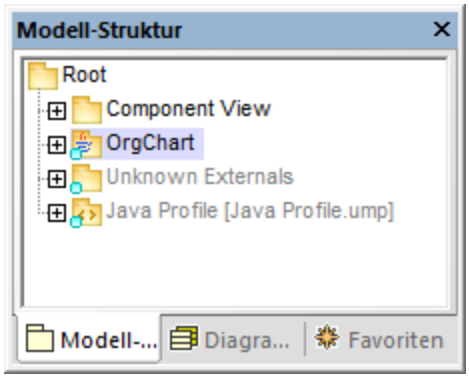


6. Klicken Sie zum Fortfahren auf **Weiter**. In diesem Dialogfeld können Sie die Einstellungen für die Generierung von Paketabhängigkeiten definieren.

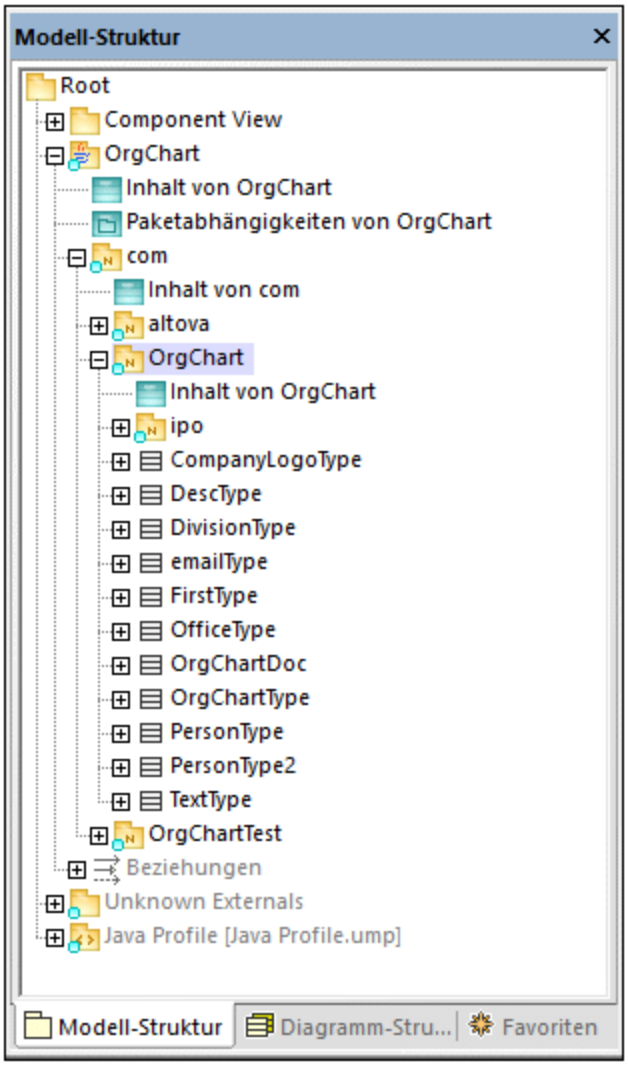
© 2016-2022 Altova GmbH



7. Klicken Sie auf "**Fertig stellen**". Wenn Sie dazu aufgefordert werden, speichern Sie das neue Modell in einem Verzeichnis auf Ihrem Rechner. Die Daten werden während der Eingabe geparst und es wird ein neues Paket namens "**OrgChart**" erstellt.

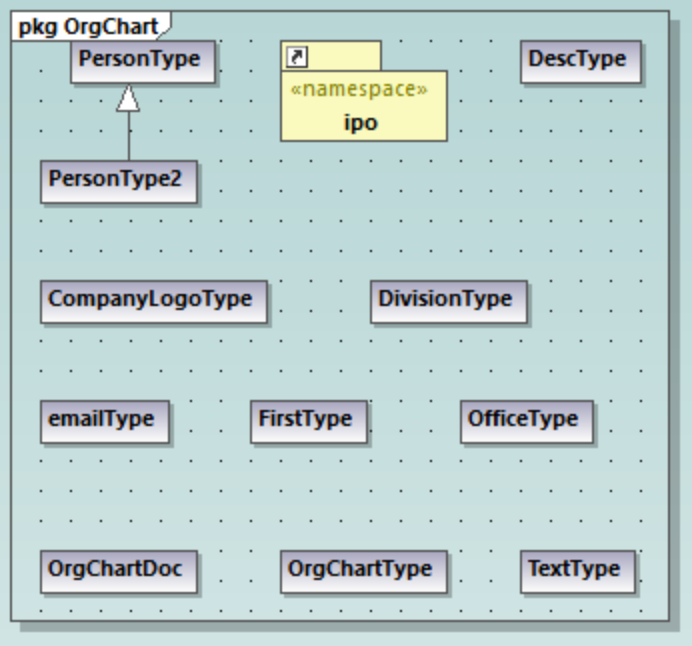


8. Erweitern Sie das neue Paket und dessen Unterpakete bis Sie zum Paket **OrgChart** (**com | OrgChart**) gelangen. Doppelklicken Sie auf das Diagrammsymbol "**Inhalt von OrgChart**":



Das Diagramm "Inhalt von OrgChart" wird nun im Hauptfenster angezeigt.

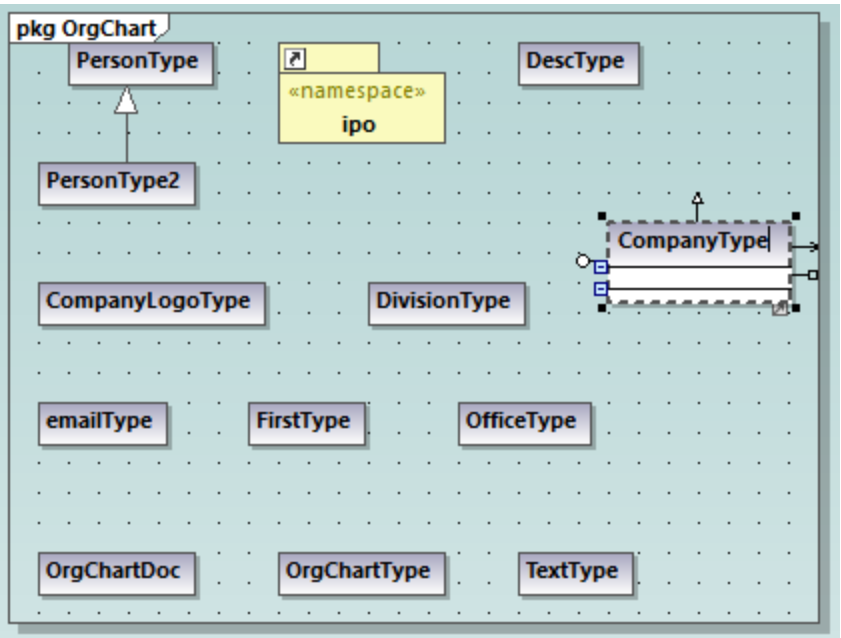
© 2016-2022 Altova GmbH



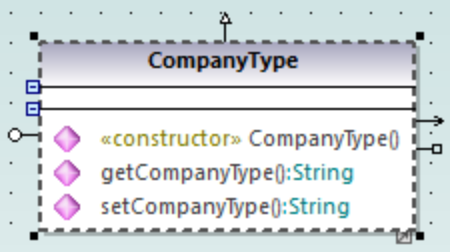
Hinzufügen einer neuen Klasse zum OrgChart-Diagramm:

Sie haben in dieser Phase des Tutorials aus bereits vorhandenem Java-Code mit Reverse Engineering ein Modell erstellt, das auch eine Reihe von automatisch generierten Diagrammen enthält. Wir werden nun einen Schritt weitergehen und das Modell um eine neue Klasse erweitern.

1. Klicken Sie mit der rechten Maustaste in das Diagramm "Inhalt von OrgChart" und wählen Sie im Kontextmenü den Befehl **Neu | Klasse**.
2. Klicken Sie auf die Überschrift der neuen Klasse und geben Sie als Name der neuen Klasse **CompanyType** ein.



3. Fügen Sie mittels der Taste **F8** in diesem Beispiel die folgenden Operationen zur Klasse hinzu: `CompanyType()`, `getCompanyType():String`, `setCompanyType():String`.

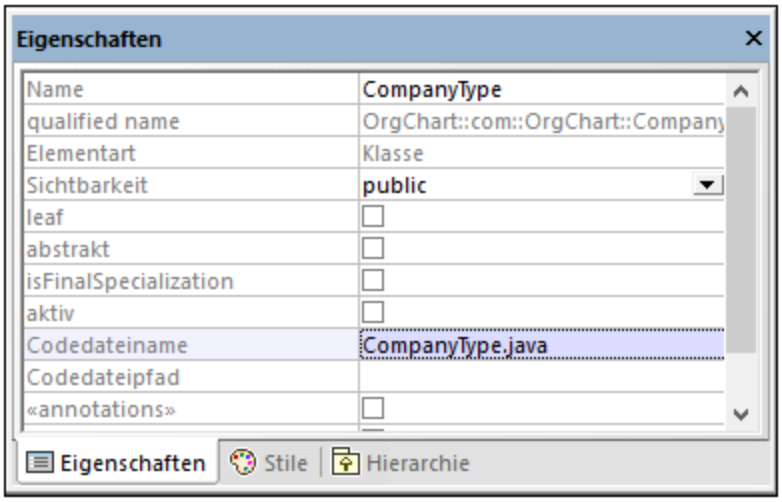


Anmerkung: Da der Klassenname `CompanyType` ist, wird der Operation `CompanyType()` automatisch das Stereotyp `<<constructor>>` zugewiesen.

Verfügbarmachen der neuen Klasse für die Codegenerierung:

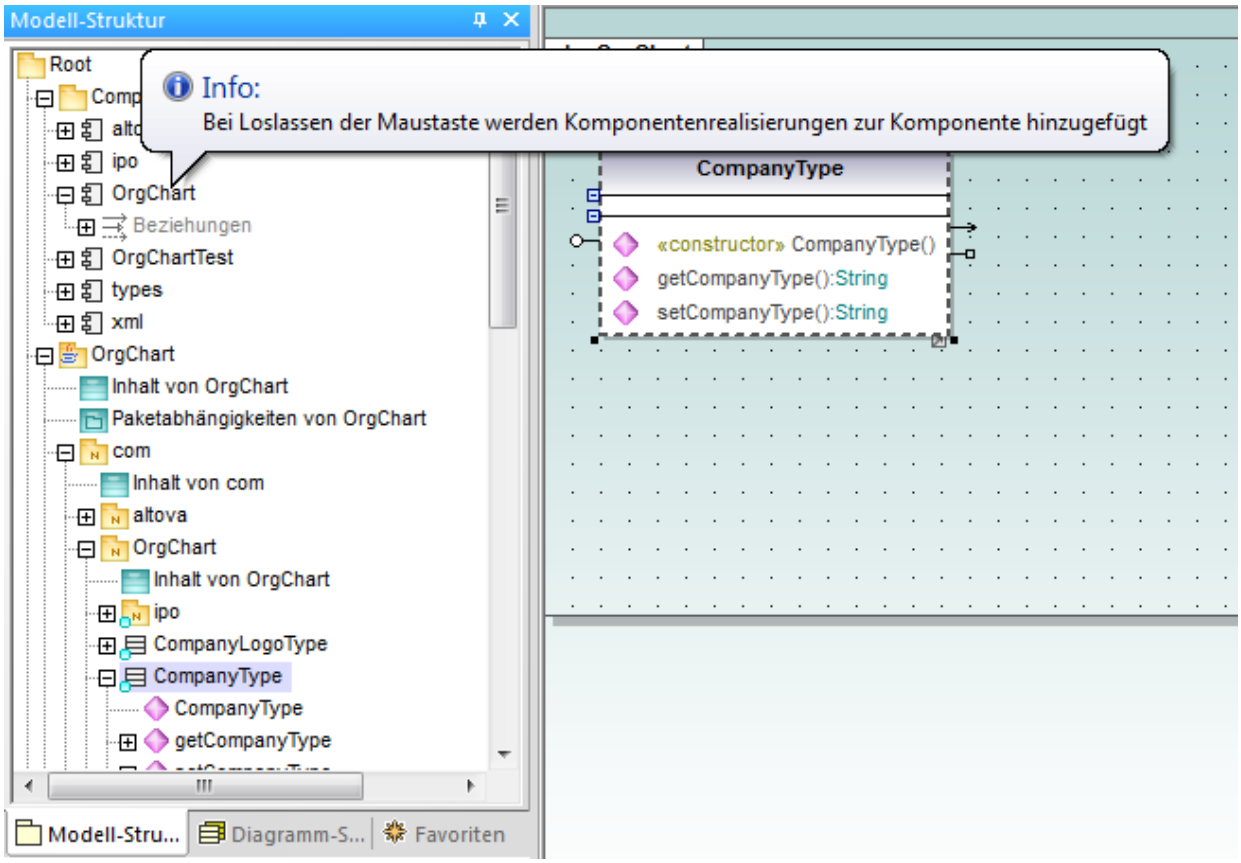
Da das Modell nun durch eine neue Klasse erweitert wurde, möchten Sie den zugrunde liegenden Code wahrscheinlich entsprechend aktualisieren, um Modell und Code zu synchronisieren. Wenn Sie jedoch jetzt **F11** drücken, um die Projektsyntax in dieser Phase des Projekts zu überprüfen, wird im Fenster "Meldungen" eine Warnung angezeigt: *'CompanyType' hat keine Komponentenrealisierung zu einer Komponente - die Komponentenrealisierung zur Komponente 'OrgChart' wird generiert*. Der Grund hierfür ist, dass für die neue Klasse eine Realisierung zu einer Komponente benötigt wird, bevor anhand dieser Klasse Code generiert werden kann (siehe Erklärung in Forward Engineering (Modell zu Code) 59. In einigen Fälle (wie in diesem Beispiel) kann UModel die erforderliche Realisierung automatisch generieren. Sie können die Realisierungsabhängigkeit allerdings auch folgendermaßen manuell definieren:

1. Während die Klasse "CompanyType" aktiv ist, klicken Sie im Fenster "Eigenschaften" in das Feld "Codedateiname" und geben Sie als Dateinamen "CompanyType.java" ein.

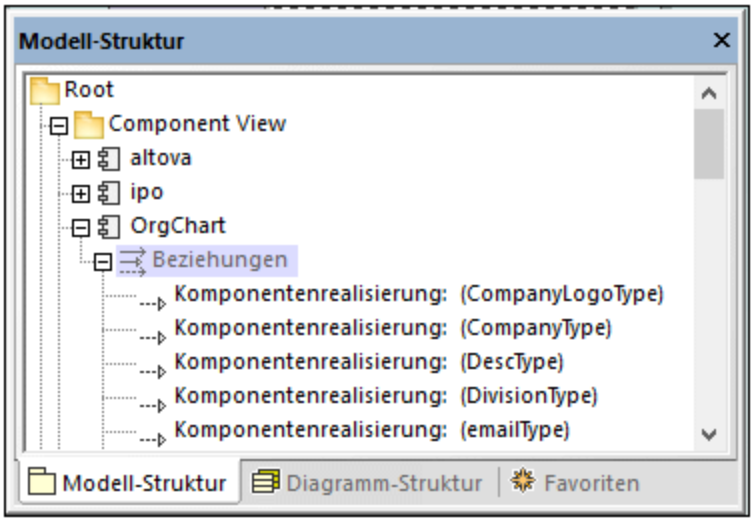


2. Klicken Sie in der Modellstruktur auf die neue Klasse `CompanyType` und ziehen Sie sie nach oben auf die **OrgChart**-Komponente unterhalb des Component View-Pakets. Es erscheint eine Info, wenn der Mauszeiger sich über einer Komponente befindet.

 © 2016-2022 Altova GmbH



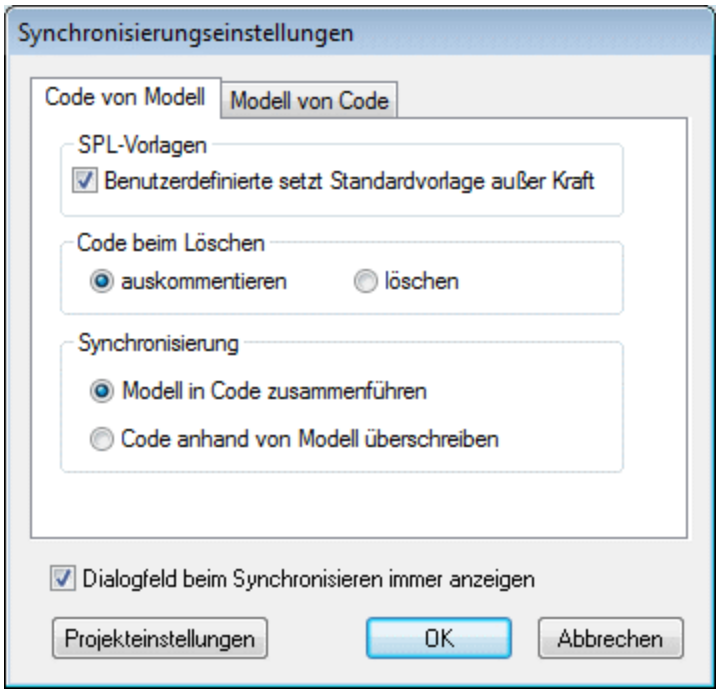
Mit dieser Methode wird eine Beziehung vom Typ "Komponentenrealisierung" zwischen einer Klasse und einer Komponente erstellt. Eine alternative Methode wäre, die Beziehung in einem Komponentendiagramm zu ziehen, siehe Komponentendiagramme [48]. Erweitern Sie das Element **Beziehungen** unterhalb der Komponente **Orgchart** um die neu erstellte Beziehung zu sehen.



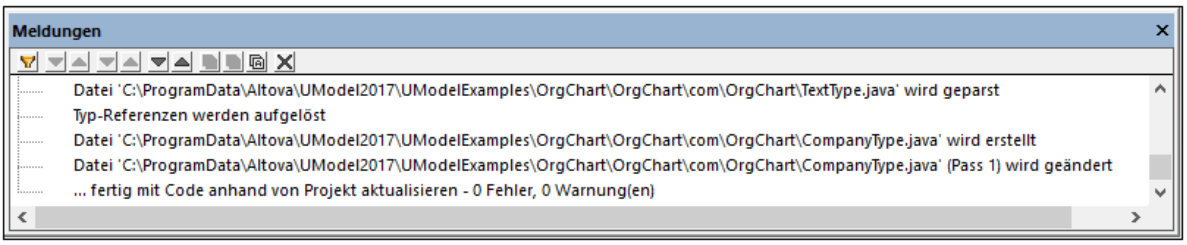
Merge von Programmcode aus einem Paket

Sie können in UModel Code auf Paket-, Komponentenebene oder für das gesamte Projekt generieren, siehe auch Synchronisieren von Modell und Quellcode 221. In diesem Beispiel generieren wir Code auf Komponentenebene. Gehen Sie dazu folgendermaßen vor:

1. Gehen Sie im Fenster "Modell-Struktur" unter "Component View" zur Komponente OrgChart.
2. Rechtsklicken Sie auf die Komponente "OrgChart**",** wählen Sie im Kontextmenü **Code Engineering | Merge Programmcode von UModel-Komponente**.



Im Meldungsfenster wird die Syntaxüberprüfung angezeigt, die durchgeführt wird, und der Status des Synchronisierungsvorgangs.



Nach Abschluss des Vorgangs ist die neue Klasse **CompanyType.java** zum Ordner **... \OrgChart\com\OrgChart** hinzugefügt.

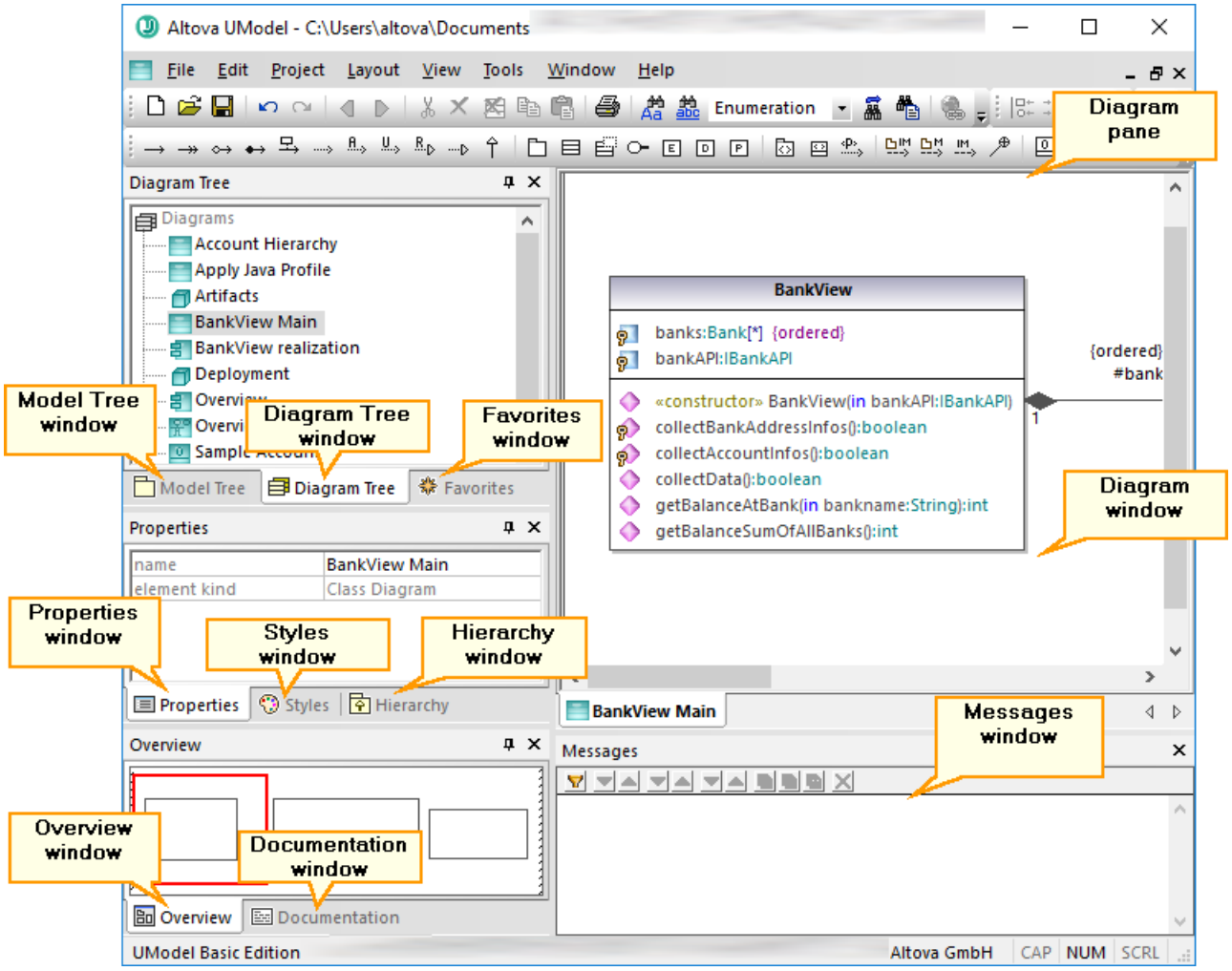
> Alle Methoden-Body-Bereiche und Änderungen am Code werden entweder auskommentiert oder gelöscht, je nachdem, welche Option im Dialogfeld "Synchronisierungseinstellungen" in der Gruppe "Code beim Löschen" ausgewählt wurde.

 © 2016-2022 Altova GmbH

Sie haben nun ein komplettes Round-Trip Engineering mit UModel durchgeführt.

3 Grafische Benutzeroberfläche von UModel

Die grafische Benutzeroberfläche von UModel besteht aus dem Diagrammhauptbereich und einer Reihe von kleineren Hilfsfenstern, über die Sie Daten eingeben oder anzeigen können. Der Diagrammbereich dient als Container für alle offenen Diagrammfenster. Um der Reihe nach alle offenen Diagrammfenster anzuzeigen, drücken Sie **Strg+Tab**.



Grafische Benutzeroberfläche von UModel

Standardmäßig sind die Hilfsfenster auf der linken Seite in Dreiergruppen angedockt und das Fenster "Meldungen" wird unterhalb des Diagrammbereichs angezeigt. Sie können alle Fenster jedoch nach Bedarf verschieben und andocken oder freischwebend anzeigen. Alle Fenster können über die Auswahlliste **Suchen** in der Hauptsymbolleiste oder durch Drücken von **Strg+F** durchsucht werden. Siehe auch Suchen und Ersetzen von Text 111.

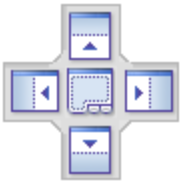
 © 2016-2022 Altova GmbH

So docken Sie ein Fenster an oder ab:

- Klicken Sie mit der rechten Maustaste auf seine Titelleiste und wählen Sie im Kontextmenü den Befehl **Angedockt** (bzw. **Abgedockt**).

So verschieben Sie ein Fenster:

1. Klicken Sie auf die Titelleiste des Fensters und ziehen Sie es an eine neue Position. Daraufhin wird eine Reihe von Andockhilfen angezeigt.



2. Ziehen Sie das Fenster über den oberen, rechten, linken oder unteren Ziehpunkt, um das Fenster an der neuen Position anzudocken.

So setzen Sie alle Symbolleisten und Fenster wieder in den Originalzustand zurück:

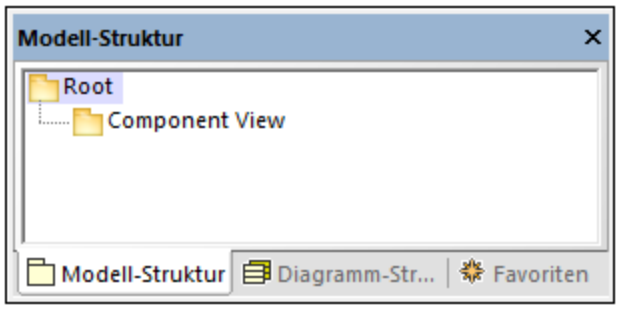
- Klicken Sie im Menü **Extras** auf **Symbolleisten und Fenster wiederherstellen**.

Dieses Kapitel bietet Referenzinformationen zu den Teilen, aus denen die grafische Benutzeroberfläche von UModel, besteht:

- Fenster "Modell-Struktur" 80
- Fenster "Diagramm-Struktur" 84
- Fenster "Favoriten" 85
- Fenster "Eigenschaften" 86
- Fenster "Stile" 87
- Fenster "Hierarchie" 88
- Fenster "Übersicht" 90
- Fenster "Dokumentation" 91
- Fenster "Meldungen" 92
- Diagrammfenster 94
- Diagrammbereich 95

3.1 Fenster "Modell-Struktur"

Im Fenster "Modell-Struktur" können Sie alle Einträge im UModel-Projekt (Pakete, Klassen, Diagramme, Beziehungen, usw.) anzeigen und bearbeiten.



Fenster "Modell-Struktur"

Wenn Sie ein neues UModel-Projekt erstellen, stehen standardmäßig zwei Pakete zur Verfügung: "Root" und "Component View". Diese beiden Pakete sind die einzigen, die nicht umbenannt oder gelöscht werden können. Das "Root"-Paket dient als Ausgangspunkt für die Modellierung aller anderen Elemente, während das "Component View"-Paket für das Code Engineering benötigt wird.

Sie können entweder über dieses Fenster oder direkt über ein Diagramm zusätzliche Pakete, Klassen, Diagramme und deren Hierarchie erstellen, siehe Erstellen von Elementen 106. Informationen zu weiteren Operationen, die Sie an Modellelementen in der Modell-Struktur ausführen können, finden Sie im Kapitel Anleitung zur Modellierung von... 104.

Anmerkung: UModel enthält eine Reihe von Beispielprojekten, anhand welcher Sie die Grundlagen der Modellierung erlernen und die grafische Benutzeroberfläche von UModel kennenlernen können. Diese Projekte befinden sich im folgenden Ordner: **C:\Benutzer\<Benutzername>\Dokumente\Altova\UModel2023\UModelExamples**.

Anzeigen, Ausblenden und Sortieren von Modellelementen in der Modell-Struktur

Um zu konfigurieren, was im Fenster "Modell-Struktur" angezeigt werden soll und um die Sortieroptionen zu definieren, klicken Sie mit der rechten Maustaste in das Fenster und wählen Sie die gewünschte Menüoption aus. Um alle Aktionen, die an im Fenster "Modell-Struktur" angezeigten Modellelementen, ausgeführt werden können, anzuzeigen, klicken Sie mit der rechten Maustaste auf das Modellelement und sehen Sie sich die Kontextmenüoptionen an.

Erweitern und Reduzieren von Modellelementen in der Modell-Struktur

So erweitern Sie Modellelemente (z.B. Pakete) im Fenster "Modell-Struktur":

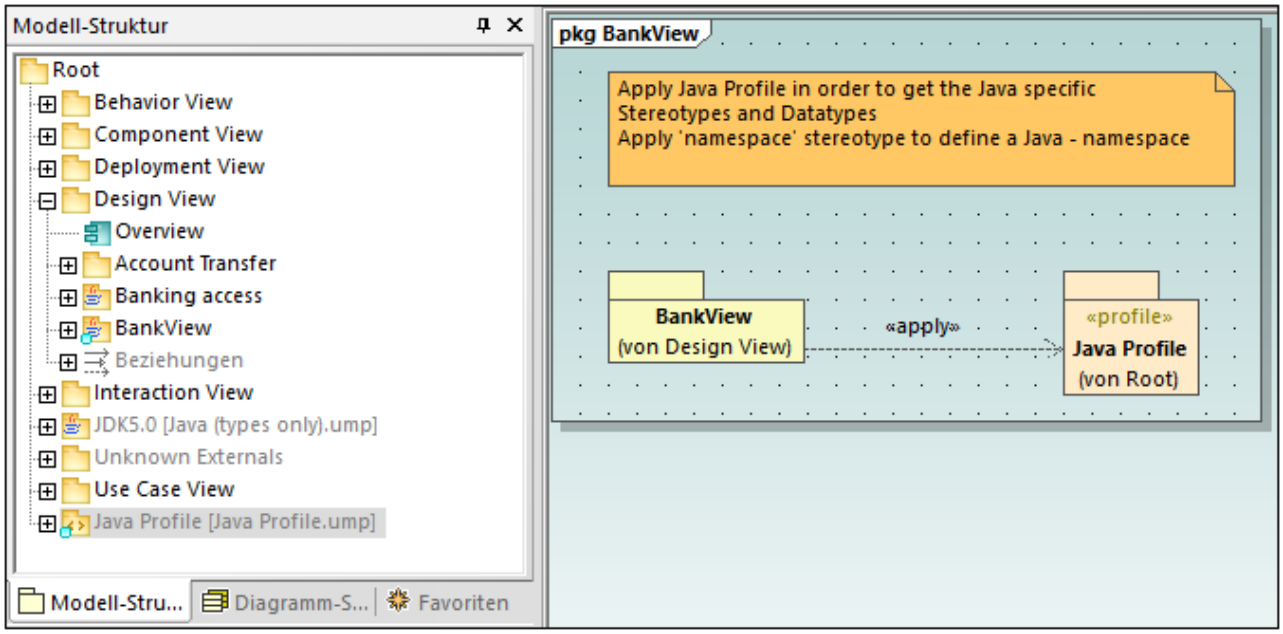
- Drücken Sie die *-Taste (Sternchen), um das aktuelle Modellelement sowie alle untergeordneten Elemente zu erweitern.
- Drücken Sie die +-Taste (Plus), um nur das aktuelle Modellelement zu erweitern.

 © 2016-2022 Altova GmbH

Um die Pakete reduziert anzuzeigen, drücken Sie die -Taste (Bindestrich). Um alle Modellelemente zu reduzieren, klicken Sie auf das "Root"-Paket und drücken Sie - (Bindestrich). Beachten Sie, dass Sie dazu sowohl die Tasten der Standardtastatur als auch die Tasten des Zahlenblocks verwenden können.

Identifizieren von aktiven Diagrammeinträgen

Wenn ein Diagramm im Diagrammbereich geöffnet ist, werden einige Einträge im Fenster "Modell-Struktur" mit einem hellblauen Punkt am unteren Rand des Eintrags angezeigt. Dies sind die Modellelemente, die im gerade aktiven Diagramm angezeigt werden (wie z.B. "BankView" und "Java Profile" im Beispiel unten):



Symbolreferenz

Im Fenster "Modell-Struktur" können zahlreiche Symbole für Elemente und Diagramme in Ihrem Projekt, in den Code Engineering-Paketen sowie den importierten Profilen oder Unterprojekten angezeigt werden. Insbesondere können die folgenden Pakettypen angezeigt werden:

Symbol	Beschreibung
	UML-Standardpaket
	Java Namespace Root-Paket. Wird für die Generierung oder das Reverse Engineering von Java-Code verwendet.
	C# Namespace Root-Paket. Wird für die Generierung oder das Reverse Engineering von C#-Code verwendet.

Symbol	Beschreibung
	Visual Basic Namespace Root-Paket. Wird für die Generierung oder das Reverse Engineering von VB.NET-Code verwendet.
	XML Schema Namespace Root-Paket. Wird für die Generierung von XML-Schemas anhand des Modells oder den Import dieser Schemas in das Modell verwendet, siehe XML-Schema-Diagramme 435.
	Ein Namespace-Paket (ein Paket auf den das `<<namespace>>`-Stereotyp angewendet wurde).
	Ein-UML-Profil

In der folgenden Tabelle sind die Diagramme, die im Fenster "Modell-Struktur" angezeigt werden können, aufgelistet.

Symbol	Beschreibung
	Aktivitätsdiagramm
	Klassendiagramm
	Kommunikationsdiagramm
	Komponentendiagramm
	Kompositionsstrukturdiagramm
	Deployment-Diagramm
	Interaktionsübersichtsdiagramm
	Objektdiagramm
	Paketdiagramm
	Profildiagramm
	Protokoll-Zustandsdiagramm
	Sequenzdiagramm
	Zustandsdiagramm
	Zeitverlaufsdiagramm
	Use Case-Diagramm
	XML-Schema-Diagramm

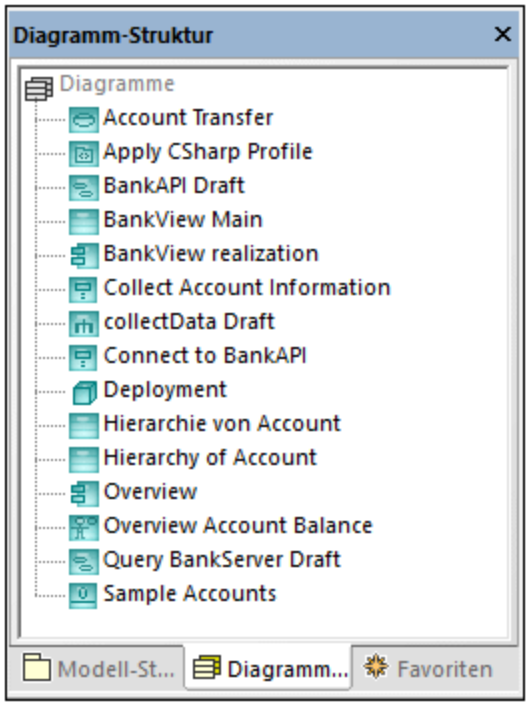
Unten finden Sie einige Beispiele für UML-Modellierungselemente, die im Fenster "Modell-Struktur" angezeigt werden können. Nähere Informationen zu UML-Elementen und den Diagrammtypen, in denen diese vorkommen, finden Sie im Kapitel UML-Diagramme 301.

© 2016-2022 Altova GmbH

Symbol	Beschreibung
	Klasse
	Eigenschaft
	Operation
	Parameter
	Akteur
	Use Case
	Komponente
	Knoten
	Artefakt
	Schnittstelle
	Klasseninstanz (Objekt)
	Klassen-Instanz-Slot
	Beziehungen
	Einschränkungen

3.2 Fenster "Diagramm-Struktur"

Im Fenster "Diagramm-Struktur" werden alle im UModel-Projekt enthaltenen Diagramme angezeigt.



Fenster "Diagramm-Struktur"

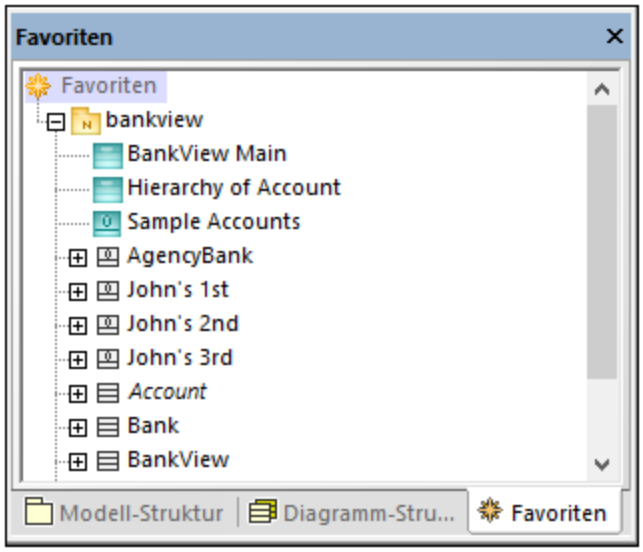
Diagramme in diesem Fenster können entweder als alphabetische Liste oder nach Typ gruppiert angezeigt werden. Um die Anzeigeoption zu ändern, klicken Sie mit der rechten Maustaste in das Fenster und aktivieren bzw. deaktivieren Sie die Option **Nach Diagrammtyp gruppieren**.

Anleitungen zum Erstellen, Öffnen und Generieren von Diagrammen sowie zum Modellieren des Inhalts finden Sie im Kapitel Anleitung zur Modellierung von... 104. Nähere Informationen zu den einzelnen Diagrammtypen finden Sie im Kapitel UML-Diagramme 301.

 © 2016-2022 Altova GmbH

3.3 Fenster "Favoriten"

Im Fenster "Favoriten" werden alle Modellierungselemente oder Diagramme, die Sie als Favoriten hinzugefügt haben, angezeigt. Die "Favoriten" stellen eine persönliche, benutzerdefinierte Liste von Modellierungselementen oder Diagrammen dar, die Sie auf diese Art schnell aufrufen können.



Fenster "Favoriten"

Standardmäßig wird der Inhalt des Fensters "Favoriten" automatisch beim Speichern des Projekts gespeichert. Sie können diese Option über das Menü **Extras | Optionen** Register **Datei** ändern. Der Name der entsprechenden Option ist **Mit Projektdatei laden und speichern | Favoriten**.

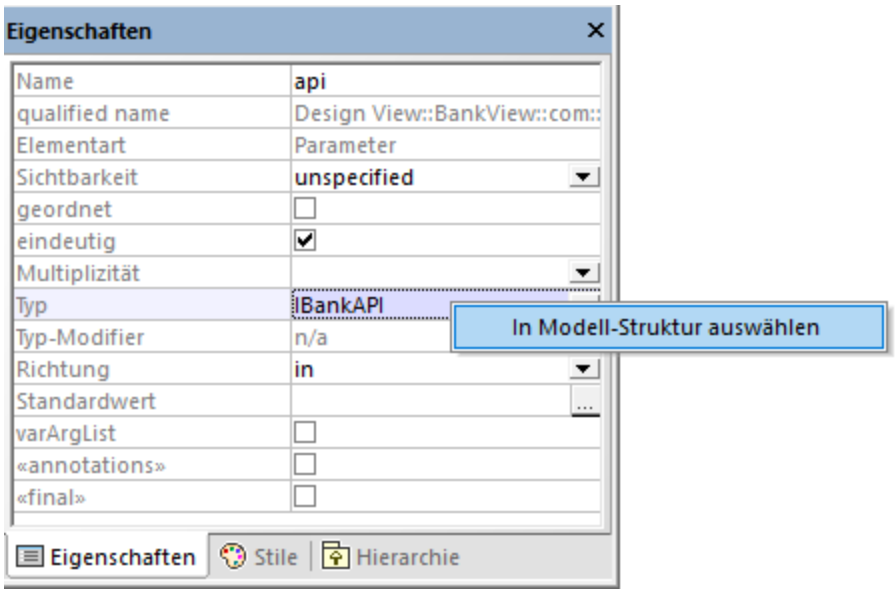
Einträge im Fenster "Favoriten" sind keine Kopien oder Klone, sondern stehen für die tatsächlichen Elemente bzw. Diagramme. Die meisten Aktionen, die Sie im Fenster "Modell-Struktur" anwenden können, lassen sich auch im Fenster "Favoriten" ausführen, darunter auch Hinzufügen und Löschen von Elementen. Nähere Informationen dazu finden Sie im Kapitel Anleitung zur Modellierung von... 104.

3.4 Fenster "Eigenschaften"

Im Fenster "Eigenschaften" werden Informationen über das gerade ausgewählte Modellelement (das sich im Fokus befindet) angezeigt. Bei dem "im Fokus" befindlichen Element kann es sich um ein im Fenster "Modell-Struktur" (oder in anderen Fenstern) ausgewähltes Element, ein im Diagramm ausgewähltes Element oder ein Diagramm selbst handeln.

Außerdem können Sie über das Fenster "Eigenschaften" die Eigenschaften des aktuell ausgewählten Elements bzw. der Beziehung ändern. Welche Eigenschaften zur Verfügung stehen, hängt von der Art des ausgewählten Elements ab. Es gibt Eigenschaften, die schreibgeschützt und somit ausgegraut (z.B. "Elementart") sind und solche, die Sie ändern können (z.B. "Name").

Wenn eine Operation oder Eigenschaft einen Parameter erhält, können Sie direkt vom Fenster "Eigenschaften" aus schnell zum entsprechenden Parametertyp im Fenster "Modell-Struktur" springen. Klicken Sie dazu im Fenster "Eigenschaften" mit der rechten Maustaste auf die Eigenschaft "Typ" des Parameters und wählen Sie im Kontextmenü den Befehl **In Modell-Struktur auswählen**. Dasselbe gilt auch für Rückgabeparameter.



Fenster "Eigenschaften"

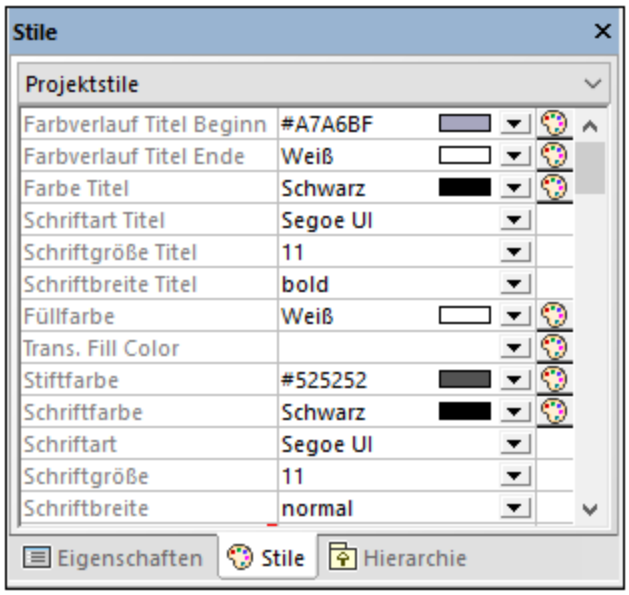
Wenn Sie die Eigenschaft eines Elements im Fenster "Eigenschaften" ändern, wird diese sofort im Diagramm angezeigt. Wenn Sie umgekehrt eine Änderung im Diagramm vornehmen (z.B. die Sichtbarkeit einer Operation von `public` in `private` ändern), wirkt sich dies auf die entsprechende Eigenschaft im Fenster "Eigenschaften" aus.

Innerhalb von doppelten spitzen Klammern stehende Eigenschaften, sind Stereotype (z.B. `«final»`). Sie können benutzerdefinierte Stereotype zum Projekt hinzufügen. Diese würden dann im Fenster "Eigenschaften" zusätzlich zu den Standardstereotypen im Fenster "Eigenschaften" angezeigt werden. Nähere Informationen dazu finden Sie unter Beispiel: Erstellen und Anwenden von Stereotypen 427.

3.5 Fenster "Stile"

Im Fenster "Stile" können Sie die visuelle Darstellung von gerade ausgewählten Diagrammen oder Elementen (im Fokus) anzeigen oder ändern. Die Stilattribute werden in zwei allgemeine Gruppen eingeteilt:

- Formatierungseinstellungen (z.B. Schriftgröße, Schriftbreite, Farbe, usw.)
- Anzeigeeinstellungen (z.B. Hintergrundfarbe anzeigen, Raster, Sichbarkeitseinstellungen, usw.).



Fenster "Stile"

Wenn Sie die Eigenschaft im Fenster "Stile" ändern, wird dies sofort auf der Benutzeroberfläche angezeigt. Wenn Sie umgekehrt an einer anderen Stelle eine Stiländerung vornehmen (z.B. die Sichtbarkeit des Diagrammrasters über die Symbolleisten-Schaltfläche **Raster anzeigen** ändern), wirkt sich dies auf die entsprechende Eigenschaft im Fenster "Stile" aus.

Das Fenster "Stile" hat im oberen Bereich eine Dropdown-Liste, über die Sie die Ebene, auf der die Stiländerung angewendet werden soll, auswählen können (z.B. auf ein einzelnes Element oder auf Projektebene). Nähere Informationen dazu finden Sie unter:

- Ändern des Stils von Elementen 119
- Ändern des Stils von Diagrammen 127
- Ändern des Stils von Linien und Beziehungen 134

3.6 Fenster "Hierarchie"

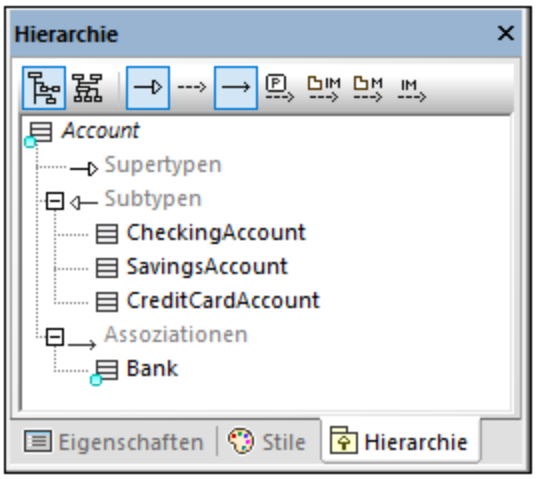
Im Fenster "Hierarchie" werden alle Beziehungen des aktuell ausgewählten Modellierungselements in zwei verschiedenen Ansichten angezeigt. Das Modellierungselement kann in einem Diagramm, im Fenster "Modell-Struktur" oder im Fenster "Favoriten" ausgewählt werden.

Einträge im Fenster "Hierarchie" können in zwei Ansichten angezeigt werden:

- Strukturansicht
- Graph-Ansicht

Um zwischen den Ansichten zu wechseln, klicken Sie in der linken oberen Ecke des Fensters auf die Schaltfläche **Strukturansicht anzeigen** bzw. **Graph. Ansicht anzeigen** .

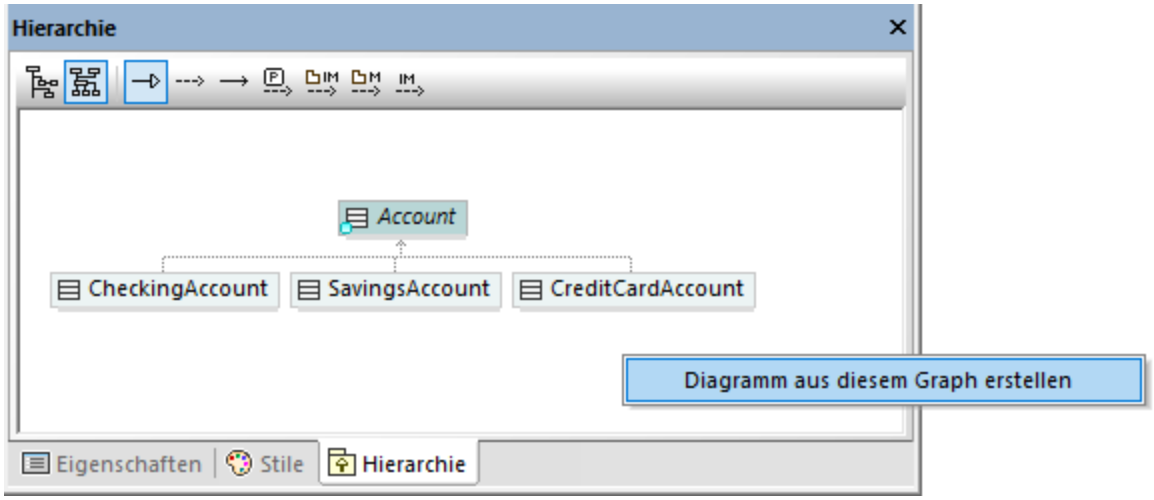
In der *Strukturansicht* werden mehrere Beziehungen des aktuell ausgewählten Elements in Form einer Struktur angezeigt. Klicken Sie auf die Schaltflächen am oberen Rand des Fensters, um die Arten der anzeigenden Beziehungen auszuwählen. In der Abbildung unten wurden nur Generalisierungen und Assoziationen für die Anzeige ausgewählt.



Fenster "Hierarchie" (Strukturansicht)

In der *Graph-Ansicht* sehen Sie eine einzige Gruppe von Beziehungen in einer hierarchischen Übersicht in Form eines Diagramms. In dieser Ansicht kann immer nur eine der Beziehungsschaltflächen aktiv sein. In der Abbildung unten ist gerade die Schaltfläche **Generalisierungen anzeigen** aktiv.

 © 2016-2022 Altova GmbH



Fenster "Hierarchie" (graphische Ansicht)

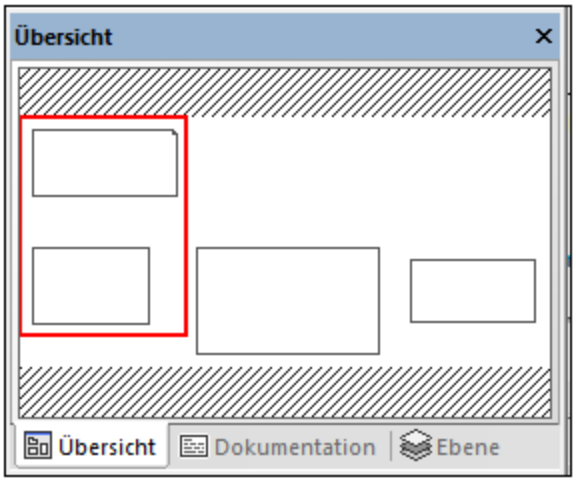
In der Graph-Ansicht können Sie Diagramme generieren, die die im Fenster sichtbaren Elemente enthalten. Klicken Sie dazu mit der rechten Maustaste in das Fenster und wählen Sie im Kontextmenü den Befehl **Diagramm aus diesem Graph erstellen**.

Die Einstellungen zum Fenster "Hierarchie" können über die Menüoption **Extras | Optionen | Ansicht** Gruppe **Hierarchie** im unteren Bereich des Dialogfelds geändert werden.

Um durch das Fenster "Hierarchie" zu navigieren, doppelklicken Sie auf eines der Elementsymbole im Fenster, um die Beziehungen dieses Elements anzuzeigen. Dies gilt sowohl für die Strukturansicht als auch für die Graph-Ansicht.

3.7 Fenster "Übersicht"

Im Fenster "Übersicht" werden die Umrisse des aktuell aktiven Diagramms in Form einer Übersicht angezeigt. Dieses Fenster ist besonders hilfreich, wenn Sie den Ansichtsbereich in großen Diagrammen verschieben möchten. Wenn Sie auf das rote Rechteck klicken und es ziehen, verschiebt sich der angezeigte Bereich im Diagramm.



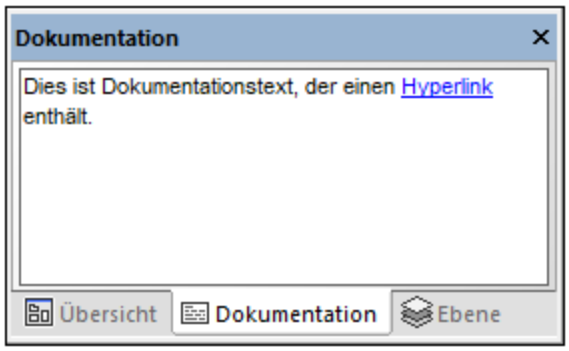
Fenster "Übersicht"

Siehe auch Vergrößern und Verkleinern von Diagrammen 132.

© 2016-2022 Altova GmbH

3.8 Fenster "Dokumentation"

Über das Fenster "Dokumentation" können alle der im Fenster "Modell-Struktur" verfügbaren UML-Elemente dokumentiert werden. Um Dokumentation hinzuzufügen, klicken Sie zuerst auf das gewünschte Element und geben Sie anschließend im Fenster "Dokumentation" Text ein. Dabei werden auch die Tastenkombinationen für die Standardbearbeitungsbefehle **Alle auswählen (Strg+A), Ausschneiden (Strg+X), Kopieren (Strg+C)** und **Einfügen (Strg+V)** unterstützt.



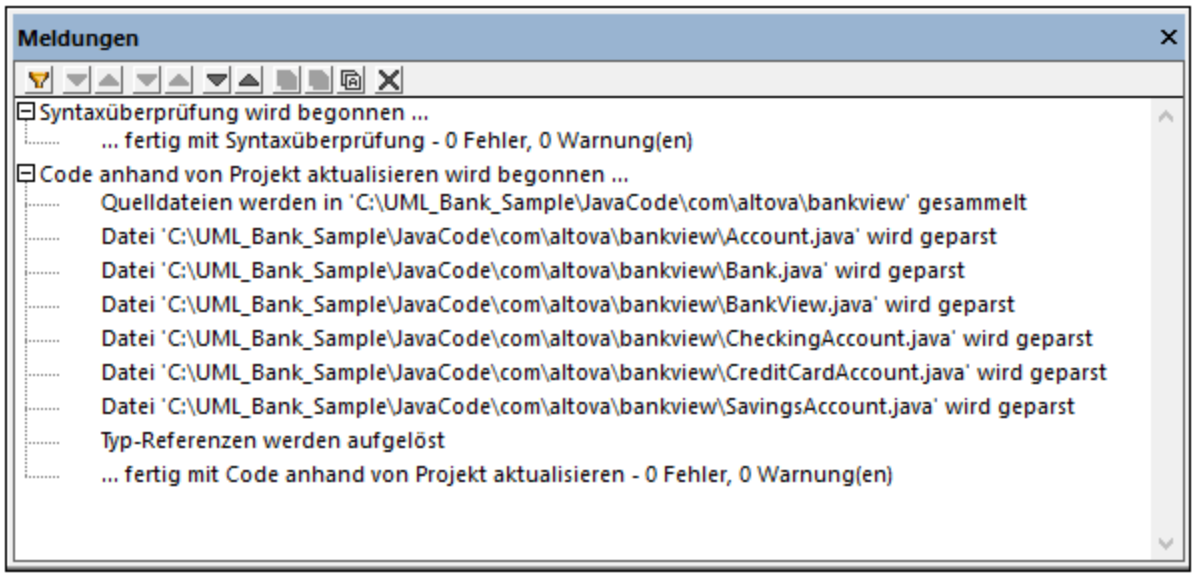
Fenster "Dokumentation"

Sie können die Rechtschreibung von Text im Dokumentationsfenster prüfen. Klicken Sie dazu mit der rechten Maustaste in das Fenster und wählen Sie im Kontextmenü den Befehl **Rechtschreibung der Dokumentation**.

Dokumentationstext kann auch in Form von Kommentaren in den generierten Quellcode exportiert oder beim Reverse Engineering aus dem Quellcode importiert werden. Nähere Informationen dazu finden Sie unter Dokumentieren von Elementen 118.

3.9 Fenster "Meldungen"

Im Fenster "Meldungen" werden die folgenden Arten von Meldungen angezeigt: Informationsmeldungen, Warnungen und Fehler. Solche Meldungen können vorkommen, wenn Sie die Projektsyntax überprüfen (siehe Überprüfen der Projektsyntax 173) oder wenn Sie Code Engineering-Aufgaben durchführen. Nähere Informationen zum Code Engineering finden Sie unter Generieren von Programmcode 170 und Importieren von Quellcode 194.



Fenster "Meldungen"

In der Tabelle unten sind mögliche Fehlerarten und deren Symbole aufgelistet.

Symbol	Beschreibung
Keines	Kennzeichnet eine Informationsmeldung.
	Kennzeichnet eine Warnmeldung. Warnungen sind weniger gravierend als Fehler, es kann aber auch bei einer Warnung vorkommen, dass Code nicht importiert oder generiert werden kann.
	Kennzeichnet eine Fehlermeldung. Bei einem Fehler schlägt die Codegenerierung oder der Codeimport fehl.

Über die Schaltflächen am oberen Rand des Fensters "Meldungen" können Sie die folgenden Aktionen ausführen:

Symbol	Beschreibung
	Filtert Meldungen nach ihrem Schweregrad (Informationsmeldungen und Warnungen). Wählen Sie **Alle aktivieren,** um alle Schweregrade zu inkludieren (dies ist die Standardeinstellung). Wählen Sie **Alle deaktivieren,** um alle Schweregrade aus dem Filter

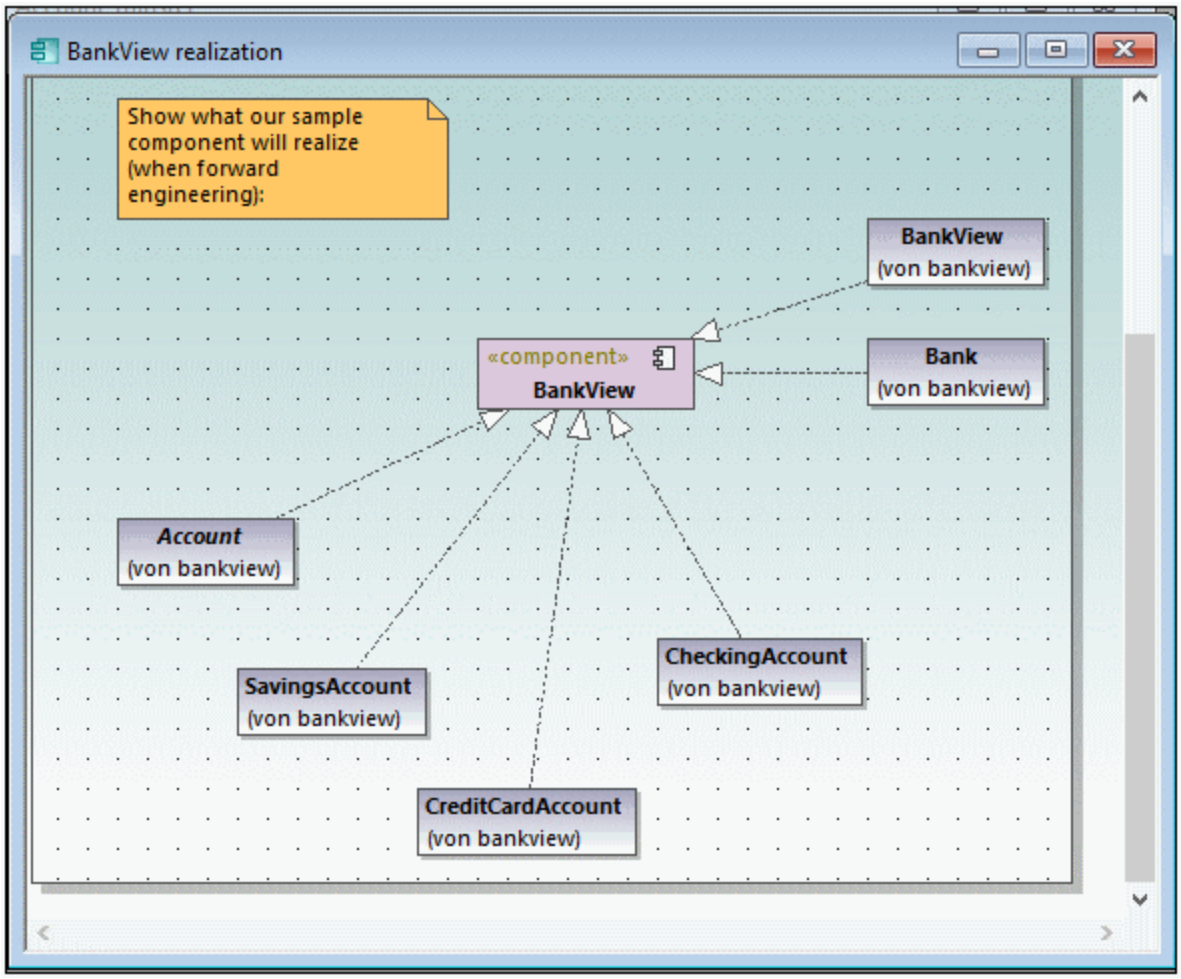
 © 2016-2022 Altova GmbH

Symbol	Beschreibung
	zu entfernen.
	Geht zum nächsten Fehler.
	Geht zum vorherigen Fehler.
	Geht zur nächsten Warnung.
	Geht zur vorherigen Warnung.
	Geht zur nächsten Zeile.
	Geht zur vorherigen Zeile.
	Kopiert die ausgewählte Zeile in die Zwischenablage.
	Kopiert die ausgewählte Zeile einschließlich aller untergeordneten Zeilen in die Zwischenablage.
	Kopiert den gesamten Inhalt des Fensters **Meldungen** in die Zwischenablage.
	Löscht die Meldungen im Fenster "Meldungen".

3.10 Diagrammfenster

Wenn Sie ein neues Diagramm erstellen oder ein vorhandenes öffnen, wird im Diagrammbereich 95 ein neues Diagrammfenster geladen. Das Diagrammfenster stellt den Zeichenbereich, in dem UML-Diagramme erstellt werden, dar. Wenn Sie mit der rechten Maustaste entweder in den Zeichenbereich selbst oder auf ein beliebiges Element darin klicken, stehen verschiedene Modellierungsbefehle zur Verfügung.

Je nachdem, welcher Diagrammtyp gerade aktiv (im Fokus) ist, stehen jeweils andere Symbolleisten-Schaltflächen und Kontextmenübefehle zur Verfügung. Wenn Sie z.B. in ein Klassendiagramm klicken, stehen nur Symbolleisten-Schaltflächen für Klassendiagrammelemente zur Verfügung. Um den Diagrammtyp anzuzeigen, klicken Sie in einen leeren Bereich im Diagramm und sehen Sie nach, was im Fenster "Eigenschaften" 86 als Elementart angezeigt wird. Man erkennt den Diagrammtyp auch am Symbol für das Diagramm, siehe Erstellen von Diagrammen 122.



Diagrammfenster

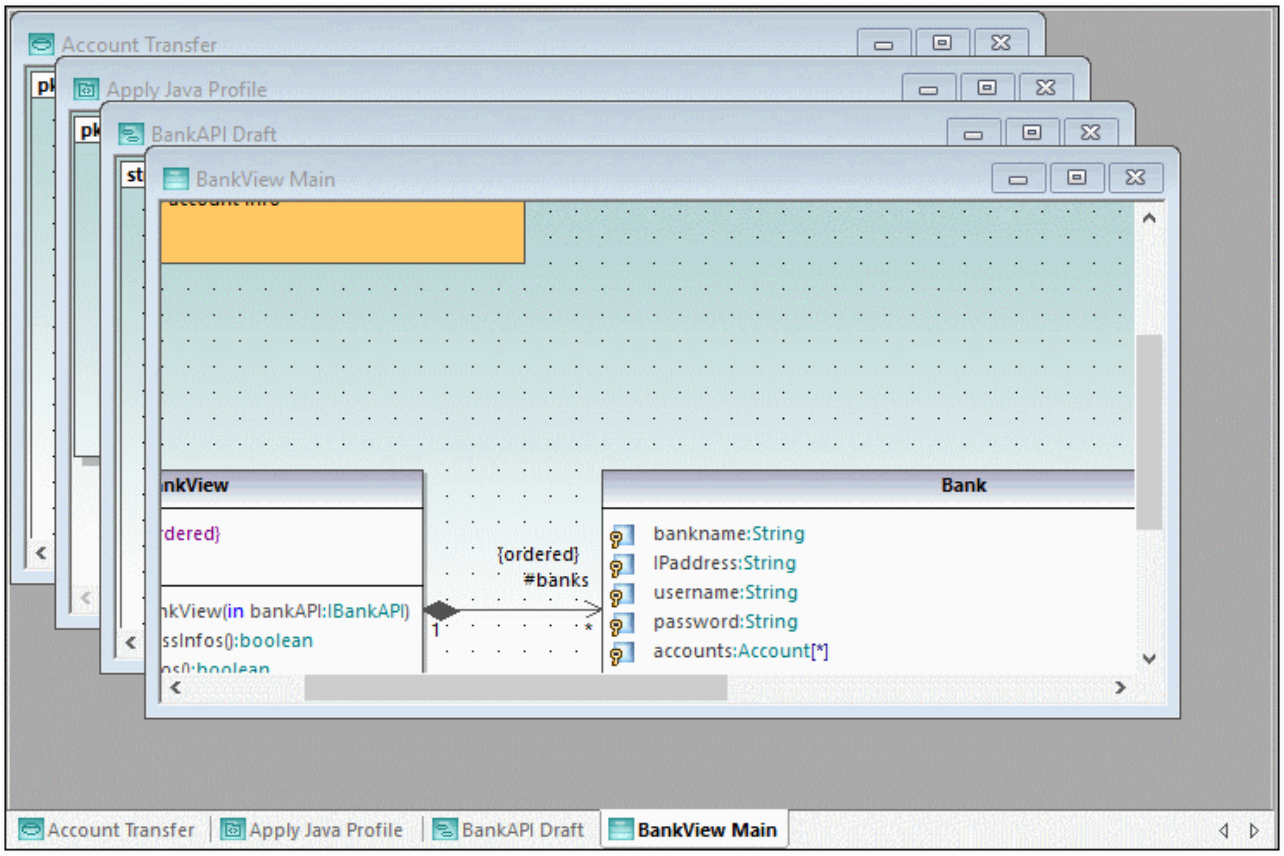
Informationen zum Erstellen von neuen Diagrammen, zum Öffnen von bestehenden und zum Bearbeiten von Elementen im Diagramm finden Sie im Kapitel Anleitung zur Modellierung von... 104.

 © 2016-2022 Altova GmbH

3.11 Diagrammbereich

Der Diagrammbereich enthält alle derzeit geöffneten Diagrammfenster. Informationen zum Erstellen neuer Diagramme, zum Öffnen bestehender und zum Bearbeiten von Elementen im Diagramm finden Sie im Kapitel Anleitung zur Modellierung von... 104.

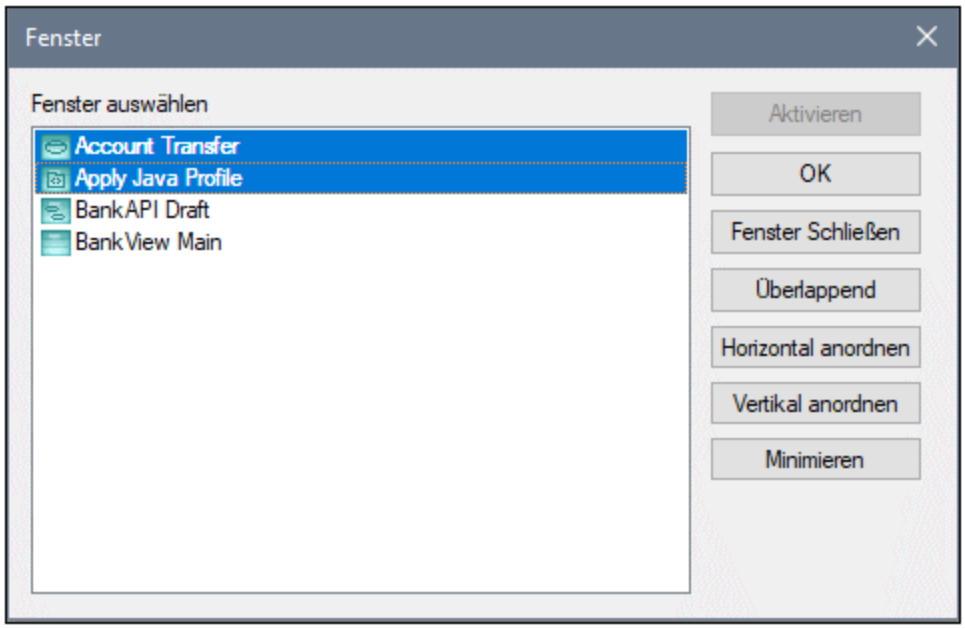
In der Abbildung unten sehen Sie den Diagrammbereich, in dem vier Fenster geöffnet sind und mit Hilfe des Menübefehls **Fenster | Überlappend** positioniert werden.



Diagrammbereich

Wenn Sie mit der rechten Maustaste im unteren Bereich des Diagrammbereichs auf das entsprechende Fensterregister klicken, stehen verschiedene Befehle für das aktuelle Diagrammfenster zur Verfügung.

Um diverse Befehle auf Fenster innerhalb des Diagrammbereichs anzuwenden, verwenden Sie die Befehle im Menü **Fenster**. Auch im Dialogfeld "Fenster" (Aufruf über den Menübefehl **Fenster | Fenster**) stehen einige Befehle zu Fenstern zur Verfügung.



Dialogfeld "Fenster"

Um im obigen Dialogfeld mehrere Fenster auszuwählen, halten Sie die **Strg**-Taste gedrückt, während Sie auf die entsprechenden Einträge klicken.

Um der Reihe nach alle offenen Diagrammfenster aufzurufen, drücken Sie **Strg+Tab**.

© 2016-2022 Altova GmbH

4 UModel Befehlszeilenschnittstelle

Zusätzlich zur grafischen Benutzeroberfläche verfügt UModel auch über eine Befehlszeilenschnittstelle. Um die Befehlszeilenschnittstelle aufzurufen, führen Sie die Datei **UModelBatch.exe** aus dem Ordner **C:\Programme\Altova\UModel2023** aus. Wenn Sie die UModel 32-Bit-Version auf einem 64-Bit-Betriebssystem ausführen, so lautet der Pfad **C:\Programme (x86)\Altova\UModel2023**.

Die Befehlszeilenparametersyntax wird unten gezeigt und kann im Fenster zur Befehlseingabe angezeigt werden, wenn Sie eingeben: **umodelbatch /?**

Hinweis: Wenn der Pfad oder Dateiname Leerzeichen enthält, setzen Sie den Pfad/Dateinamen bitte in Anführungszeichen, d.h. "C:\Programme\...\Dateiname"

```
usage: UModelBatch.exe [project] [options]

/? oder /help ... diese Hilfeinformationen anzeigen

Projekt     ... Projektdatei (*.ump)
/new[=Datei]... neues Projekt erstellen/speichern/unter einem neuen Namen speichern,
siehe Erstellen, Laden und Speichern von Projekten im Batch-Modus 102
/set        ... Optionen permanent festlegen
/gui        ... UModel-Benutzeroberfläche anzeigen

Befehle (in der angeführten Reihenfolge ausgeführt):
/chk        ... Projektsyntax überprüfen
/isd=Pfad   ... Quellverzeichnis importieren
/isp=Datei  ... Quellprojektdatei importieren
(*.project,*.xml,*.jpx,*.csproj,*.csdproj,*.vcxproj,*.vbproj,*.vbdproj,*.sln,*)
/ibt=Liste  ... Binärtypen importieren (Liste der Binär[Typnamen] definieren)
                (';'=Trennzeichen, '*'=alle Typen, '#' vor Assembly-Namen)
/ixd=Pfad   ... XML-Schemaverzeichnis importieren
/ixs=Datei  ... XML-Schemadatei importieren (*.xsd)
/m2c        ... Programmcode anhand von Modell aktualisieren (exportieren/Forward
Engineering)
/c2m        ... Modell anhand von Programmcode aktualisieren (importieren/Reverse
Engineering)
/ixf=Datei  ... XMI-Datei importieren
/exf=Datei  ... XMI-Datei exportieren
/inc=Datei  ... Datei inkludieren
/mrg=Datei  ... Datei zusammenführen
/doc=Datei  ... Dokumentation in angegebene Datei schreiben
/lue[=cpri] ... alle Elemente auflisten, die in keinem Diagramm verwendet werden (d.h.
nicht verwendete)
/ldg        ... alle Diagramme auflisten
/lcl        ... alle Klassen auflisten
/lsp        ... alle freigegebenen Pakete auflisten
/lip        ... alle inkludierten Pakete auflisten

Optionen zum Speichern als neues Projekt:
/npad=opt   ... relative Dateipfade anpassen (Yes | No | MakeAbsolute)

Optionen für Import-Befehle:
```

```
/iclg[=lang ... Codesprache (Java1.4 | Java5.0 | Java6.0 | Java7.0 | Java8.0 | Java9.0
|
                                Java10.0 | Java11.0 | Java12.0 | Java13.0 | Java14.0 |
Java15.0 |
                                C#1.2 | C#2.0 | C#3.0 | C#4.0 | C#5.0 | C#6.0 | C#7.0 |
C#7.1 | C#7.2 | C#7.3 | C#8.0 | C#9.0 |
                                VB7.1 | VB8.0 | VB9.0 |
                                C++98 | C++11 | C++14 | C++17)
/ipsd[=0|1] ... Unterverzeichnisse verarbeiten (rekursiv)
/irpf[=0|1] ... relativ zur UModel-Projektdatei importieren
/ijdc[=0|1] ... JavaDocs als Java-Kommentare
/icdc[=0|1] ... DocComments als C#-Kommentare
/icds[=lst] ... C# definierte Symbole
/ivdc[=0|1] ... DocComments als VB-Kommentare
/ivds[=lst] ... VB-definierte Symbole (benutzerdefinierte Konstanten)
/icppdm[=lst] ... C++-definierte Makros
/icpphi[=0|1] ... schreibgeschützte C++-Header-Dateien
/icpphc[=0|1] ... .h-Dateien als .cpp-Dateien behandeln
/icppms[=0|1] ... C++ Microsoft Compiler-Kompatibilität aktivieren compatibility
/icppmv[=ver] ... zu verwendende MSVC-Version (1900 | 1800 | 1700 | 1600 | 1500 | 1400
| 1310 | 1300 | 1200)
/icppsy[=0|1] ... C++ System Include-Dateien automatisch ermitteln
/icppid[=lst] ... Liste der zu verwendenden C++ Include-Verzeichnisse
/icppsd[=lst] ... Liste der zu verwendenden C++ System-Include-Verzeichnisse
/icppag[=arg] ... zusätzliche C++-Argumente für den Compiler
/imrg[=0|1] ... zusammengeführte synchronisieren
/iudf[=0|1] ... Verzeichnisfilter verwenden
/iflt[=lst] ... Verzeichnisfilter (Voreinstellungen /iudf)


Optionen für den Import von Binärtypen (nach /iclg):
/ibrt=vers  ... Runtime Version
/ibpv=Pfad  ... Variable PATH zum Durchsuchen von nativen Code-Bibliotheken außer
Kraft setzen
/ibro[=0|1] ... nur Reflection-Kontext verwenden
/ibua[=0|1] ... hinzugefügte referenzierte Typen mit Paketfilter verwenden
/ibar[=flt] ... referenzierte Typpaketfilter hinzufügen (presets /ibua)
/ibot[=0|1] ... nur Typen importieren
/ibuv[=0|1] ... Mindestsichtbarkeitsfilter verwenden
/ibmv[=key] ... Schlüsselwort der erforderlichen Mindestsichtbarkeit (presets /ibuv)
/ibsa[=0|1] ... Attributabschnitte / Annotations-Modifier unterdrücken
/iboa[=0|1] ... nur ein Attribut pro Attributabschnitt erstellen
/ibss[=0|1] ... 'Attribute'-Suffix bei Attributtypnamen unterdrücken


Optionen für die Diagrammgenerierung:
/dgen[=0|1] ... Diagramme generieren
/dopn[=0|1] ... generierte Diagramme öffnen
/dsac[=0|1] ... Attributbereich anzeigen
/dsoc[=0|1] ... Operation-Bereich anzeigen
/dscc[=0|1] ... Bereiche für geschachtelte Classifier anzeigen
/dstv[=0|1] ... Eigenschaftswerte anzeigen
/dudp[=0|1] ... .NET-Eigenschaftsbereich verwenden
/dspd[=0|1] ... .NET-Eigenschaftsbereich anzeigen
```

 © 2016-2022 Altova GmbH

```
Optionen für Export-Befehle:
/ejdc[=0|1] ... Java-Kommentare als JavaDocs
/ecdc[=0|1] ... C#-Kommentare als DocComments
/evdc[=0|1] ... VB-Kommentare als DocComments
/espl[=0|1] ... benutzerdefinierte SPL-Vorlagen verwenden
/ecod[=0|1] ... gelöschte auskommentieren
/emrg[=0|1] ... zusammengeführte synchronisieren
/egfn[=0|1] ... fehlende Dateinamen generieren
/eusc[=0|1] ... Syntaxüberprüfung verwenden

Optionen für den XMI-Export:
/exid[=0|1] ... UUIDs exportieren
/exex[=0|1] ... UModel-spezifische Erweiterungen exportieren
/exdg[=0|1] ... Diagramme exportieren (presets /exex)
/exuv[=ver] ... UML-Version (UML2.0 | UML2.1.2 | UML2.2 | UML2.3 | UML2.4 | UML2.5 |
UML2.5.1)
)

Optionen für die Dateizusammenführung:
/mcan=Datei  ... gemeinsame Vorgängerdatei

Optionen für die Dokumentationsgenerierung:
/doof=fmt  ... Ausgabeformat (HTML | RTF | MSWORD | PDF )
/dsps=Datei      ...SPS-Designdatei
```

Beispiel 1: Importieren von Java-Quellcode unter Beibehaltung der Einstellungen

Der folgende Befehl importiert Quellcode und erstellt eine neue Projektdatei. Beachten Sie, dass der Projektpfad Leerzeichen enthält und innerhalb von Anführungszeichen steht.

```
"C:\Programme\Altova\UModel2023\UModelBatch.exe" /new="C:\My
Projects\Fred.ump" /isd="X:TestCases\UModel\Fred" /set /gui /iclg=Java8.0 /ipsd=1 /ijdc=1
/dgen=1 /dopn=1 /dmax=5 /chk
```

Im folgenden sehen Sie eine Liste aller Optionen mit ihrer Beschreibung:

`/new`	Gibt an, dass die neu erstellte Projektdatei in **C:\My Projects** den Namen "Fred.ump" erhalten soll.
`/isd`	Gibt an, dass das Quellverzeichnis **X:\TestCases\UModel\Fred** sein soll.
`/set`	Gibt an, dass alle in der Befehlszeile verwendeten Optionen in der Registrierungsdatei gespeichert werden sollen (Wenn UModel anschließend gestartet wird, werden diese Einstellungen die Standardeinstellungen).
`/gui`	die grafische Benutzeroberfläche von UModel bei der Batch-Verarbeitung anzeigen
`/iclg`	UModel importiert den Code als Java 8.0
`/ipsd=1`	alle im Parameter **/isd** definierten Unterverzeichnisse des Root-Verzeichnisses rekursiv verarbeiten
`/ijdc=1`	gegebenenfalls JavaDoc anhand von Kommentaren erstellen
`/dgen=1`	Diagramme generieren
`/dopn=1`	generierte Diagramme öffnen
`/chk`	Syntaxüberprüfung durchführen

Beispiel 2: Synchronisieren von Code anhand des Modells

Der folgende Befehl aktualisiert Code anhand einer vorhandenen Projektdatei ("**C:\UModel\Fred.ump**").

```
"C:\Programme\Altova\UModel2023\UModelBatch.exe" "C:
\UModel\Fred.ump" /m2c /ejdc=1 /ecod=1 /emrg=1 /egfn=1 /eusc=1
```

Die Optionen haben dieselbe Bedeutung, wie in den vorhergehenden Beispielen:

`/m2c`	Code anhand von Modell aktualisieren
`/ejdc`	Kommentare im Projektmodell sollen als JavaDoc generiert werden
`/ecod=1`	gelöschten Code auskommentieren
`/emrg=1`	zusammengeführten Code synchronisieren

© 2016-2022 Altova GmbH

`/egfn=1`	fehlende Dateinamen im Projekt generieren
`/eusc=1`	Syntaxüberprüfung verwenden

Beispiel 3: Importieren von Java-Binärdateien in das Modell

Angenommen, das Verzeichnis **C:\JavaProject\bin** enthält einige Java-Klassen-Binärdateien, die Sie in UModel importieren möchten. Führen Sie dazu den folgenden Befehl aus:

```
"<C:\Programme\Altova\UModel2023\UModelBatch.exe>" /new="C:
\JavaProject\Result.ump" /ibt=*C:
\JavaProject\bin /iclg=Java8.0 /ibrt=JDK1.8.0_144 /dgen=1 /chk
```

Es werden die folgenden Optionen verwendet:

`/new`	Erstellt unter dem angegebenen Pfad ein neues UModel-Projekt.
`/ibt`	Weist UModel an, Binärtypen zu importieren. Das Sternchen vor dem Pfad gibt an, dass alle Binärtypen unter diesem Pfad importiert werden sollen.
`/iclg`	Definiert die Codegenerierungssprache (in diesem Beispiel "Java8.0").

`/ibrt`	Definiert das Runtime Environment (in diesem Beispiel "JDK1.8.0_144"). Dies ist derselbe Wert, der im Dialogfeld "Binärtypen importieren" angezeigt wird, siehe Importieren von Java-, C#- und VB.NET-Binärdateien 206. Sie können auch einen Wert wie z.B. "jdk-10.0.1", wie in der `JAVA_HOME` Umgebungsvariablen definiert, verwenden.
	
	Für C# können Sie den Wert `/ibrt:any` oder Werte, wie Sie in der "Runtime" Dropdown-Liste der Benutzeroberfläche angezeigt werden, verwenden. Stellen Sie dabei sicher, dass Sie alle Leerzeichen eliminieren. Beispiele:
	
	`/ibrt:any`
	`/ibrt:.NET5`
	`/ibrt:.NETFramework4.8(v4.8.3752)`
	
	Die Option "any" ist dieselbe wie wenn Sie in der Runtime-Dropdown-Liste "beliebige (Disassembler verwenden)" auswählen und ist die empfohlene Option.

`/dgen=1`	Generiert Diagramme.
`/chk`	Führt nach dem Import eine Syntaxüberprüfung durch.

4.1 Datei: New / Load / Save-Optionen

Wenn Sie **UModelBatch.exe** mit einem Befehl wie `UModelBatch MeinProjekt.ump` ausführen, können Sie die folgenden Parameter verwenden:

`/new`	Dieser Parameter definiert den Pfad und Dateinamen der neu zu erstellenden UModel-Projektdatei (*.ump). Sie können mit diesem Befehl auch ein vorhandenes Projekt laden und unter einem anderen Namen speichern, z.B.: `UmodelBatch.exe MyFile.ump /new=MyBackupFile.ump`
`/set`	Dieser Parameter überschreibt die aktuellen Standardeinstellungen in der Registry durch die von Ihnen definierten Optionen.
`/gui`	Dieser Parameter zeigt die grafische Benutzeroberfläche von UModel während des Batch-Vorgangs an.

In den folgenden Beispielen wird gezeigt, wie Sie Projekte im vollständigen Batch-Modus (d.h. ohne, dass der Parameter `/gui` gesetzt ist) erstellen, laden oder speichern.

new

UModelBatch /new=xxx.ump (options)
Erstellt ein neues Projekt, führt die Optionen aus, xxx.ump wird immer (unabhängig von den Optionen) gespeichert

auto save

UModelBatch xxx.ump (options)
Lädt das Projekt xxx.ump, führt die Optionen aus, xxx.ump wird **nur dann** gespeichert, wenn sich das Dokument geändert hat (wie **/ibt**)

save

UModelBatch xxx.ump (options) /new
Lädt das Projekt xxx.ump, führt die Optionen aus, xxx.ump wird **immer** gespeichert (unabhängig von den Optionen)

save as

UModelBatch xxx.ump (options) /new=yyy.ump
Lädt das Projekt xxx.ump, führt die Optionen aus, speichert xxx.ump immer als yyy.ump (unabhängig von den Optionen)

In den folgenden Beispielen wird gezeigt, wie Sie Projekte im Batch-Modus, wobei die Benutzeroberfläche von UModel angezeigt wird (Parameter `/gui` ist gesetzt) erstellen, laden oder speichern.

new

UModelBatch /gui /new (options)

© 2016-2022 Altova GmbH

Erstellt ein neues Projekt, führt die Optionen aus, nichts wird gespeichert, die GUI bleibt offen

save new

UModelBatch /gui /new=xxx.ump (options)
Erstellt ein neues Projekt, führt die Optionen aus, xxx.ump wird gespeichert, die GUI bleibt offen

user mode

UModelBatch /gui xxx.ump (options)
Lädt das Projekt xxx.ump, führt die Optionen aus, nichts wird gespeichert, die GUI bleibt offen

save

UModelBatch /gui xxx.ump (options) /new
Lädt das Projekt xxx.ump, führt die Optionen aus, xxx.ump wird gespeichert, die GUI bleibt offen

save as

UModelBatch /gui xxx.ump (options) /new=yyy.ump
Lädt das Projekt xxx.ump, führt die Optionen aus, speichert xxx.ump als yyy.ump, die GUI bleibt offen

Das Projekt wird erfolgreich gespeichert, wenn bei Ausführung der Optionen kein schwerwiegender Fehler auftritt.

5 Anleitung zur Modellierung von...

Altova Website: UML-Modellierung

In diesem Kapitel wird erläutert, wie Sie UML-Elemente, Diagramme und Beziehungen über die grafische Benutzeroberfläche von UModel erstellen und bearbeiten. Es dient als Anleitung zur Modellierung mit UModel. Die Anleitungen in diesem Kapitel gelten für UModel als ganzes und sind nicht für ein bestimmtes Element oder Diagramm spezifisch, es sei denn, dies ist explizit erwähnt. Informationen zu den einzelnen Diagrammtypen (gruppiert nach Diagrammtyp) finden Sie im Kapitel UML-Diagramme 301.

Die Informationen in diesem Kapitel sind in die folgenden Kategorien gegliedert: Elemente, Diagramme, Beziehungen und Stereotype.

Elemente	Diagramme	Beziehungen	Stereotype
Erstellen von Elementen 106	Erstellen von Diagrammen 122	Erstellen von Beziehungen 133	Stereotype und Eigenschaftswerte 145
Einfügen von Elementen aus dem Modell in ein Diagramm 107	Generieren von Diagrammen 123	Ändern des Stils von Linien und Beziehungen 134	Eigenschaftswerte 146
Umbenennen, Verschieben und Kopieren von Elementen 109	Öffnen von Diagrammen 125	Anzeigen von Elementbeziehungen 136	Anwenden von Stereotypen 147
Löschen von Elementen 110	Löschen von Diagrammen 126	Assoziationen 137	Anzeigen oder Ausblenden von Eigenschaftswerten 149
Konvertieren von Elementen 111	Ändern des Stils von Diagrammen 127	Collection-Assoziationen 140	
Suchen und Ersetzen von Text 111	Ausrichten von Modellierungselementen und Anpassen der Größe 128	Enthältbeziehung 143	
Überprüfen, wo und ob Elemente verwendet werden 113	Typ-Autokomplettierung in Klassen 130		
Einschränken von Elementen 114	Vergrößern und Verkleinern von Diagrammen 132		
Erstellen von Hyperlinks zu Elementen 115			

 © 2016-2022 Altova GmbH

Elemente	Diagramme	Beziehungen	Stereotype
Dokumentieren von Elementen 118			
Ändern des Stils von Elementen 119			

Anmerkung: UModel enthält eine Reihe von Beispielprojekten, anhand welcher Sie die Grundlagen der Modellierung erlernen und die grafische Benutzeroberfläche von UModel kennenlernen können. Diese Projekte befinden sich im folgenden Ordner: **C:\Benutzer\<Benutzername>\Dokumente\Altova\UModel2023\UModelExamples**.

5.1 Elemente

5.1.1 Erstellen von Elementen

Sie können Elemente mit UModel auf folgende Arten erstellen:

- Über das Fenster Modell-Struktur 80. Bei dieser Methode werden die Elemente nur zum Modell hinzugefügt und Sie können diese später gegebenenfalls zu Diagrammen hinzufügen.
- Über das Diagrammfenster. Elemente, die zu einem Diagramm hinzugefügt werden, werden automatisch auch zum Modell hinzugefügt. Falls Sie ein Element später löschen müssen, können Sie wählen, ob Sie es nur aus dem Diagramm oder auch aus dem Modell löschen möchten.

So fügen Sie Elemente über das Fenster "Modell-Struktur" hinzu:

- Klicken Sie im Fenster Modell-Struktur 80 (oder im Fenster Favoriten 85) mit der rechten Maustaste auf das Element (z.B. Paket), unter dem das neue Element angezeigt werden soll und wählen Sie im Kontextmenü den Befehl **Neues Element | <Elementname>**. Um z.B. unterhalb des Pakets "Root" ein neues Paket hinzuzufügen, klicken Sie mit der rechten Maustaste auf das Paket "Root" und wählen Sie **Neues Element | Paket**.

So fügen Sie Elemente über das Diagrammfenster hinzu:

1. Erstellen Sie ein neues Diagramm (siehe Erstellen von Diagrammen 122) oder öffnen Sie ein vorhandenes (siehe Öffnen von Diagrammen 125).
2. Wählen Sie eine der folgenden Methoden:
   a. Klicken Sie mit der rechten Maustaste in das Diagramm und wählen Sie den Kontextmenübefehl **Neu | <Elementname>**.
   b. Klicken Sie auf die Symbolleistenschaltfläche des gewünschten Elements und klicken Sie in das Diagramm. Um mehrere Elemente desselben Typs einzufügen, halten Sie die **Strg**-Taste gedrückt, bevor Sie in das Diagramm klicken.

Pakete

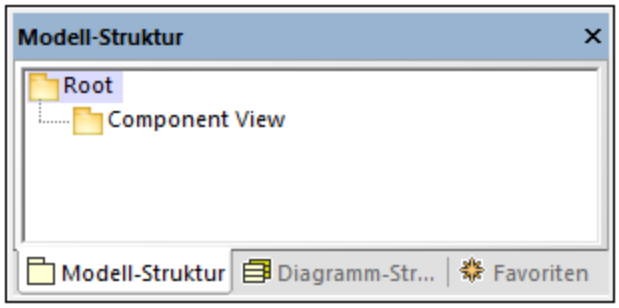
Sie werden beim Modellieren von Elementen wahrscheinlich mit Paketen öfter als mit anderen Elementen arbeiten müssen. Jeder mit einem Ordnersymbol markierte Eintrag im Fenster "Modell-Struktur" steht für ein UML-Paket. Pakete in UModel dienen als Container für alle anderen UML-Modellierungselemente (einschließlich Diagramme, Klassen usw.) und verhalten sich folgendermaßen:

- Sie können an jeder Stelle der Modell-Struktur erstellt werden.
- Sie können in andere Pakete (und auch in gültige Modelldiagramme) verschoben und kopiert werden, siehe Umbenennen, Verschieben und Kopieren von Elementen 109.
- Sie können beim Generieren oder Synchronisieren von Code und Modell als Quell- oder Zielelemente verwendet werden, siehe Forward Engineering (Modell zu Code) 59 und Reverse Engineering (Code zu Modell) 69.

Wenn Sie ein neues UModel-Projekt erstellen, stehen standardmäßig zwei Pakete zur Verfügung: "Root" und "Component View". Diese beiden Pakete sind die einzigen, die nicht umbenannt oder gelöscht werden können. Das "Root"-Paket dient als Ausgangspunkt für die Modellierung aller anderen Elemente, während das

 © 2016-2022 Altova GmbH

"Component View"-Paket für das Code Engineering benötigt wird.



UModel-Standardpakete

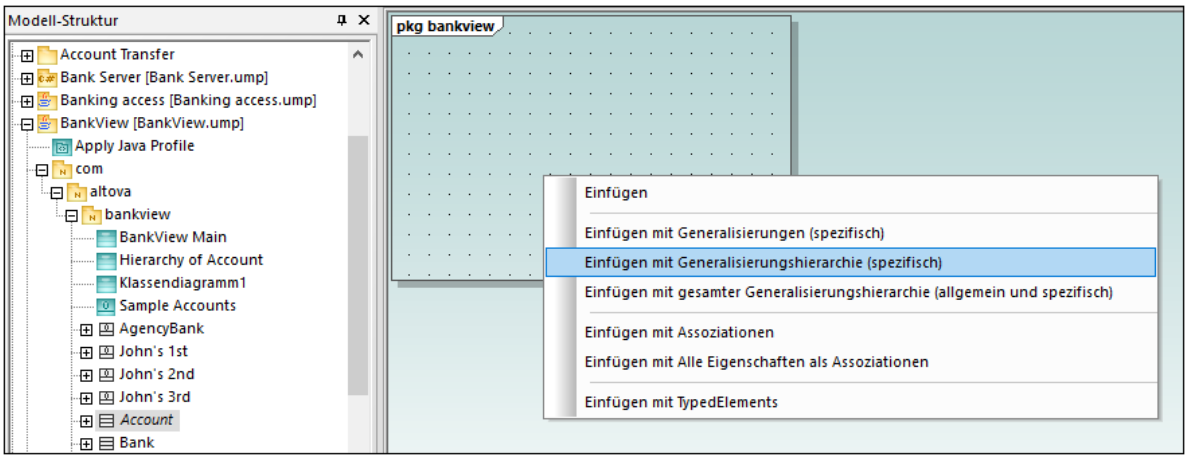
5.1.2 Einfügen von Elementen aus dem Modell in ein Diagramm

Im Modell bereits vorhandene Elemente können einzeln oder als Gruppe in ein Diagramm eingefügt werden. Um mehrere Elemente aus dem Fenster "Modell-Struktur" zu markieren, halten Sie die **Strg**-Taste gedrückt und klicken Sie auf die einzelnen Elemente. Zum Einfügen der Elemente in das Diagramm stehen zwei verschiedene Methoden zur Verfügung: links ziehen und rechts ziehen.

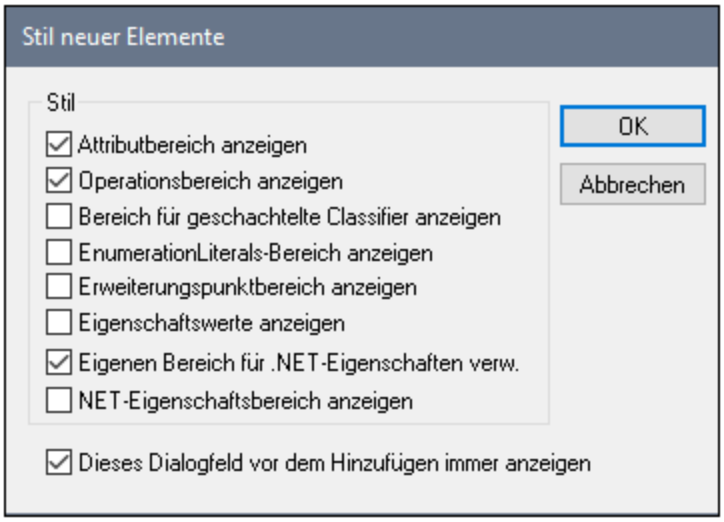
- **Links ziehen** (Gedrückt-halten der linken Maustaste und Loslassen der Maustaste im Diagramm): Fügt Elemente direkt an der Cursorposition ein. In diesem Fall werden automatisch alle Assoziationen, Abhängigkeiten usw., die zwischen den aktuell eingefügten und dem neuen Element bestehen, angezeigt.
- **Rechts ziehen** (Gedrückt-halten der rechten Maustaste und Loslassen der Maustaste im Diagramm): Öffnet ein Kontextmenü, aus dem Sie die Assoziationen und Generalisierungen, die angezeigt werden sollen, auswählen können.

Angenommen, Sie möchten anhand einer bereits im Modell vorhandenen Klasse ein neues Klassendiagramm erstellen. Öffnen Sie dazu das Beispielprojekt **Bank_MultiLanguage.ump** aus dem folgenden Ordner: **C:\Benutzer\<Benutzername>\Dokumente\Altova\UModel2023\UModelExamples**. Um z.B. das Diagramm "Account Hierarchy" in einem neuen Klassendiagramm zu replizieren, gehen Sie folgendermaßen vor:

1. Rechtsklicken Sie auf das **bankview**-Paket und wählen Sie **Neues Diagramm | Klassendiagramm**.
2. Suchen Sie in der Modell-Struktur die abstrakte Klasse `Account` und verwenden Sie den "**Rechts ziehen**"-Vorgang, um sie in das neue Diagramm zu platzieren. In diesem Beispiel soll die Klasse zusammen mit ihren abgeleiteten Klassen angezeigt werden. Wählen Sie dazu im Kontextmenü den Befehl **Einfügen mit Generalisierungshierarchie (spezifisch)**.



3. Aktivieren oder deaktivieren Sie die Kontrollkästchen für die Einträge, die in den Elementen angezeigt werden sollen.



4. Klicken Sie auf OK. Die Klasse `Account` und **ihre drei Unterklassen** werden alle auf dem Diagrammregister eingefügt. Die Generalisierungspfeile werden ebenfalls automatisch angezeigt. Um die Klassen automatisch im Diagramm anzuordnen, starten Sie den Befehl **Layout | Automatisches Layout | Hierarchisch**.

Hätten Sie anstelle des Befehls **Einfügen mit Generalisierungshierarchie (spezifisch)** den Befehl **Einfügen** gewählt, wäre die Klasse ohne abgeleitete Klassen zum Diagramm hinzugefügt worden. Beachten Sie, dass Sie die Generalisierungshierarchie auch später anzeigen können. Gehen Sie dazu folgendermaßen vor:

- Klicken Sie mit der rechten Maustaste im Diagramm auf die Klasse `Account` und wählen Sie im Kontextmenü den Befehl **Anzeigen | Generalisierungshierarchie**. Dadurch werden auch die abgeleiteten Klassen in das Diagramm eingefügt.

 © 2016-2022 Altova GmbH

5.1.3 Umbenennen, Verschieben und Kopieren von Elementen

Sie können Elemente im Fenster Modell-Struktur 80 und innerhalb von Diagrammen desselben Typs ausschneiden, kopieren, umbenennen und verschieben. Diese Aktionen sind, falls anwendbar, auch zwischen Diagrammen unterschiedlichen Typs möglich. Sie können Elemente auch aus dem Fenster "Modell-Struktur" in ein Diagramm kopieren oder verschieben, vorausgesetzt das entsprechende Element ist gemäß der UML-Spezifikation in diesem Diagramm zulässig.

So benennen Sie ein Element um:

- Doppelklicken Sie auf den Elementnamen und bearbeiten Sie ihn.
- Klicken Sie alternativ dazu auf das Element und drücken Sie **F2**.

Die oben beschriebenen Verfahren gelten für jedes Fenster, in dem das Element angezeigt wird, darunter auch die Fenster "Modell-Struktur", "Eigenschaften" und das Diagrammfenster.

> Die Pakete "Root" und "Component View" werden immer im Fenster "Modell-Struktur" angezeigt und können nicht umbenannt oder gelöscht werden.

So kopieren oder verschieben Sie Elemente:

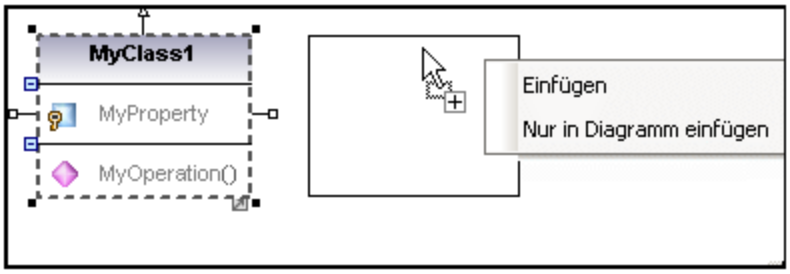
- Verwenden Sie die Windows-Standardbefehle **Ausschneiden, Kopieren** oder **Einfügen**. Diese Befehle können über die Tastaturkürzel (**Strg+X**, **Strg+C** bzw. **Strg+V**), über die entsprechenden Symbolleisten-Schaltflächen sowie über das Menü **Bearbeiten** aufgerufen werden.
- Alternativ dazu können Sie ein Element in ein Zielpaket (oder Element) ziehen. Durch das Ziehen eines Elements verschieben Sie es. Halten Sie die **Strg**-Taste beim Ziehen gedrückt, um es zu kopieren.

So können Sie etwa ein Mitglied einer Klasse in einem Diagramm in eine andere Klasse verschieben, indem Sie es mit der Maus von der Quellklasse in die Zielklasse ziehen. Um das Mitglied der Klasse zu kopieren, anstatt es zu verschieben, wählen Sie es aus und ziehen Sie es in die Zielklasse, während Sie die **Strg**-Taste gedrückt halten.

Wenn Sie eine Klasse in dasselbe Paket einfügen, wird eine laufende Nummer an die neue Klasse angehängt, z.B. "MyClass1". Wenn Sie eine Eigenschaft innerhalb derselben Klasse einfügen, wird ebenfalls eine laufende Nummer an das Ende der Eigenschaft angehängt, z.B. "MyProperty1". Dies gilt auch für andere Klassenmitglieder wie Operationen und Enumerationen oder beim Einfügen von Elementen in dasselbe Diagramm, vorausgesetzt das Diagramm gehört zum selben Paket wie die eingefügten Elemente.

Wenn Sie eine Klasse in ein anderes Paket einfügen, erhält die neue Klasse denselben Namen wie die ursprüngliche Klasse. Dies gilt auch für das Kopieren von Klassenmitgliedern (wie Eigenschaften, Operationen, usw.) in eine andere Klasse.

Standardmäßig wird jedes Element, das in ein Diagramm eingefügt wird, automatisch auch zum Modell hinzugefügt (und daher auch im Fenster "Modell-Struktur" angezeigt). Sie können ein Element aber auch nur in das aktuelle Diagramm kopieren, ohne es zum Modell hinzuzufügen. Kopieren Sie dazu zuerst das Element, klicken Sie mit der rechten Maustaste in das Diagramm und wählen Sie im Kontextmenü den Befehl **Nur in Diagramm einfügen**. Der Befehl **Nur in Diagramm einfügen** wird auch angezeigt, wenn Sie ein vorhandenes Element in dasselbe Diagramm ziehen, während Sie die **Strg**-Taste gedrückt halten.



Im obigen Beispiel erstellt **Einfügen** die neue Klasse im Diagramm und fügt sie auch zur Modell-Struktur hinzu, während mit **Nur in Diagramm einfügen** nur eine zweite Ansicht der Klasse im Diagramm angezeigt wird. Beachten Sie, dass es sich bei Kopien, die mit der zweiten Methode erstellt wurden, nur um zusätzliche Ansichten des Originalelements handelt, zu dem eine Verknüpfung erstellt wird, und nicht um eine eigenständige Kopie. (Wenn Sie z.B. eine Eigenschaft in der duplizierten Klasse umbenennen, wird diese Änderung automatisch auf die Originalklasse angewendet.)

5.1.4 Löschen von Elementen

Elemente können auf die folgenden Arten gelöscht werden:

- Über das Fenster "Modell-Struktur". Verwenden Sie diese Methode, wenn das Element sowohl aus dem Projekt als auch aus allen Diagrammen, in denen es vorkommt, gelöscht werden soll.
- Direkt über die Diagramme, in denen sie vorkommen. In diesem Fall können Sie auswählen, ob das Element nur aus dem Diagramm entfernt oder auch aus dem Modell (Projekt) gelöscht werden soll.

So löschen Sie Elemente aus dem Projekt und allen dazugehörigen Diagrammen (Methode 1):

1. Klicken Sie im Fenster "Modell-Struktur" auf das gewünschte Element. Halten Sie die **Strg**-Taste gedrückt, um mehrere Elemente auszuwählen.
2. Drücken Sie **Entf**.

So löschen Sie Elemente aus dem Projekt und allen dazugehörigen Diagrammen (Methode 2):

1. Öffnen Sie ein Diagramm und klicken Sie auf das gewünschte Element. Halten Sie die **Strg**-Taste gedrückt, um mehrere Elemente auszuwählen.
2. Drücken Sie **Entf**. Daraufhin wird ein Dialogfeld angezeigt, in dem Sie gefragt werden, ob Sie das Element sowohl aus dem Projekt als auch aus dem Diagramm löschen möchten.
3. Klicken Sie auf **Ja**. Das Element wird sowohl aus dem Diagramm als auch aus dem Projekt gelöscht.

So löschen Sie Elemente aus dem Diagramm, nicht aber aus dem Projekt:

1. Öffnen Sie ein Diagramm und klicken Sie auf das/die gewünschte/n Element/e. Halten Sie die **Strg**-Taste gedrückt, um mehrere Elemente auszuwählen.
2. Halten Sie die **Strg**-Taste gedrückt und drücken Sie **Entf**. Die Elemente werden aus dem Diagramm, nicht aber aus dem Projekt gelöscht.

 © 2016-2022 Altova GmbH

Bevor Sie Elemente aus einem Projekt löschen, sollten Sie eventuell überprüfen, ob sie in einem Diagramm verwendet werden.

- Klicken Sie mit der rechten Maustaste auf ein Element in der Modell-Struktur und wählen Sie im Kontextmenü den Befehl **Element in allen Diagrammen anzeigen**.

Auf dieselbe Weise können Sie, während ein Diagramm geöffnet ist, schnell ein Element in der Modell-Struktur auswählen:

- Klicken Sie mit der rechten Maustaste auf ein Element im Diagramm und wählen Sie im Kontextmenü den Befehl **In Modell-Struktur auswählen**.
- Klicken Sie alternativ dazu auf das Element im Diagramm und drücken Sie **F4**.

5.1.5 Konvertieren von Elementen

Einige der Elemente unterstützen eine schnelle Konvertierung in eine andere Elementart. Dies kann sich als nützlich erweisen, wenn Sie z.B. begonnen haben, eine Klasse zu erstellen, diese später aber in eine Schnittstelle umwandeln möchten oder umgekehrt. Die folgenden Arten von Elementen unterstützen die Konvertierung in ein beliebiges anderes Element in der Liste:

- Klasse
- Schnittstelle
- Enumeration
- Primitivtyp
- Datentyp

Die oben aufgelisteten Elementarten können entweder über das Diagrammfenster84 oder die Modell-Struktur80 konvertiert werden.

So konvertieren Sie Elemente:

1. Öffnen Sie ein Diagramm, das Klassen, Schnittstellen, Enumerationen, Primitivtypen oder Datentypen enthält (z.B. ein Klassendiagramm). Navigieren Sie alternativ dazu in der Modell-Struktur zu einer dieser Elementarten.
2. Klicken Sie mit der rechten Maustaste auf das gewünschte Element (z.B. eine Klasse) und wählen Sie im Kontextmenü den Befehl **Konvertieren in | <Elementart>**.

Der Name des Elements bleibt nach der Konvertierung erhalten. Wenn möglich, bleiben auch die mit dem Element verknüpften Daten erhalten. So bleiben etwa bei einer Konvertierung von einer Schnittstelle in eine Klasse oder einer Klasse in eine Schnittstelle Daten wie Eigenschaften oder Operationen erhalten. Bei einer Konvertierung von einer Klasse oder Schnittstelle in eine Enumeration gehen hingegen Daten verloren. In einem solchen Fall können Sie den vorherigen Zustand des Elements, falls nötig, mit Hilfe des Befehls **Rückgängig** (**Strg+Z**) wiederherstellen.

5.1.6 Suchen und Ersetzen von Text

Sie können in jedem der folgenden Fenster nach Modellierungselementen, Diagrammen, Text, usw. suchen.:

- Diagrammfenster

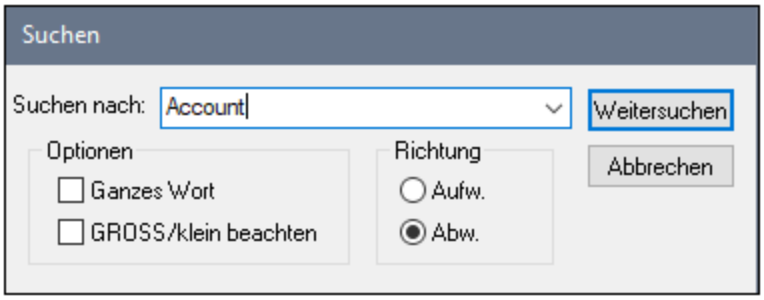
- Fenster "Modell-Struktur"
- Fenster "Diagramm-Struktur"
- Fenster "Favoriten"
- Fenster "Dokumentation"
- Fenster "Meldungen"

Die Suche wird dabei im Geltungsbereich des Fensters, in dem sich der Cursor gerade befindet, durchgeführt. Wenn Sie also in einem Diagramm nach Text suchen möchten, klicken Sie zuerst in das Diagramm. Wenn Sie nach einem Objekt im UModel-Projekt suchen möchten, klicken Sie zuerst in das Fenster "Modell-Struktur".

So suchen Sie nach Text oder Elementen:

1. Klicken Sie in das Fenster, in dem Sie Text suchen möchten.
2. Wählen Sie eine der folgenden Methoden:
   a. Geben Sie den Suchtext in das Textfeld der Hauptsymbolleiste ein und klicken Sie dann auf **Weitersuchen** oder drücken Sie **F3**. Um zum vorherigen Treffer zu gelangen, drücken Sie **Umschalt+F3**.

   b. Klicken Sie im Menü **Bearbeiten** auf **Suchen** (oder drücken Sie **Strg+F**).



Suchen und ersetzen

Sie können Text auch suchen und ersetzen (z.B. um Modellierungselemente schnell umzubenennen). Wenn das Element gefunden wird, wird es sowohl im Diagramm als auch in der Modell-Struktur markiert. Die Such- und Ersetzungsfunktion funktioniert in den folgenden Fenstern:

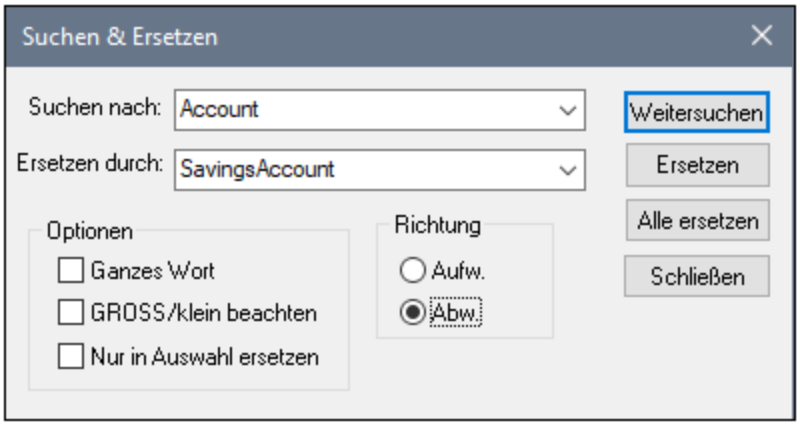
- Diagrammfenster
- Fenster "Modell-Struktur"
- Fenster "Diagramm-Struktur"
- Fenster "Favoriten"
- Fenster "Dokumentation"

So suchen und ersetzen Sie Text:

1. Klicken Sie in das Fenster, in dem Sie Text suchen und ersetzen möchten.
3. Wählen Sie eine der folgenden Methoden:
   a. Klicken Sie auf die Symbolleisten-Schaltfläche **Ersetzen**.

 © 2016-2022 Altova GmbH

b. Wählen Sie im Menü **Bearbeiten** den Befehl **Ersetzen** (oder drücken Sie **Strg+H**).



5.1.7 Überprüfen, wo und ob Elemente verwendet werden

Manchmal ist es beim Navigieren durch die Elemente in der Modell-Struktur hilfreich zu sehen, wo oder ob das Element in einem Modelldiagramm vorkommt. Um herauszufinden, wo Elemente verwendet werden, wählen Sie eine der folgenden Methoden:

- Klicken Sie mit der rechten Maustaste auf das Element im Fenster "Modell-Struktur" und wählen Sie den Befehl **Element in allen Diagrammen anzeigen** (oder, falls ein Diagramm bereits geöffnet ist, auf **Element in aktivem Diagramm anzeigen**).

Sie können auch im gesamten Projekt oder einzelnen Paketen nach Elementen suchen, die in keinem Diagramm verwendet werden.

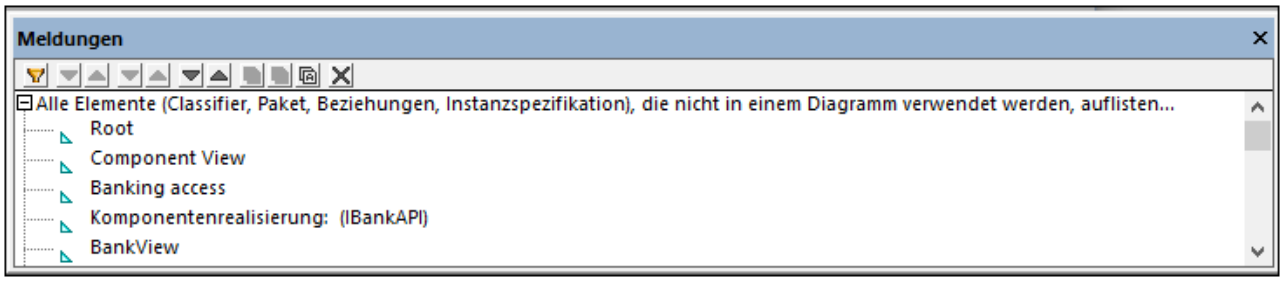
So suchen Sie im gesamten Projekt nicht verwendete Elemente:

- Klicken Sie im Menü **Projekt** auf **In keinem Diagramm verwendete Elemente auflisten.**

So suchen Sie in einem bestimmten Paket nicht verwendete Elemente:

- Klicken Sie mit der rechten Maustaste auf das gewünschte Paket und wählen Sie den Befehl **In keinem Diagramm verwendete Elemente auflisten.**

Daraufhin wird im Fenster "Meldungen" eine Liste nicht verwendeter Elemente angezeigt. Beachten Sie, dass die nicht verwendeten Elemente für das aktuell ausgewählte Paket und seine Unterpakete angezeigt werden. Objekte innerhalb von Klammern sind Elemente, die über **Extras | Optionen |** Register "**Ansicht**" als in der Liste der nicht verwendeten Elemente anzuzeigende Elemente konfiguriert wurden.



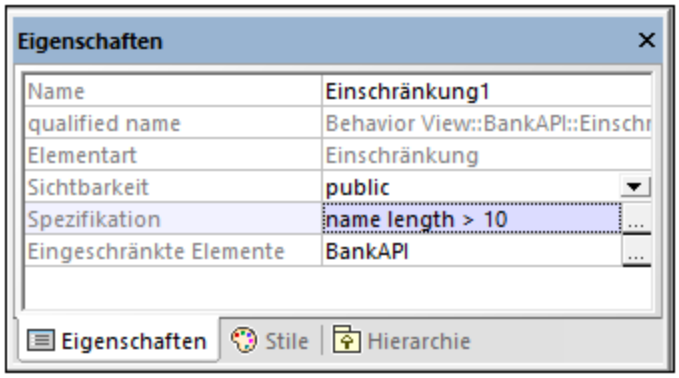
Klicken Sie im Fenster "Meldungen" auf den Elementnamen, um das Element in der Modell-Struktur zu finden.

5.1.8 Einschränken von Elementen

Es können für die meisten Modellelemente in UModel Einschränkungen definiert werden. Beachten Sie bitte, dass diese bei der Syntaxüberprüfung nicht berücksichtigt werden, da Einschränkungen nicht Teil der Java-Codegenerierung sind.

So schränken Sie ein Element (über die Modell-Struktur) ein:

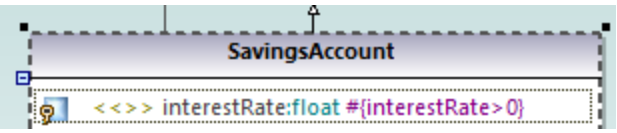
1. Rechtsklicken Sie auf das gewünschte Element und wählen Sie **Neues Element | Einschränkungen | Einschränkung**.
2. Geben Sie den Namen der Einschränkung ein und drücken Sie die **Eingabetaste**.
3. Geben Sie im Fenster "Eigenschaften" in das Feld "Spezifikation" die Einschränkung ein, `name length > 10`.



So schränken Sie ein Element (über ein Diagramm) ein:

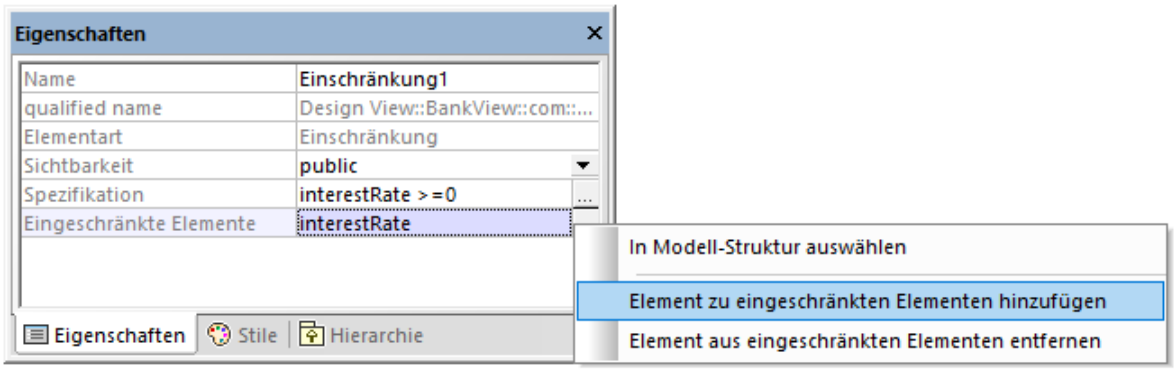
1. Doppelklicken Sie auf das jeweilige Element, um es editieren zu können.
2. Geben Sie #, gefolgt vom Einschränkungstext innerhalb einer geschweiften Klammer ein z.B. `#{interestRate >=0}`.

 © 2016-2022 Altova GmbH

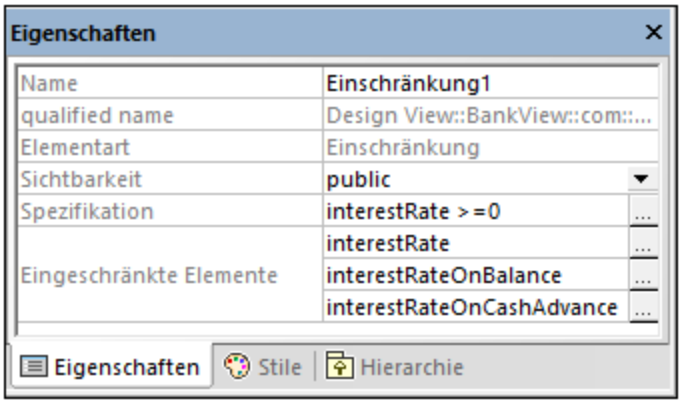


So weisen Sie eine Einschränkung mehreren Modellierungselementen zu:

1. Wählen Sie eine Einschränkung in der Modell-Struktur aus.
2. Klicken Sie im Fenster "Eigenschaften" mit der rechten Maustaste auf die Eigenschaft "eingeschränkte Elemente" und wählen Sie den Befehl **Element eingeschränkten Elementen hinzufügen**.



3. Wählen Sie das jeweilige Element aus, dem Sie die aktuelle Einschränkung zuweisen möchten. Halten Sie die **Strg**-Taste gedrückt, um mehrere Elemente auszuwählen.



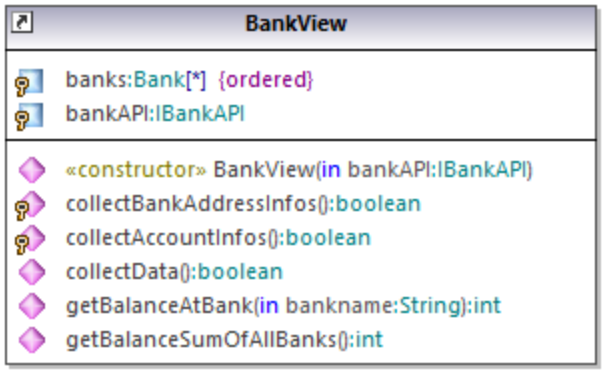
Das Feld "Eingeschränkte Elemente" enthält die Namen der Modellierungselemente, denen es zugewiesen wurde. In der obigen Abbildung wurde etwa `Einschränkung1` den folgenden Eigenschaften zugewiesen: `interestRate`, `interestRateOnBalance`, `interestRateOnCashAdvance`.

5.1.9 Erstellen von Hyperlinks zu Elementen

Sie können zwischen den meisten Modellierungselementen (mit Ausnahme von Linien) und jedem der folgenden Elemente Hyperlinks erstellen:

- Andere Elemente (entweder im Diagramm oder in der Modell-Struktur)
- Diagramme
- Externe Dateien, die nicht im Projekt enthalten sind (z.B. PDF-, Word- oder Excel-Dokumente, grafische Dateien, usw.)
- Webseiten

Ein einzelnes Element kann einen oder mehrere Hyperlinks jeder der oben aufgelisteten Arten haben. In einem Diagramm werden Elemente, die Hyperlinks enthalten, durch ein Hyperlink-Symbol neben dem Element (entweder in der rechten oder linken Ecke) gekennzeichnet. Um das Ziel des Hyperlink zu öffnen, klicken Sie mit der rechten Maustaste auf das Hyperlink-Symbol des Elements und wählen Sie das Ziel aus. Wenn nur ein Hyperlink definiert ist, können Sie auch einfach auf klicken und das Ziel direkt aufrufen.



Klasse, die Hyperlinks enthält

Tipp: Während Sie entweder über Hyperlinks oder auf andere Weise über die grafische Benutzeroberfläche von UModel navigieren, können Sie über die Symbolleisten-Schaltflächen **Zurück** bzw. **Weiter** einfach von einer Ansicht zur anderen weiter- oder zurückgehen.

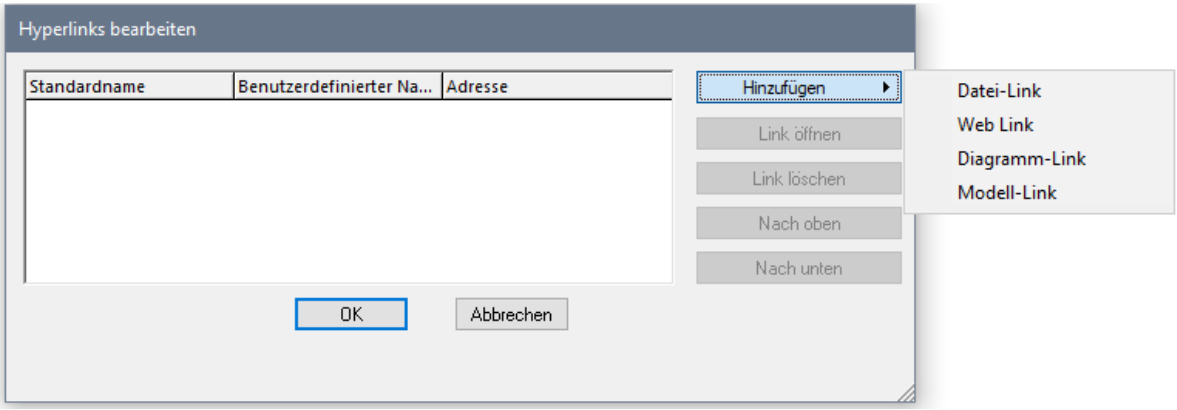
Wenn Sie Quellcode oder Binärdateien in ein Modell importieren, können Sie automatisch Hyperlinks zwischen abhängigen Paketen und Diagrammen generieren, vorausgesetzt, Sie haben im Import-Dialogfeld die entsprechenden Einstellungen ausgewählt. Nähere Informationen dazu finden Sie unter Importieren von Quellcode 194 und Importieren von Java-, C#- und VB.NET-Binärdateien 206. Außerdem können Sie beim Generieren von UML-Dokumentation anhand des Projekts auswählen, ob Hyperlinks in die generierte Ausgabe inkludiert werden sollen, siehe Generieren von UML-Dokumentation 290.

Sie können Hyperlinks nicht nur anhand der im Diagramm oder der Modell-Struktur angezeigten Elemente, sondern auch anhand von Text in Anmerkungen sowie anhand von Text im Fenster "Dokumentation" generieren. Eine Anleitung dazu finden Sie weiter unten.

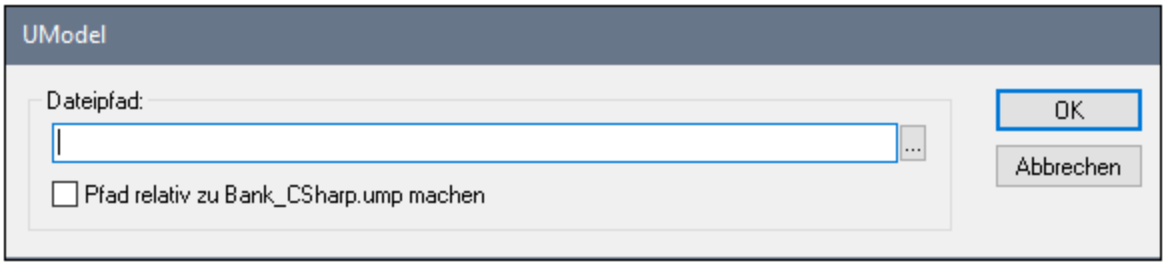
So erstellen Sie einen Hyperlink anhand eines Elements:

1. Klicken Sie mit der rechten Maustaste in einem Diagramm oder im Fenster "Modell-Struktur" auf ein Element und wählen Sie im Kontextmenü den Befehl **Hyperlinks | Hyperlinks einfügen/bearbeiten**.
2. Klicken Sie auf **Hinzufügen** und wählen Sie eine Hyperlinkart (Element-, Diagramm-, Datei- oder Web Link aus).

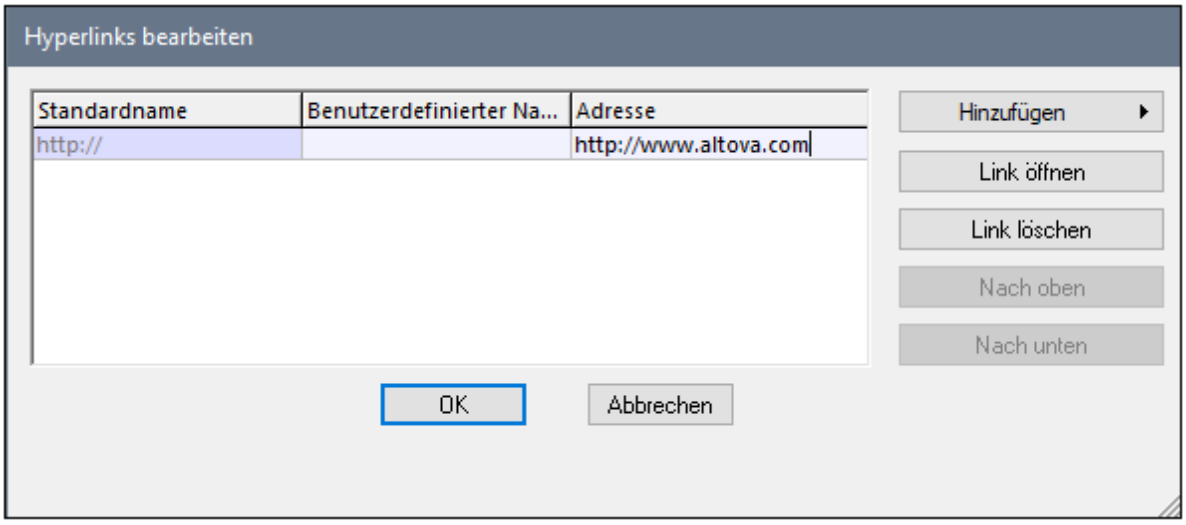
 © 2016-2022 Altova GmbH



3. Wählen Sie eine der folgenden Methoden:
   - Um ein Diagramm oder einen Hyperlink zu erstellen, wählen Sie das Zielelement oder -diagramm aus, wenn Sie dazu aufgefordert werden.
   - Um einen Datei-Hyperlink zu erstellen, klicken Sie auf die Auslassungszeichen und navigieren Sie zur Zieldatei.



   - Um einen Weblink zu erstellen, geben Sie die gewünschte URL in die Spalte "Adresse" des Dialogfelds ein, z.B:



4. Geben Sie optional einen benutzerdefinierten Linknamen in die Spalte "Benutzerdefinierter Name" ein. Falls einer definiert ist, wird anstelle des Zielpfads (oder der Adresse) dieser Name auf der grafischen Benutzeroberfläche von UModel angezeigt.

So erstellen Sie einen Hyperlink in einer Anmerkung:

- Wählen Sie einen Textbereich in der Anmerkung aus, klicken Sie mit der rechten Maustaste darauf und wählen Sie im Kontextmenü den Befehl **Hyperlinks einfügen/bearbeiten**. Auf dieselbe Art und Weise kann auch im Fenster "Dokumentation" ein Hyperlink erstellt werden.

So ändern oder entfernen Sie einen Hyperlink:

- Klicken Sie mit der rechten Maustaste auf das Hyperlinksymbol im Element (oder auf den mit einem Hyperlink versehenen Text) und verwenden Sie den entsprechenden Befehl im Dialogfeld "Hyperlinks bearbeiten".

5.1.10 Dokumentieren von Elementen

Sie können Dokumentationskommentare folgendermaßen zu Modellierungselementen hinzufügen:

- Klicken Sie (entweder im Diagramm oder in der Modell-Struktur) auf das Element.
- Geben Sie im Fenster "Dokumentation" Text ein.

Dokumentationstext wird zusammen mit dem Projekt gespeichert.

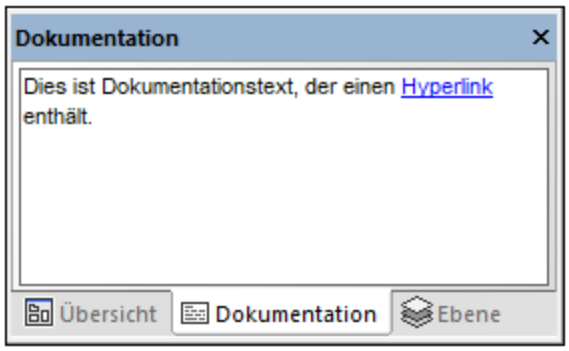
Wenn ein Element ausgewählt ist, wird die Dokumentation dazu, falls vorhanden, immer im Fenster "Dokumentation" angezeigt. Sie können auch ein automatisches Layout für alle Elemente im Diagramm durchführen.

- Klicken Sie mit der rechten Maustaste auf das Element im Diagramm und wählen Sie im Kontextmenü den Befehl **Anzeigen | Anmerkende Kommentare**.

Dokumentations-Hyperlinks

Um im Fenster "Dokumentation" einen Hyperlink zu erstellen, wählen Sie Text im Fenster aus, klicken Sie mit der rechten Maustaste darauf und wählen Sie im Kontextmenü den Befehl **Hyperlinks einfügen/bearbeiten**. Beim Ziel des Hyperlink kann es sich um eine Webseite, ein Diagramm, eine Datei oder ein anderes Element handeln, siehe auch Erstellen von Hyperlinks zu Elementen 115.

 © 2016-2022 Altova GmbH



Fenster "Dokumentation"

Codegenerierung und Dokumentationskommentare

Wenn Sie anhand von Klassendiagrammen Code generieren, können alle auf Klassen und deren Mitglieder angewendete Kommentare (in Klassendiagrammen) ebenfalls in den generierten Code exportiert werden. Aktivieren Sie dazu vor der Generierung von Programmcode das Kontrollkästchen **Dokumentation als JavaDocs schreiben** (für Java) oder **Dokumentation als DocComments verfassen** (für C#, VB.NET), siehe auch Codegenerierungsoptionen 176.

Ebenso können Codekommentare auch beim Reverse Engineering von Programmcode zu einem Modell in das Modell importiert werden. Aktivieren Sie dazu vor dem Reverse Engineering das Kontrollkästchen **JavaDocs als Dokumentation** (für Java) oder **DocComments als Dokumentation** (für C#, VB.NET), siehe auch Optionen für den Code-Import 196.

Informationen zu den Entsprechungen der Kommentare in Programmcode (oder XML-Schemas) für UModel-Kommentare finden Sie in den Entsprechungstabellen zu den einzelnen Sprachen:

- C#-Entsprechungen 228
- VB.NET-Entsprechungen 248
- Java-Entsprechungen 263
- XML Schema-Entsprechungen 268

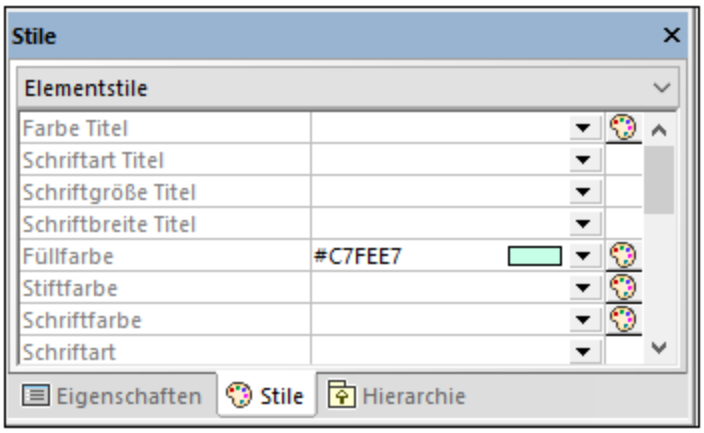
5.1.11 Ändern des Stils von Elementen

Sie können das Aussehen (den Stil) von Modellierungselementen, wie Farbe, Schriftgröße und -breite, Hintergrundfarbe, Liniendicke und andere Einstellungen ändern. Das Aussehen von Elementen kann auf verschiedenen Ebenen geändert werden: global für alle Elemente im Projekt, selektiv für alle Elemente derselben Familie (z.B. Klassen) oder für jedes einzelne Element. Informationen dazu, wie Sie den Stil des Diagramms selbst ändern können, finden Sie unter Ändern des Stils von Diagrammen 127.

Wenn Sie anstelle der konventionellen Elementdarstellung in Diagrammen Ihre eigenen Bilder verwenden möchten, können Sie Ihr Projekt um benutzerdefinierte Profile und Stereotypen erweitern. Nähere Informationen dazu finden Sie unter Beispiel: Anpassen von Symbolen und Stilen 432.

So ändern Sie das Aussehen von Elementen:

1. Klicken Sie auf das Element in einem Diagramm.
2. Am oberen Rand des Fensters "Stile" befindet sich eine Dropdown-Liste. Hier haben Sie folgende Auswahlmöglichkeiten:
   a. Um nur die Eigenschaften des aktuellen Elements zu bearbeiten, wählen Sie in der Liste den Eintrag "Elementstile".

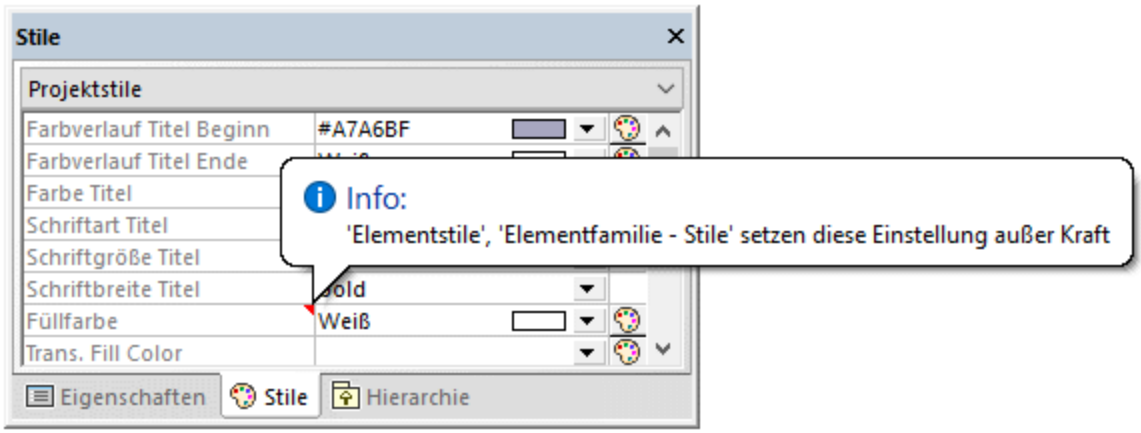


   b. Um die Eigenschaften aller Elemente derselben Art (z.B. Klassen) zu bearbeiten, wählen Sie in der Liste "Elementfamilie - Stile" aus.
   c. Um die Eigenschaften aller Elemente global auf Projektebene zu bearbeiten, wählen Sie "Projektstile".
   d. Um die Eigenschaften aller Linien im Projekt, einschließlich aller Assoziations-, Abhängigkeits- und Realisierungslinien zu bearbeiten, wählen Sie "Linienarten". (Dieser Wert ist nur sichtbar, wenn es sich beim aktuell ausgewählten Element um eine Linie handelt.)
   e. Um im gesamten Projekt die Eigenschaften aller Elemente, bei denen es sich nicht um Linien handelt (die so genannten Knoten), zu bearbeiten, wählen Sie "Knotenstile". (Dieser Wert ist nur sichtbar, wenn es sich beim aktuell ausgewählten Element nicht um eine Linie handelt.)
3. Ändern Sie den Wert der gewünschten Eigenschaft (z.B. "Füllfarbe").

Spezifischere Stile setzen generischere Stile außer Kraft, d.h. Stile, die auf einer Elementebene angewendet werden, setzen solche, die auf Elementfamilienebene angewendet werden, außer Kraft. Ebenso setzen auf Elementfamilienebene angewendete Stile auf Projektebene angewendete Stile außer Kraft.

Wenn ein Stil außer Kraft gesetzt wird, erscheint in der rechten oberen Ecke der außer Kraft gesetzten Eigenschaft ein kleines rotes Dreieck. Bewegen Sie den Cursor über das Dreieck, um einen Tooltipp mit Informationen über diese Stilpriorität zu sehen.

 © 2016-2022 Altova GmbH



Außer Kraft gesetzter Elementstil

5.2 Diagramme

5.2.1 Erstellen von Diagrammen

In Diagrammen wird visuell dargestellt, wie Modellierungselemente miteinander interagieren, welche Struktur, Abhängigkeiten, Hierarchie, usw. sie haben. Diagramme müssen zu einem Paket im Projekt gehören und daher im Fenster "Modell-Struktur" unterhalb von einem bestehenden Paket erstellt werden. Sie können Diagramme jederzeit durch Ziehen mit dem Maus von einem Paket in ein anderes verschieben.

So erstellen Sie ein neues Diagramm:

1. Klicken Sie im Fenster "Modell-Struktur" 80 mit der rechten Maustaste auf ein Paket.
2. Wählen Sie **Neues Diagramm | <Diagrammart>**.

Sie können ein neues Diagramm auch über das Fenster Diagramm-Struktur 84 erstellen. Gehen Sie dazu folgendermaßen vor:

1. Klicken Sie im Fenster "Diagramm-Struktur" mit der rechten Maustaste auf den Root-Node ("Diagramme").
2. Wählen Sie ein Paket für das Diagramm aush und klicken Sie auf **OK**.

Wenn das Diagrammfenster aktiv ist, werden in den Symbolleisten nur Modellierungselemente angezeigt, die auf die aktuelle Diagrammart angewendet werden können. Die Diagrammart wird im Fenster "Eigenschaften" angezeigt, wenn Sie in einen leeren Bereich des Diagramms klicken. Zusätzlich dazu wird die Art des Diagramms durch die folgenden Symbole gekennzeichnet.

Symbol	Beschreibung
	Aktivitätsdiagramm
	Klassendiagramm
	Kommunikationsdiagramm
	Komponentendiagramm
	Kompositionsstrukturdiagramm
	Deployment-Diagramm
	Interaktionsübersichtsdiagramm
	Objektdiagramm
	Paketdiagramm
	Profildiagramm
	Protokoll-Zustandsdiagramm

 © 2016-2022 Altova GmbH

Symbol	Beschreibung
	Sequenzdiagramm
	Zustandsdiagramm
	Zeitverlaufsdiagramm
	Use Case-Diagramm
	XML-Schema-Diagramm

5.2.2 Generieren von Diagrammen

Anstatt Diagramme von Grund auf neu zu erstellen, können Sie bestimmte Diagramme auch automatisch anhand vorhandener Modellierungselemente oder anhand von Programmcode erstellen. In diesem Kapitel wird erläutert, wie Sie Diagramme anhand von vorhandenen Modellierungselementen generieren. Informationen darüber, wie Sie Diagramme anhand von Quellcode generieren, finden Sie hier:

- Generieren von Klassendiagrammen 407
- Generieren von Sequenzdiagrammen anhand von Quellcode 371
- Generieren von Paketdiagrammen 418

Um Diagramme anhand von vorhandenen Elementen zu generieren, klicken Sie mit der rechten Maustaste auf ein Element (z.B. Paket) in der Modell-Struktur und wählen Sie anschließend im Kontextmenü die Option **In neuem Diagramm anzeigen | <Option>**. Unten finden Sie einige Beispiele:

So erstellen Sie ein Diagramm, in dem der Inhalt eines vorhandenen Pakets angezeigt wird:

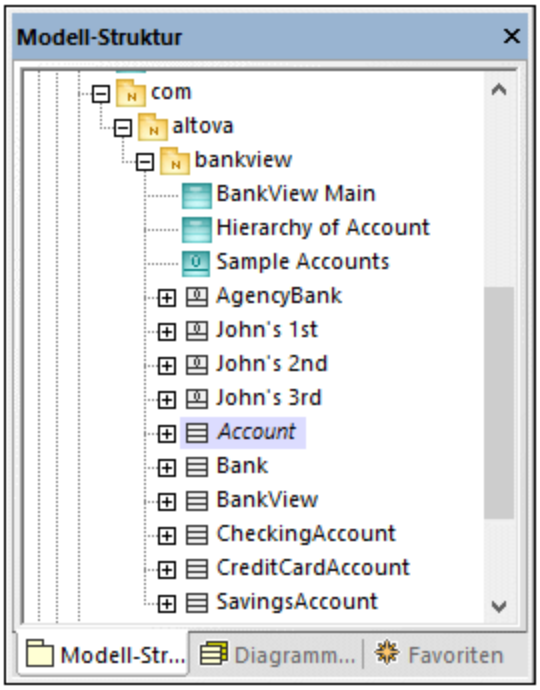
- Klicken Sie mit der rechten Maustaste im Fenster "Modell-Struktur" auf ein Paket und wählen Sie im Kontextmenü den Befehl **In neuem Diagramm anzeigen | Inhalt**.

So erstellen Sie ein Diagramm, in dem die Abhängigkeiten eines vorhandenen Pakets angezeigt werden:

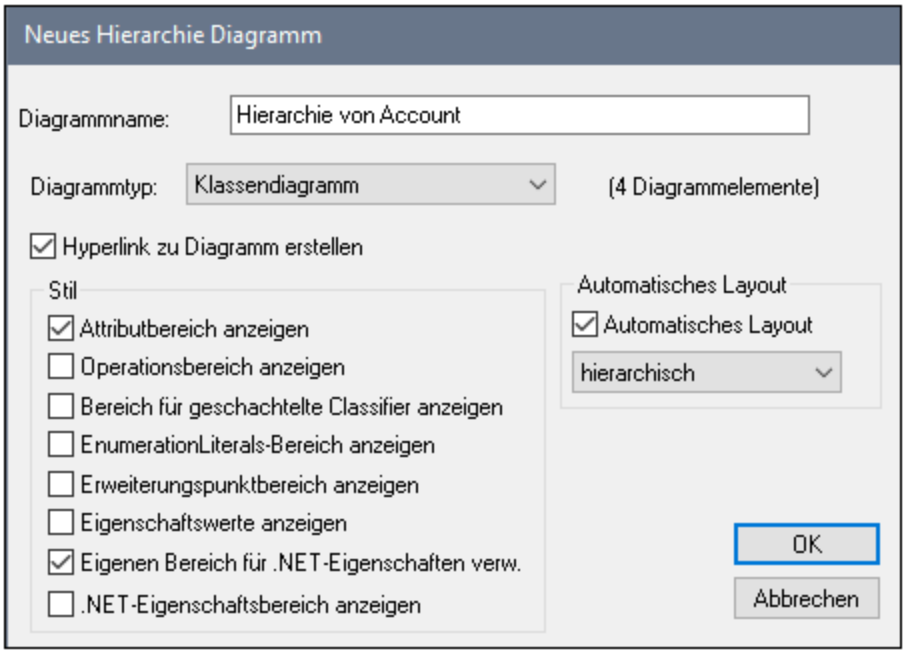
- Klicken Sie mit der rechten Maustaste im Fenster "Modell-Struktur" auf ein Paket und wählen Sie im Kontextmenü den Befehl **In neuem Diagramm anzeigen | Paketabhängigkeiten**.

So erstellen Sie ein Diagramm, in dem die Generalisierungshierarchie einer Klasse angezeigt wird:

1. Klicken Sie mit der rechten Maustaste im Fenster "Modell-Struktur" auf eine Klasse, die eine Generalisierungsbeziehung zu oder von anderen Klassen aufweist (z.B. auf die Klasse `Account` aus dem Beispielprojekt **C:\Benutzer\<Benutzername>\Dokumente\Altova\UModel2023\UModelExamples\Bank_CSharp.ump**).

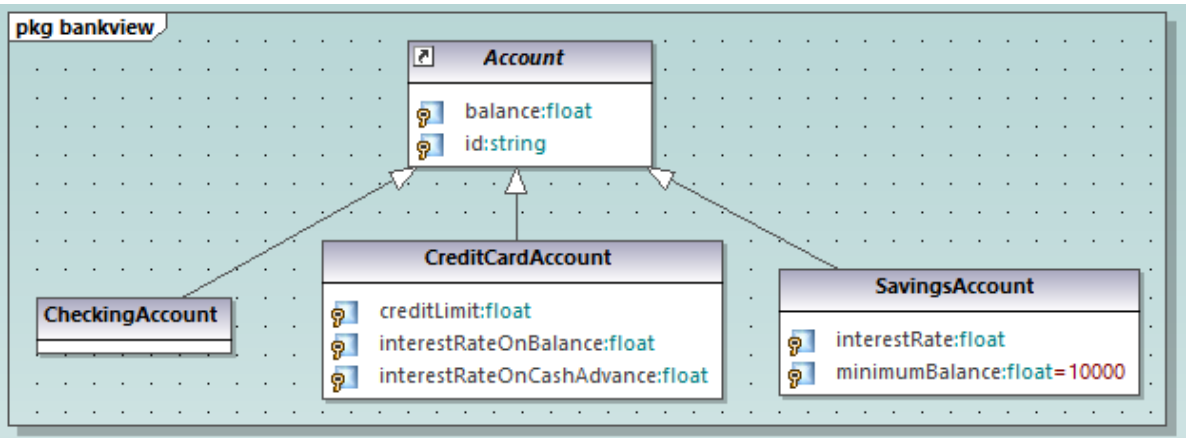


2. Wählen Sie im Kontextmenü die Option **In neuem Diagramm anzeigen | Generalisierungshierarchie**. Daraufhin wird ein Dialogfeld angezeigt, in dem Sie die Einstellungen für das zu erstellende Diagramm, darunter auch den Diagrammtyp anpassen können. Beachten Sie den Text "N Diagrammelemente", der die Anzahl der Elemente angibt, die zum Diagramm hinzugefügt werden sollen. Im Beispiel unten wird als Diagrammtyp "Klassendiagramm" ausgewählt. Das Diagramm enthält vier Diagrammelemente (Klassen): die Klasse `Account` und die davon abgeleiteten Klassen.



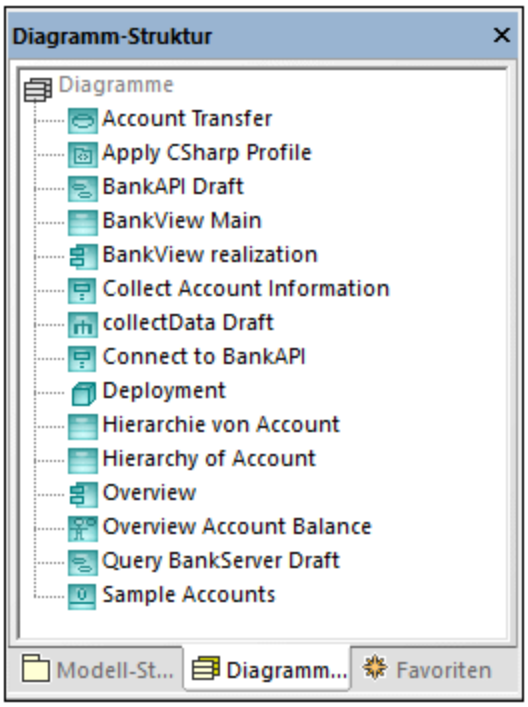
 © 2016-2022 Altova GmbH

3. Klicken Sie auf **OK**. Das anhand der ausgewählten Optionen generierte Diagramm wird im Diagrammfenster geöffnet, z.B.:



5.2.3 Öffnen von Diagrammen

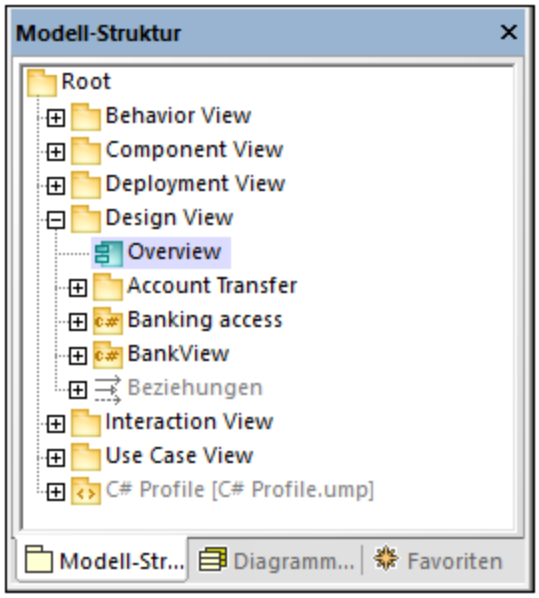
Wenn das UModel-Projekt Diagramme enthält, werden diese im Fenster "Diagramm-Struktur" angezeigt.



Fenster "Diagramm-Struktur"

Anmerkung: BStandardmäßig werden Diagramm im Fenster "Diagramm-Struktur" nach Typ gruppiert. Um nur Diagramme (ohne die übergeordnete Struktur) zu sehen, klicken Sie mit der rechten Maustaste in das Fenster und deaktivieren Sie im Kontextmenü die Option **Nach Diagrammtyp gruppieren**.

Die Diagramme werden auch im Fenster "Modell-Struktur" unterhalb der Pakete, zu denen sie gehören, angezeigt., z.B.:



So öffnen Sie ein vorhandenes Diagramm:

- Doppelklicken Sie im Fenster "Modell-Struktur" (oder im Fenster "Diagramm-Struktur" oder im Fenster "Favoriten") auf das Diagrammsymbol.
- Klicken Sie mit der rechten Maustaste auf das Diagramm und wählen Sie im Kontextmenü den Befehl **Diagramm öffnen**.

5.2.4 Löschen von Diagrammen

UModel-Diagramme können auf die folgenden Arten gelöscht werden:

- Klicken Sie im Fenster "Modell-Struktur" (oder im Fenster "Diagramm-Struktur" oder im Fenster "Favoriten") mit der rechten Maustaste auf das Diagramm und wählen Sie im Kontextmenü den Befehl **Löschen**.
- Klicken Sie in einem der oben erwähnten Fenster auf das Diagramm und drücken Sie die **Entf**-Taste.

Durch das Löschen eines Diagramms wird nur das Diagramm selbst aus dem Projekt gelöscht, nicht aber die Elemente. Um zu überprüfen, ob Elemente in einem Diagramm verwendet werden, klicken Sie mit der rechten Maustaste auf das gewünschte Paket und wählen Sie den Befehl **In keinem Diagramm verwendete Elemente auflisten**, siehe auch Überprüfen, wo und ob Elemente verwendet werden 113.

Wie Sie Elemente aus einem Diagramm oder Projekt löschen können, finden Sie unter Löschen von Elementen 110.

 © 2016-2022 Altova GmbH

5.2.5 Ändern des Stils von Diagrammen

Sie können das Aussehen (den Stil) von Diagrammen, wie Hintergrundfarbe, Rasterfarbe, Rastergröße und Farbe sowie das Aussehen der Diagrammüberschrift ändern. Dabei können Sie entweder den Stil einzelner Diagramme im Projekt ändern oder dieselben Eigenschaften auf alle Diagramme im Projekt anwenden. Informationen darüber, wie Sie den Stil von Elemente in einem Diagramm ändern, finden Sie unter Ändern des Stils von Elementen 119.

Die Größe von Diagrammen ist durch die Elemente und ihre Anordnung definiert. Um die Größe eines Diagramms anzupassen, ziehen Sie ein Element an einen der Diagrammränder und die Größe wird automatisch angepasst.

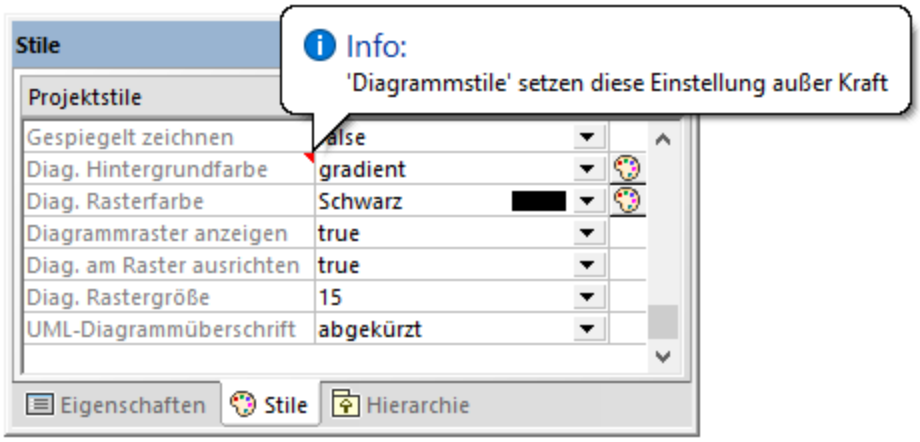
So ändern Sie das Aussehen von Diagrammen:

1. Öffnen Sie ein Diagramm (siehe Öffnen von Diagrammen 125).
2. Beachten Sie die Dropdown-Liste am oberen Rand des Fensters "Stile" und wählen Sie nach Bedarf eine der folgenden Methoden:
   a. Um nur die Eigenschaften des aktuellen Diagramms zu bearbeiten, wählen Sie in der Liste den Eintrag "Diagrammstile". Dieser Wert wird standardmäßig ausgewählt, wenn Sie auf den leeren Hintergrund des Diagramms klicken (d.h., wenn kein Diagrammelement ausgewählt ist).

   b. Um Änderungen auf alle Diagramme im Projekt anzuwenden, wählen Sie den Eintrag "Projektstile" aus. Scrollen Sie in diesem Fall bis zum Ende des Fensters "Stile", bis Sie die zu den Stilen gelangen, die auf Diagramme angewendet werden (d.h. zu den Stilen, die mit "Diag." beginnen).
3. Ändern Sie den Wert der gewünschten Eigenschaft (z.B. "Diag. Hintergrundfarbe").

> Stile, die auf Diagrammebene angewendet wird, setzen diejenigen, die auf Projektebene angewendet werden, außer Kraft.

Wenn ein Stil außer Kraft gesetzt wird, erscheint in der rechten oberen Ecke der außer Kraft gesetzten Eigenschaft ein kleines rotes Dreieck. Bewegen Sie den Cursor über das Dreieck, um einen Tooltipp mit Informationen über die Stilpriorität zu sehen



Außer Kraft gesetzter Diagrammstil

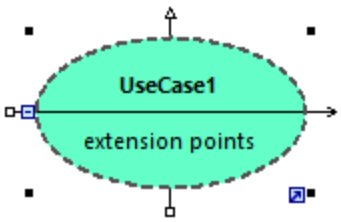
Die folgenden diagrammspezifischen Eigenschaften stehen in Form von Symbolleisten-Schaltflächen zur Verfügung. Wenn Sie die Eigenschaft im Fenster "Stile" ändern, wird der Status der Symbolleisten-Schaltfläche aktualisiert und umgekehrt.

	Raster anzeigen	Blendet das Diagrammraster ein bzw. aus.
	Diagrammüberschrift anzeigen	Blendet die Diagrammüberschrift ein bzw. aus.
	Am Raster ausrichten	Wenn diese Eigenschaft aktiviert ist, werden alle Elemente am Raster ausgerichtet. Bei deaktivierter Eigenschaft werden die Elemente beliebig unabhängig vom Diagrammraster positioniert.

5.2.6 Ausrichten von Modellierungselementen und Anpassen der Größe

Sie können die Größe von Elementen im Diagramm folgendermaßen anpassen:

1. Klicken Sie auf ein Element im Diagramm. Daraufhin werden rund um das Diagramm herum mehrere schwarze Punkte angezeigt..

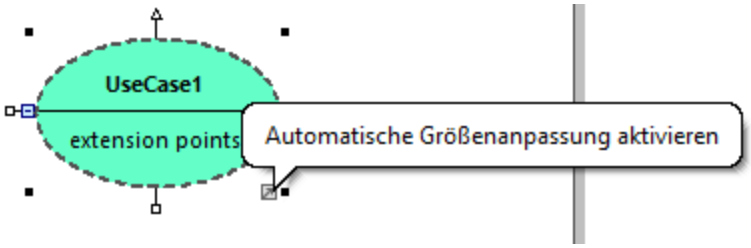


2. Ziehen Sie einen der schwarzen Punkte in die Richtung, in die das Diagramm vergrößert werden soll.

Um die Elementgröße wieder auf die Originalgröße zurückzusetzen, wählen Sie eine der folgenden Methoden:

 © 2016-2022 Altova GmbH

- Klicken Sie an der rechten unteren Ecke des Elements auf das Symbol **Automatische Größenanpassung aktivieren**.



- Klicken Sie mit der rechten Maustaste auf ein Element im Diagramm und wählen Sie im Kontextmenü den Befehl **Größe automatisch anpassen.**.
- Wählen Sie ein oder mehrere Elemente aus. Klicken Sie im Menü **Layout** auf **Größe automatisch anpassen**.

Wenn in Diagramm mindestens zwei Modellierungselemente ausgewählt wurden, können diese aneinander ausgerichtet werden (so können z.B. beide waagrecht oder senkrecht aneinander ausgerichtet werden, oder in der Größe angeglichen werden). Die Befehle zum Ausrichten oder Anpassen der Größe von Elementen finden Sie im Menü **Layout** sowie in der gleichnamigen Symbolleiste.

Layout-Symbolleiste

Wenn Sie mehrere Elemente markieren, wird für die Aktion, die Sie daran ausführen (Ausrichten oder Größe anpassen), als Vorlage das **zuletzt** ausgewählte Element verwendet. Wenn Sie z.B. drei Klassenelemente auswählen und den Befehl **Breite angleichen** aufrufen, dann werden alle drei Elemente in der Breite an die zuletzt ausgewählte Klasse angepasst. Das zuletzt ausgewählte Element wird immer mit einer strichlierten Umrandung angezeigt.

In der folgenden Tabelle finden Sie die Befehle für die Ausrichtung von Elementen und die Anpassung ihrer Größe:

Symbol	Befehl	Anmerkungen
	Linksbündig	
	Rechtsbündig	
	Bündig oben	
	Bündig unten	
	Vertikal zentrieren	
	Horizontal zentrieren	
	Waagrecht verteilen	Dieser Befehl steht zur Verfügung, wenn drei oder mehr Elemente ausgewählt sind. Die waagrechten Abstände zwischen den

Symbol	Befehl	Anmerkungen
		ausgewählten Elementen werden dadurch gleich groß gemacht.
	Senkrecht verteilen	Dieser Befehl steht zur Verfügung, wenn drei oder mehr Elemente ausgewählt sind. Die senkrechten Abstände zwischen den ausgewählten Elementen werden dadurch gleich groß gemacht.
	Horizontal anordnen	Mit diesem Befehl werden alle ausgewählten Elemente im Diagramm neu positioniert, so dass sie waagrecht nebeneinander angeordnet werden.
	Vertikal anordnen	Mit diesem Befehl werden alle ausgewählten Elemente im Diagramm neu positioniert, so dass sie senkrecht übereinander angeordnet werden.
	Breite angleichen	
	Höhe angleichen	
	Größe angleichen	

Sie können auch ein automatisches Layout für alle Elemente im Diagramm durchführen. Gehen Sie dazu folgendermaßen vor:

- Klicken Sie im Menü **Layout** auf **Automatisches Layout** und wählen Sie eine der folgenden Optionen aus: **Zentriert**, **Hierarchisch** oder **Block**.

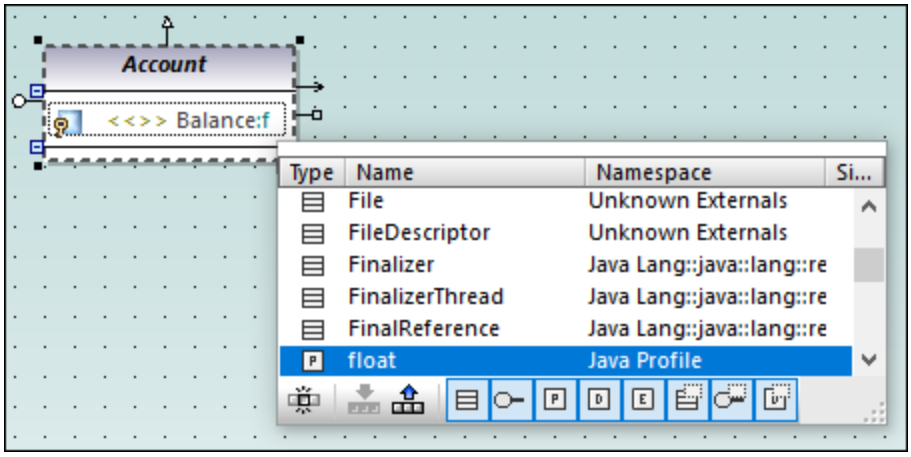
Zentriert	Zeigt die Modellierungselemente in der Mitte zentriert an.
Hierarchisch	Zeigt die Elemente nach ihren hierarchischen Beziehungen an. So wird z.B. eine übergeordnete Klasse oberhalb der davon abgeleiteten Klassen angezeigt. Die Optionen für hierarchisches Layout können über das Menü **Extras | Optionen,** Register **Ansicht,** Gruppe **Hierarchisch anordnen** angepasst werden.
Block	Zeigt die Elemente rechteckig nach Elementgröße gruppiert an.

5.2.7 Typ-Autokomplettierung in Klassen

Die Autokomplettierung von Datentypen ist in UModel standardmäßig beim Hinzufügen von Operationen und Attributen zu einer Klasse aktiviert. Auf diese Art können Sie den Datentyp der Operation oder Eigenschaft direkt im Diagramm definieren, z.B.:

1. Klicken Sie mit der rechten Maustaste auf eine Klasse und wählen Sie im Kontextmenü den Befehl **Neu | Operation**.
2. Geben Sie nach den doppelten spitzen Klammern << >> den Namen der Operation, gefolgt von einem Doppelpunkt (:) ein.
3. Daraufhin wird ein Autokomplettierungsfenster geöffnet.

 © 2016-2022 Altova GmbH



Autokomplettierungsfenster

Das Autokomplettierungsfenster bietet die folgenden Funktionalitäten:

- Durch Klick auf einen Spaltennamen wird das Fenster in auf- oder absteigender Reihenfolge nach diesem Attribut sortiert.
- Durch Ziehen der rechten unteren Ecke können Sie die Größe des Fensters anpassen.
- Der Inhalt des Fensters kann durch Auswahl der entsprechenden Filter (Kategorien) am unteren Rand des Fensters gefiltert werden: Klasse, Schnittstelle, Primitivtyp, Datentyp, Enumeration, Klassen-Vorlage, Schnittstellen-Vorlage, Datentyp-Vorlage.

So aktivieren Sie nur einen der Filter auf einmal:

- Klicken Sie auf die Schaltfläche **Einzelmodus** . In der obigen Abbildung sehen Sie das Autokomplettierungsfenster im "Mehrfachmodus", d.h. alle Filter sind aktiviert. Die Schaltfläche "Einzelmodus" ist deaktiviert.

So aktivieren oder deaktivieren Sie alle Filter auf einmal:

- Klicken Sie auf **Alle Kategorien definieren** bzw. **Alle Kategorien löschen** .

So deaktivieren Sie die Autokomplettierung:

1. Klicken Sie im Menü **Extras** auf **Optionen** und anschließend auf das Register **Diagrammbearbeitung**.
2. Deaktivieren Sie das Kontrollkästchen **Automatische Eingabehilfe aktivieren.**

So rufen Sie die Autokomplettierung bei Bedarf auf (wenn sie deaktiviert ist):

1. Stellen Sie sicher, dass sich der Cursor innerhalb von einem Attribut, einer Operation oder Klasse hinter dem Doppelpunkt (:) befindet.
2. Drücken Sie **Strg+Leerzeichen**.

5.2.8 Vergrößern und Verkleinern von Diagrammen

Um ein Diagramm auf dem Bildschirm zu vergrößern bzw. zu verkleinern, wählen Sie eine der folgenden Methoden:

- Rufen Sie den Menübefehl **Ansicht | Vergrößern (Strg+Umschalt+I)** oder **Ansicht | Verkleinern (Strg+Umschalt+O)** auf.
- Wählen Sie in der Symbolleiste Vergrößern/Verkleinern einen vordefinierten Prozentwert aus.

- Halten Sie die Strg-Taste gedrückt, während Sie das Mausrad drehen.

So passen Sie den Diagrammbereich an die Größe des sichtbaren Fensters an:

- Rufen Sie den Menübefehl **Ansicht | An Fenstergröße anpassen** auf (oder klicken Sie auf die Symbolleisten-Schaltfläche **An Fenstergröße anpassen**).

© 2016-2022 Altova GmbH

5.3 Beziehungen

5.3.1 Erstellen von Beziehungen

Für eine Beziehung werden normalerweise zwei Elemente benötigt, daher muss Ihr Diagramm die Elemente, zwischen denen Sie Beziehungen hinzufügen möchten, bereits enthalten. Beziehungen können folgendermaßen erstellt werden:

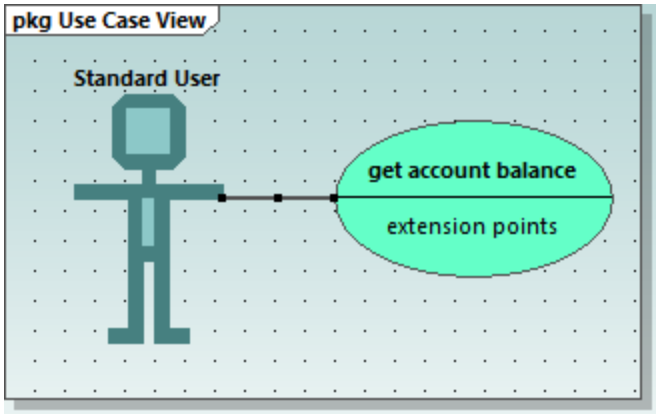
1. Über eine Symbolleisten-Schaltfläche für die gewünschte Beziehung (z.B. Assoziation).
2. Über die Ziehpunkte, die angezeigt werden, wenn Sie auf ein Element im Diagramm klicken.

Erstellen von Beziehungen mit Hilfe von Symbolleisten-Schaltflächen

Wenn im Hauptfenster von UModel ein Diagrammfenster aktiv (im Fokus) ist, werden in der Symbolleiste alle von diesem Diagramm unterstützten Elemente und Beziehungen angezeigt. So bietet etwa ein Klassendiagramm Symbolleisten-Schaltflächen für alle unterstützten Beziehungen. Dazu gehören etwa Assoziation , Collection-Assoziation , Aggregation , Komposition , Realisierung , Generalisierung und andere. Ein Use Case-Diagramm bietet z.B. Symbolleisten-Schaltflächen für Assoziationen , Generalisierungen sowie Include- und Extend- Beziehungen.

In der Anleitung unten wird beschrieben, wie Sie eine Assoziationsbeziehung zwischen einem Akteur und einem Use Case (Anwendungsfall) erstellen. Auf dieselbe Art können Sie bei Bedarf auch andere Beziehungen erstellen.

1. Klicken Sie auf ein Element im Diagramm (Akteur "Standardbenutzer" in der Abbildung unten).
2. Klicken Sie auf die Symbolleisten-Schaltfläche für die gewünschte Beziehung (in diesem Fall Assoziation).
3. Platzieren Sie die Maus über "Standardbenutzer" und ziehen Sie sie auf ein Zielelement (den Use Case "get account balance"). Beachten Sie, dass das Zielelement grün markiert wird und die Beziehung nur akzeptiert, wenn diese gemäß der UML-Spezifikation sinnvoll ist.



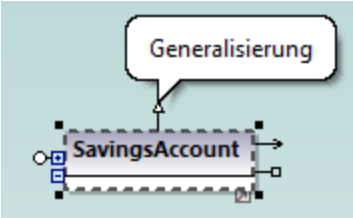
Assoziation in einem Use Case-Diagramm

Erstellen von Beziehungen mit Hilfe von Ziehpunkten

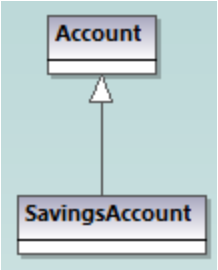
Wenn Sie auf ein Element in einem Diagramm klicken, werden eventuell links, rechts, oberhalb und unterhalb des Elements mehrere Ziehpunkte angezeigt. Die Ziehpunkte werden nur bei Elementen angezeigt, die Beziehungen unterstützen. Jeder Ziehpunkt entspricht einer Beziehungsart. Klassenelemente haben z.B. die folgenden Ziehpunkte:

- Schnittstellenrealisierung
- Generalisierung
- Assoziation
- Collection-Assoziation

Um zu sehen, für welche Art von Beziehung ein Ziehpunkt verwendet wird, platzieren Sie den Cursor über den Ziehpunkt. In der Abbildung oben dient der ausgewählte obere Ziehpunkt z.B. zum Erstellen einer Generalisierungsbeziehung.



Um die Beziehung zu erstellen, klicken Sie auf den Ziehpunkt und ziehen Sie den Cursor über das gewünschte Elemente. Dadurch wird die entsprechende Beziehung (in diesem Fall eine Generalisierung) erstellt.



Generalisierungsbeziehung zwischen zwei Klassen

5.3.2 Ändern des Stils von Linien und Beziehungen

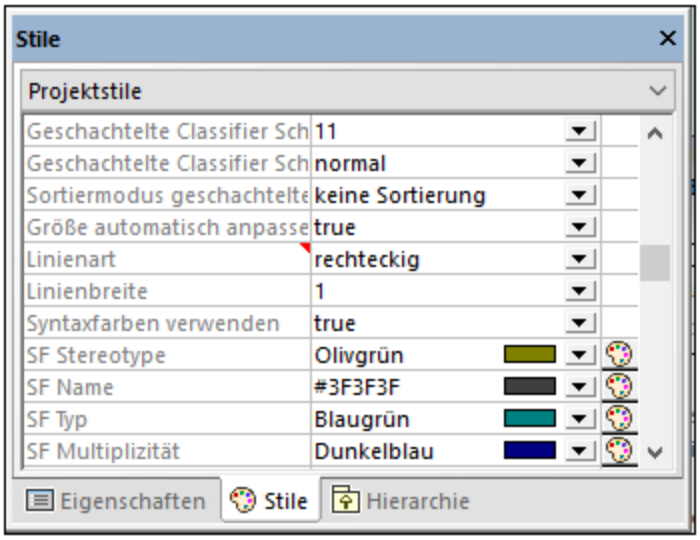
Sie können die Dicke, Farbe und den Krümmungsstil von Linien über das Fenster "Stile" ändern. Außerdem können Sie Text (Beschriftungen) zu Beziehungen hinzufügen, Beschriftungen neu positionieren und diese im Diagramm entweder für jede Beziehung einzeln oder alle gemeinsam ein- und ausblenden.

Anmerkung: Achten Sie in der Anleitung unten auf den Unterschied zwischen "Linien" (jede Linie im Diagramm) und "Beziehungen" wie Assoziation, Generalisierung, Komposition, usw. Alle Beziehungen sind Linien, dies gilt aber nicht für den umgekehrten Fall. So ist ein Kommentar oder eine Anmerkung nur eine Linie, nicht aber eine Beziehung.

 © 2016-2022 Altova GmbH

So ändern Sie Linieneigenschaften:

1. Klicken Sie im Diagramm auf eine Linie.
2. Definieren Sie im Fenster "Stile" die gewünschte Eigenschaft (z.B. "Linienbreite").



Die für die Eigenschaft "Linienart" verfügbaren Werte stehen auch als Befehle im Menü **Layout | Linienart** und als Symbolleisten-Schaltflächen zur Verfügung. Wenn Sie diese Eigenschaft ändern, wird die entsprechende Symbolleiste aktiv und umgekehrt.

	Orthogonale Linie	Eine Linie, die nur im rechten Winkel abgeknickt wird.
	Gerade Linie	Eine Linie dieser Art bildet eine gerade Verbindung ohne Wegpunkte zwischen zwei Elementen.
	Benutzerdefinierte Linie	Eine Linie dieser Art kann in jedem Winkel gekrümmt werden. Um die Linie zu verschieben, ziehen Sie einen beliebigen Wegpunkt (kleine schwarze Punkte) der Linie. Um neue Wegpunkte zu erstellen, klicken Sie an eine Stelle zwischen zwei Wegpunkten und ziehen Sie die Linie. Um Wegpunkte zu löschen, ziehen Sie einen Wegpunkt direkt auf einen vorhandenen.

Linienarten können wie andere Elementstile für jede einzelne Linie oder auf einer allgemeineren Ebene (z.B. auf Projektebene) definiert werden. Der spezifischere Stil setzt den allgemeineren außer Kraft. Wenn ein Stil außer Kraft gesetzt wurde, wird dies durch ein rotes Dreieck neben der betreffenden Eigenschaft im Fenster "Stile" angezeigt, siehe auch Ändern des Stils von Elementen 119.

So fügen Sie Beschriftungstext zu einer Beziehung hinzu:

- Klicken Sie auf eine Beziehung im Diagramm und geben Sie Text ein.

So verschieben Sie Beschriftungstext:

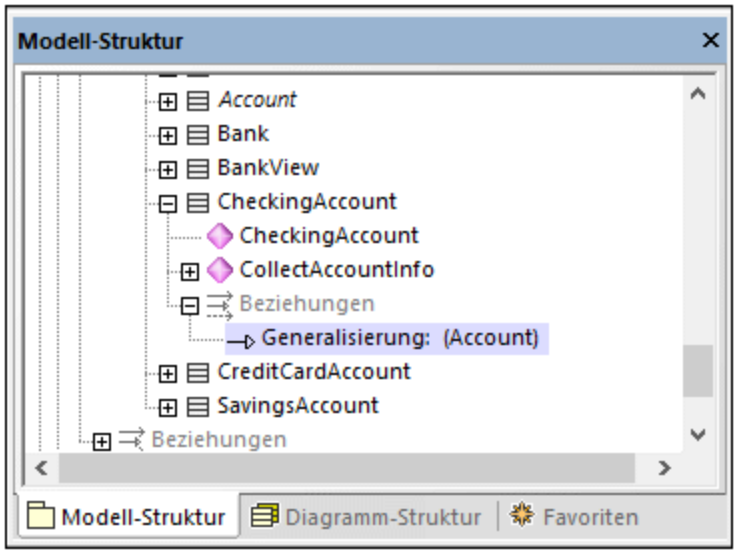
- Klicken Sie auf die Beschriftung und ziehen Sie sie an eine andere Stelle im Diagramm.
- So verschieben Sie die Beschriftung wieder an die Standardposition: Klicken Sie mit der rechten Maustaste auf die Beziehung und wählen Sie im Kontextmenü den Befehl **Textlabels | Textlabels neu positionieren.**
- So positionieren Sie mehrere Beschriftungen gleichzeitig neu: Wählen Sie eine oder mehrere Beziehungen im Diagramm aus und starten Sie den Menübefehl **Layout | Textlabels neu positionieren.**

So blenden Sie den Beschriftungstext ein oder aus:

- Klicken Sie mit der rechten Maustaste auf die Beziehung und wählen Sie im Kontextmenü den Befehl **Textlabels | Alle Textlabels anzeigen/ausblenden**.

5.3.3 Anzeigen von Elementbeziehungen

Standardmäßig werden die Beziehungen eines Elements in der Modell-Struktur unter dem jeweiligen Element angezeigt. So hat etwa die unten gezeigte Klasse `CheckingAccount` eine Generalisierungsbeziehung zur Klasse `Account`:



Beziehung im Fenster "Modell-Struktur"

Anmerkung: Um Beziehungen aus dem Fenster "Modell-Struktur" auszublenden, klicken Sie mit der rechten Maustaste in das Fenster und deaktivieren Sie die Option **Beziehungen in Struktur anzeigen.**

Um die Beziehungen eines Elements im Diagramm anzuzeigen, klicken Sie mit der rechten Maustaste auf das Element im Diagramm und wählen Sie im Kontextmenü die Option **Anzeigen | <Beziehungsart>**.

 © 2016-2022 Altova GmbH

5.3.4 Assoziationen

Eine Assoziation ist eine begriffliche Verbindung zwischen zwei Elementen. Assoziationsbeziehungen werden wie alle anderen Beziehungen in UModel erstellt, siehe Erstellen von Beziehungen 133.

Wenn Sie eine Assoziation zwischen zwei Klassen erstellen, wird in die Ursprungsklasse automatisch ein neues Attribut eingefügt. Wenn Sie z.B. eine Assoziation zwischen den Klassen `Car` und `Engine` erstellen, wird eine Eigenschaft vom Typ `Engine` zur Klasse `Car` hinzugefügt.

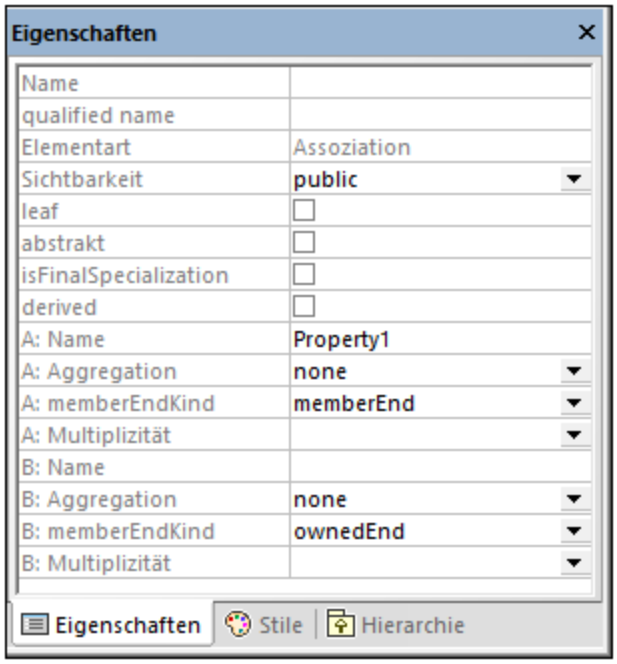
Wenn eine Klasse zu einem Diagramm hinzugefügt wird, werden ihre Assoziationen im Diagramm automatisch angezeigt, vorausgesetzt, die folgenden Bedingungen treffen zu:

- Die Option **Assoziationen automatisch erstellen** wurde unter **Extras | Optionen |** Register **Diagrammbearbeitung** aktiviert.
- Der Typ des Attributs wurde definiert (im Bild oben hat `Property1` den Typ `Engine`)
- Die Klasse des referenzierten "Typs" ist auch im aktuellen Diagramm vorhanden (in der Abbildung oben die Klasse `Engine`).

Sie können die Klasseneigenschaften jeder Klasse explizit als Assoziationen im Diagramm anzeigen. Klicken Sie dazu mit der rechten Maustaste auf eine Klasseneigenschaft und wählen Sie einen der folgenden Befehle:

- **Anzeigen | <Eigenschaft> als Assoziation**
- **Anzeigen | Alle Eigenschaften als Assoziationen**

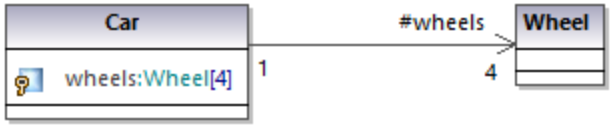
Wenn Sie im Diagramm auf eine Assoziation klicken, können ihre Eigenschaften bei Bedarf über das Fenster "Eigenschaften" geändert werden.



Beachten Sie besonders die unten aufgelisteten Eigenschaften. Wenn Sie diese Eigenschaften ändern, ändert sich das Aussehen der Assoziation im Diagramm oder es werden verschiedene informative Textbeschriftungen hinzugefügt. Informationen zum Anzeigen oder Ausblenden von Textbeschriftungen oder zum Ändern des Aussehens der Beziehung (z.B. Farben oder Linienbreite) finden Sie unter Ändern des Stils von Linien und Beziehungen 134.

Eigenschaft	Zweck
A: Name	Der Name des Mitglieds am Ende A der Beziehung. Im Car-Beispiel oben ist es `Property1`.
A: Aggregation	Damit können Sie den Typ der Assoziation am Ende A ändern. Wenn Sie diese Eigenschaft ändern, wird auch die Darstellung der Assoziation im Diagramm geändert. Gültige Werte: **none** Kennzeichnet eine normale Assoziation **shared** Ändert die Assoziation in eine Aggregation **composite** Ändert die Assoziation in eine Komposition
A: memberEndKind	Attribute, die an einer Beziehung beteiligt sind, können entweder zu einer Klasse oder zur Assoziation gehören. Diese Eigenschaft definiert, wer der Inhaber dieses Endes der Beziehung ist und ob dieses Ende der Beziehung "navigierbar" ist. (Mit "navigierbar" ist gemeint, dass das Ende einen "Pfeil" aufweist). Gültige Werte: **memberEnd** Das Mitglied an diesem Ende gehört zur Klasse.

 © 2016-2022 Altova GmbH

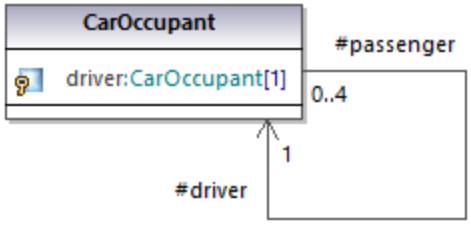
Eigenschaft	Zweck
	ownedEnd Das Mitglied an diesem Ende gehört zur Assoziation **navigableOwnedEnd** Das Mitglied an diesem Ende gehört zur Assoziation und diese Ende wird navigierbar. Wenn Sie sowohl das Ende A als auch das Ende B auf **ownedEnd** setzen, wird die Assoziation bidirektional.
A: Multiplizität	Multiplizität definiert die Anzahl der Objekte an diesem Ende der Beziehung. Wenn ein Auto z.B. vier Räder hat, wäre die Multiplizität an einem Ende der Beziehung 1 und am anderen Ende 4. 

Dieselbe Gruppe von Attributen steht auch für das Ende B der Beziehung zur Verfügung.

Wenn im Fenster "Stile" die Eigenschaft **Show Assoc. Ownership Punkt anzeigen** aktiviert ist, werden Ownership-Punkte für die ausgewählte Beziehung angezeigt. Standardmäßig ist diese Eigenschaft auf **False** gesetzt. Im Folgenden sehen Sie ein Beispiel für eine Klasse, für die **Show Assoc. Ownership Punkt anzeigen** auf **True** gesetzt ist:

Erstellen reflexiver Assoziationen

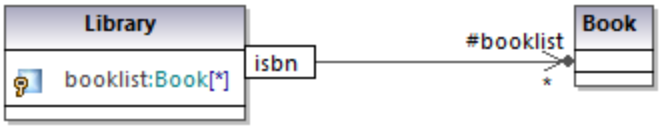
Assoziationen können mit derselben Klasse als Quelle und Ziel erstellt werden. Dies ist ein sogenanntes "self link" oder eine reflexive Assoziation. Damit kann z.B. die Fähigkeit eines Objekts, zum Zweck rekursiver Aufrufe eine Nachricht an sich selbst zu senden, beschrieben werden. Um ein self-link zu erstellen, klicken Sie auf die Symbolleisten-Schaltfläche "Assoziation" und ziehen Sie diese dann vom Element an eine andere Stelle im selben Element.



Erstellen von Assoziations-Qualifiern

Assoziationen können optional mit Assoziations-Qualifiern versehen werden. Qualifier sind Attribute einer Assoziation. Im Beispiel unten definiert der Assoziations-Qualifier `isbn`, dass ein Buch anhand dieses Attributs aus der Liste der Bücher abgerufen werden kann. So fügen Sie einen Qualifier hinzu:

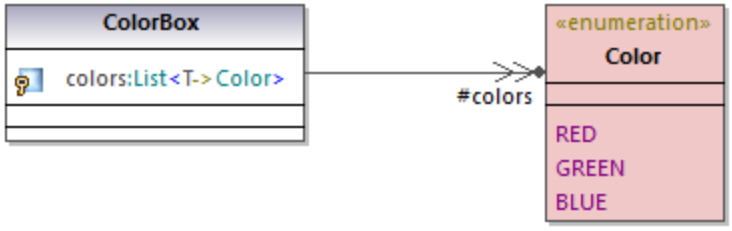
1. Erstellen Sie eine Assoziation zwischen zwei Klassen.
2. Klicken Sie mit der rechten Maustaste auf die Assoziation und wählen Sie **Neu | Qualifier**.



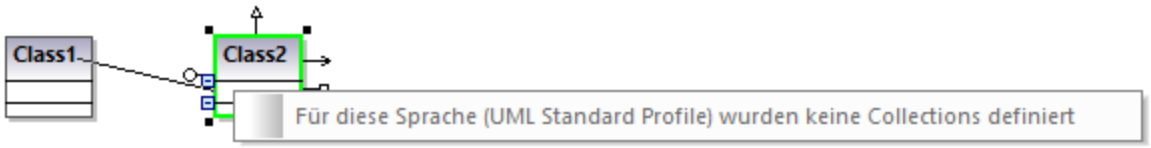
Um Assoziations-Qualifier umzubenennen oder zu löschen, gehen Sie auf dieselbe Weise vor wie bei anderen Elementen, siehe Umbenennen, Verschieben und Kopieren von Elementen 109 und Löschen von Elementen 110.

5.3.5 Collection-Assoziationen

Eine Collection-Assoziationsbeziehung eignet sich, um anzuzeigen, dass eine Klasseneigenschaft eine Collection irgendeiner Art ist. So ist etwa die Eigenschaft `colors` der Klasse `ColorBox` im Diagramm unten eine Liste von Farben. Dieser Typ ist in diesem Fall als Enumeration definiert; es kann sich dabei jedoch auch um eine andere Klasse oder sogar eine Schnittstelle handeln.



Bevor Sie Collection-Assoziationen erstellen, muss das UModel-Projekt die Collection-Vorlagen für die gewünschte Projektsprache (wie z.B. Java, C# oder VB.NET) enthalten. Andernfalls wird ein Tooltipp mit dem Text "Für diese Sprache wurden keine Collections definiert" angezeigt, wenn Sie versuchen, die Collection-Assoziation zu erstellen.



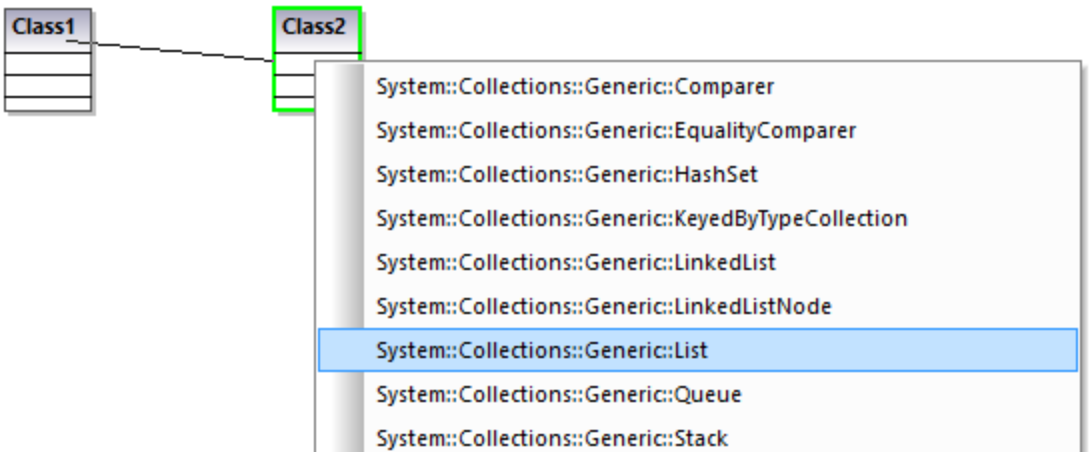
Wenn es sich bei Ihrem Projekt um ein reines UML-Projekt handelt (ohne Unterstützung für eine bestimmte Code Engineering-Sprache), können Sie über das Menü **Extras | Optionen | Diagrammbearbeitung | Collection-Vorlagen |** Register **UML** Collection-Vorlagen definieren.

 © 2016-2022 Altova GmbH

Wenn Ihr Projekt bereits einen Sprach-Namespace (wie z.B. Java, C#, VB.NET) enthält, sind die Collection-Vorlagen über das Profil dieser Sprache vordefiniert. Sie können über das Menü **Extras | Optionen | Diagrammbearbeitung | Collection-Vorlagen** zusätzliche Vorlagen hinzufügen.

So erstellen Sie (z.B. zwischen zwei Klassen) eine Collection-Assoziation:

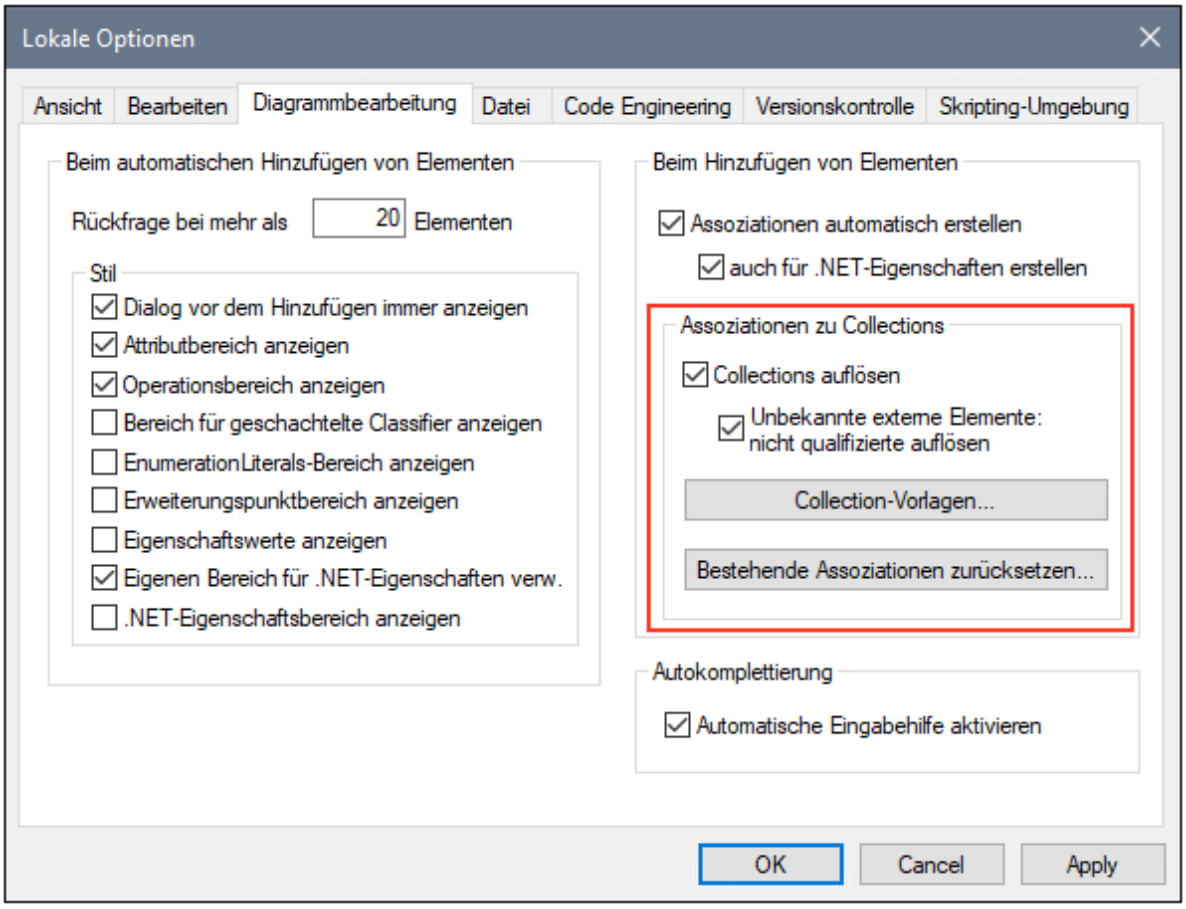
1. Fügen Sie zwei Klassen zum Diagramm hinzu.
2. Klicken Sie auf die Symbolleisten-Schaltfläche **Collection-Assoziation** .
3. Ziehen Sie die Maus von der ersten Klasse auf die zweite Klasse. Daraufhin werden im Kontextmenü die für das Projekt definierten Collection-Vorlagen angezeigt und Sie können die gewünschte auswählen.



Collection-Assoziationen und Code Engineering

Wenn Sie Programmcode in das Modell importieren, werden standardmäßig automatisch Collection-Assoziationen auf Basis der vordefinierten Collection-Vorlagen erstellt. So aktivieren bzw. deaktivieren Sie diese Option:

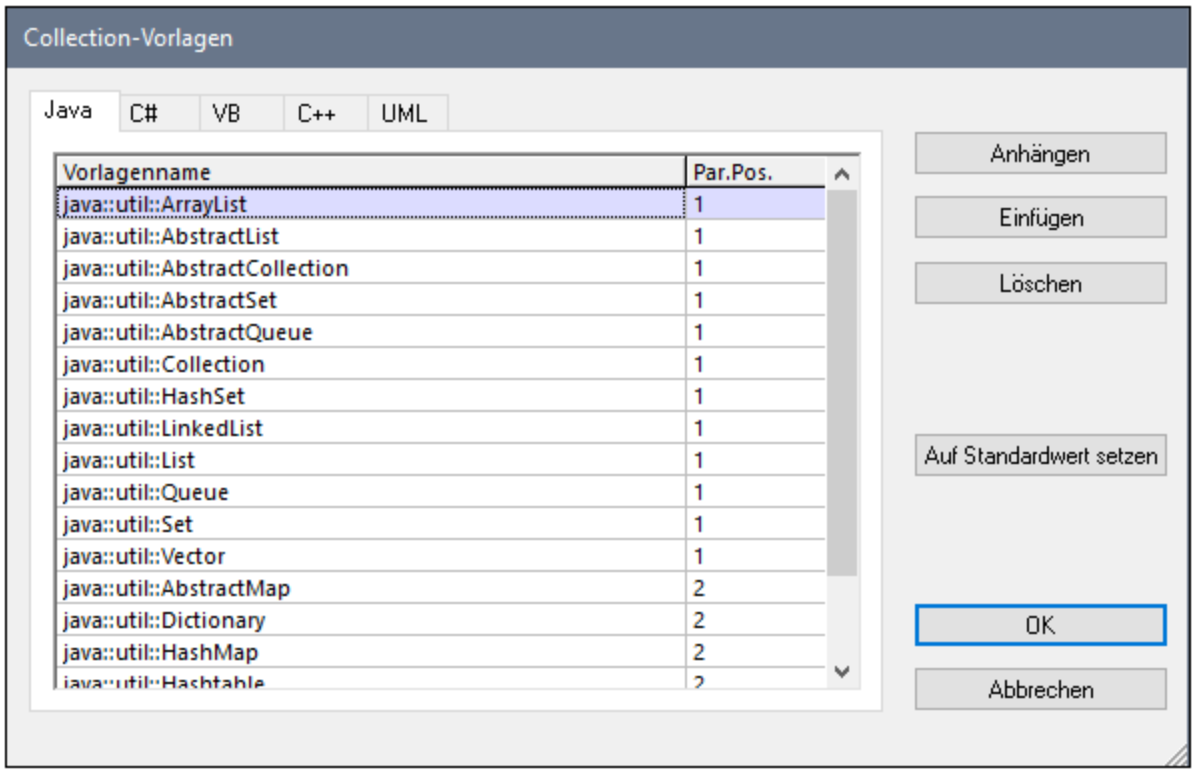
1. Klicken Sie im Menü **Extras** auf **Optionen**.
2. Klicken Sie auf das Register **Diagrammbearbeitung**.
3. Aktivieren bzw. deaktivieren Sie je nach Bedarf das Kontrollkästchen **Collections auflösen.**



Die Collection-Assoziationen werden standardmäßig auf Basis einer Liste vordefinierter Collection-Vorlagen aufgelöst. Um die vordefinierten Collection-Vorlagen anzuzeigen oder zu ändern, klicken Sie auf **Collection-Vorlagen**.

Mit Hilfe der Dialogfeld-Schaltflächen Anhängen, Einfügen bzw. Löschen können Sie benutzerdefinierte Collection-Arten einfügen. Die Spalte **Par.Pos.** gibt die Position des Parameters, der den Wertetyp der Collection enthält, an.

 © 2016-2022 Altova GmbH



Dialogfeld "Collection-Vorlagen"

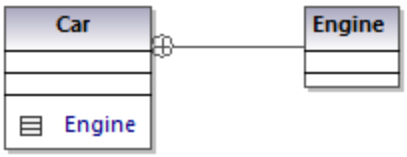
Um die Collection-Vorlagen auf ihre Standardwerte zurückzusetzen, klicken Sie auf **Auf Standardwerte setzen**.

5.3.6 Enthältbeziehung

Mit Hilfe einer Enthältbeziehungsassoziation werden z.B. Beziehungen Parent-Child-Beziehungen zwischen zwei Klassen oder Paketen dargestellt.

So stellen Sie eine Enthältbeziehung zwischen zwei Klassen dar:

1. Klicken Sie (in einer Klasse oder einem Paketdiagramm) auf die Symbolleisten-Schaltfläche **Enthältbeziehung** .
2. Ziehen Sie die Linie mit der Maus von der Klasse, die enthalten sein soll, auf die enthaltende Klasse.



Beachten Sie, dass die enthaltene Klasse - in diesen Fall `Engine` - jetzt in einem Bereich von `Car` zu sehen ist. Dadurch wird die enthaltene Klasse in denselben Namespace platziert wie die enthaltende Klasse.

© 2016-2022 Altova GmbH

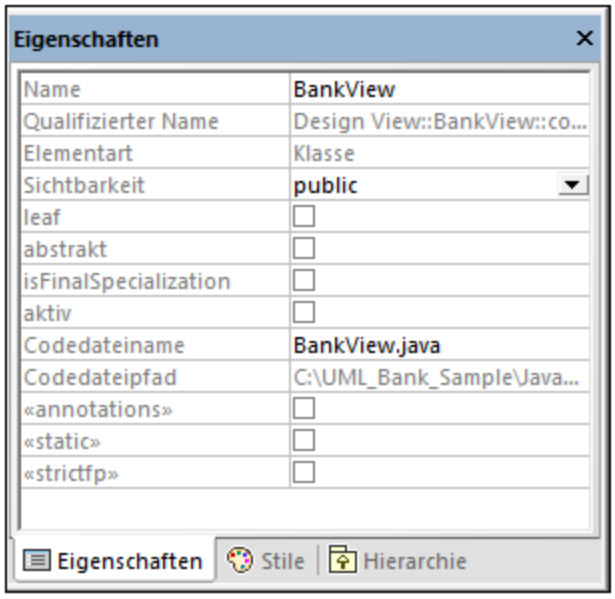
5.4 Stereotype und Eigenschaftswerte

Ein Stereotyp ist ein Erweiterungsmechanismus, mit dem ein vorhandenes XML-Element auf flexible Art erweitert werden kann, wodurch einige in Standard-UML nicht behandelten Aspekte abgedeckt werden können. Wenn Stereotype auf ein Element angewendet werden, bedeutet dies, dass das Element zu einem speziellen Zweck dient. Die vordefinierten UModel-Profile (C#, Java, VB.NET, usw.) enthalten alle für das Modellieren von Projekten in den entsprechenden Sprachen erforderlichen Stereotype. Sie können jedoch auch Ihre eigenen Profile (und die entsprechenden Stereotype) erstellen, siehe Erstellen und Anwenden von benutzerdefinierten Profilen 422.

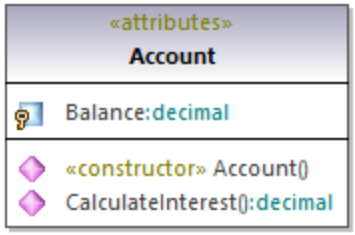
Wenn Sie Quellcode oder Binärdateien in das Modell importieren, wendet UModel automatisch auf Basis der Struktur des ursprünglichen Codes Stereotype auf Elemente an. Wenn z.B. im importierten Java-Quellcode annotations Modifier vorhanden sind, erhalten die entsprechenden Elemente im Modell das Stereotyp `«annotations»`. Informationen zu den Elemententsprechungen für verschiedene Sprachkonstrukte in UModel und deren Darstellung als Stereotype finden Sie unter UModel-Elementzuordnungen 228.

Sie können Stereotype auch manuell auf Elemente anwenden, während Sie diese modellieren. So können Sie etwas das Stereotyp `«attributes»` auf eine C#-Klasse anwenden, wodurch angegeben wird, dass die Klasse im generierten Code mit Attributen versehen werden muss. Um die Attributwerte im generierten Code zu definieren, können Sie in UModel so genannte "Eigenschaftswerte" hinzufügen, wie unter Anwenden von Stereotypen 147 gezeigt. Auch bei der XML-Schemamodellierung kommen sehr häufig Stereotype zum Einsatz, um Elemente wie simpleTypes, complexTypes, Facets, usw. zu definieren.

Auf der grafischen Benutzeroberfläche von UModel werden Stereotype innerhalb von doppelten spitzen Anführungszeichen (z.B. `«static»`) angezeigt. Alle in den vordefinierten UModel-Profilen enthaltenen Stereotype, werden im Fenster "Eigenschaften" angezeigt, wenn Sie auf ein Element klicken. Wenn Sie z.B. in der Modell-Struktur auf eine Java-Klasse klicken, werden im Fenster "Eigenschaften" ausschließlich Klassenstereotype für das Java-Profil angezeigt (in diesem Beispiel, `«annotations»`, `«static»`, `«strictfp»`).



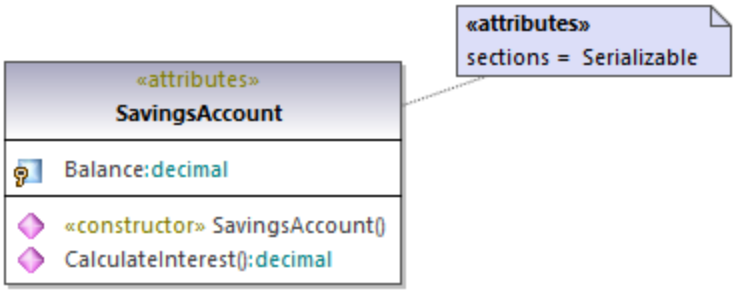
In Klassendiagrammen werden Stereotype oberhalb des Namens der Klasse angezeigt. Die unten gezeigte Klasse hat z.B. das Stereotyp `«attributes»`.



Bei Methoden oder Eigenschaften werden Stereotype inline angezeigt, wie z.B. das Stereotyp `«constructor»`, das in der oben gezeigten Klasse auf die Methode **Account()** angewendet wurde.

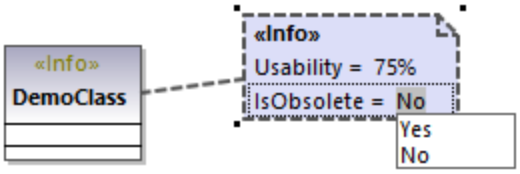
5.4.1 Eigenschaftswerte

Stereotype können mit Attributen (Eigenschaftswerten) verknüpft sein. Eigenschaftswerte sind Name-Wert-Paare, die Zusatzinformationen zum entsprechenden Stereotyp enthalten. So wurde z.B. auf die unten gezeigte Klasse das Stereotyp `«attributes»` angewendet. Beachten Sie, dass mit dem Stereotyp `«attributes»` Eigenschaftswerte verknüpft sind: ein Schlüssel(name) namens "sections" und der Wert "Serializable".



Eigenschaftswerte

Ein Stereotyp kann mehrere Eigenschaftswertpaare haben. Ein Wert kann auch aus einer Gruppe von Enumerationswerten ausgewählt werden.



Sie können die Art, wie Eigenschaftswerte im Diagramm angezeigt werden, ändern oder diese komplett ausblenden, siehe Anzeigen oder Ausblenden von Eigenschaftswerten 149. Informationen darüber, wie Sie die Eigenschaftswerte eines Stereotyps ändern können, finden Sie unter Anwenden von Stereotypen 147. Ein Beispiel für die Erstellung eines Stereotyps mit Eigenschaftswerten finden Sie unter Beispiel: Erstellen und Anwenden von Stereotypen 427.

 © 2016-2022 Altova GmbH

5.4.2 Anwenden von Stereotypen

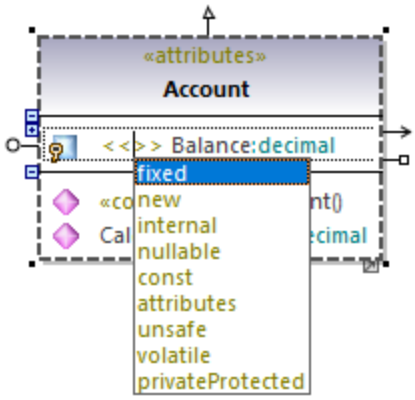
Durch Anwendung eines Stereotyps auf ein Element geben Sie an, dass das Element einen bestimmten Verwendungszweck hat. Im Falle der von UModel unterstützten Codesprachen (wie z.B. C#, VB.NET, Java) werden Stereotype normalerweise angewendet, um die Grammatik dieser Sprache abzubilden. So kann etwa auf eine Java-Klasse das Stereotyp `«static»` angewendet werden.

> Bevor Sie Stereotype anwenden können, muss zuerst das entsprechende Profil auf Ihr(e) Paket(e) angewendet werden. Dies wird in UModel automatisch durchgeführt, wenn Sie mit der rechten Maustaste auf ein Paket klicken und den Befehl **Code Engineering | Als {language} Namespace Root definieren** auswählen. Nähere Informationen dazu finden Sie unter Anwenden von UModel-Profilen 160.
>
> Wenn Sie benutzerdefinierte Profile erstellt haben, so müssen diese manuell auf das Paket angewendet werden, siehe Erstellen und Anwenden von benutzerdefinierten Profilen 422.

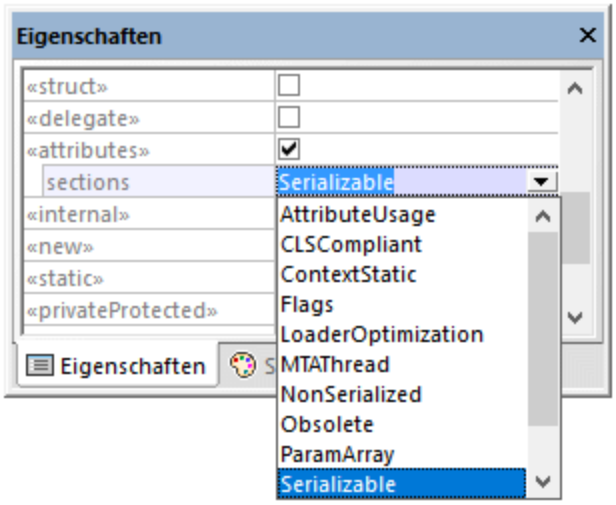
So wenden Sie ein Stereotyp auf ein Element an:

1. Klicken Sie im Fenster "Modell-Struktur" auf das Element. Wenn das Element durch Stereotype erweitert werden kann, werden diese im Fenster "Eigenschaften" als innerhalb von doppelte spitze Anführungszeichen ("«" und "»") gesetzte Eigenschaften angezeigt.
2. Aktivieren Sie im Fenster "Eigenschaften" das Kontrollkästchen des Stereotyps (z.B. `«static»`).

Sie können Stereotype auch anwenden, während Sie Elemente innerhalb eines Klassendiagramms erstellen. Klicken Sie dazu auf eine Eigenschaft einer Klasse und geben Sie innerhalb der Zeichen "<< und ">>" die ersten Buchstaben des Texts ein.

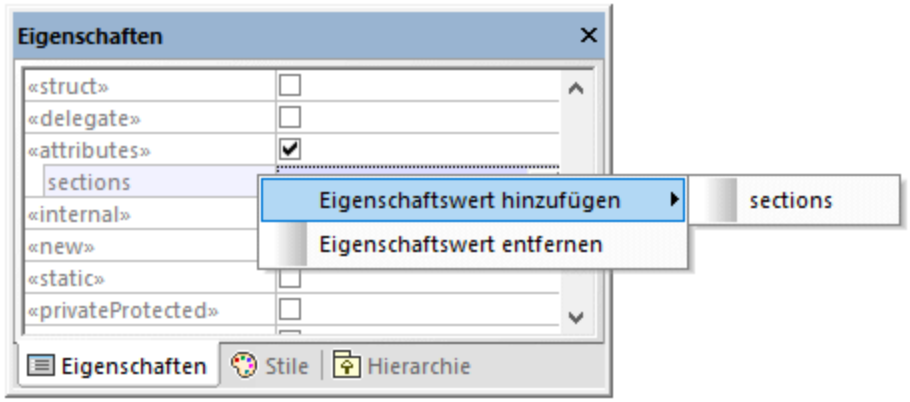


Einige Stereotype sind mit einer in UML als "Eigenschaftswerte" bezeichneten Liste von Name-Wert-Paaren verknüpft. Um ein Stereotyp mit Eigenschaftswerten auf ein Element anzuwenden, aktivieren Sie im Fenster "Eigenschaften" das Kontrollkästchen dieses Stereotyps (in diesem Beispiel `«attributes»`). Dadurch wird ein eingerückter Eintrag hinzugefügt, wo Sie den erforderlichen Wert aus einer vordefinierten Liste auswählen können.

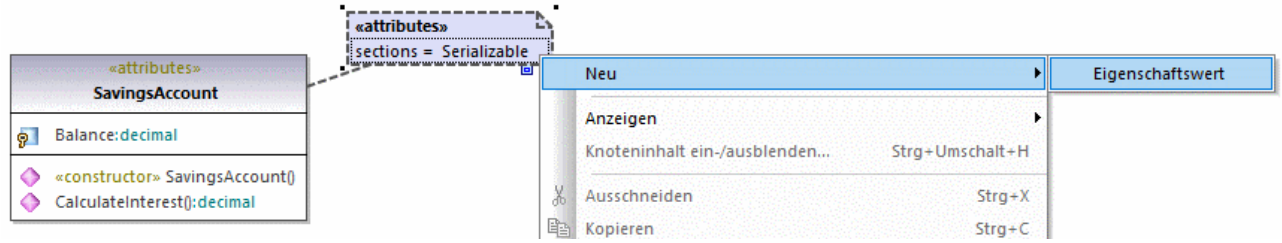


Eigenschaftswerte

Sie können auch mehrere Werte zum selben Schlüssel hinzufügen. Klicken Sie dazu mit der rechten Maustaste auf den eingerückten Eintrag und wählen Sie im Kontextmenü den Befehl **Eigenschaftswert hinzufügen | <name>**.



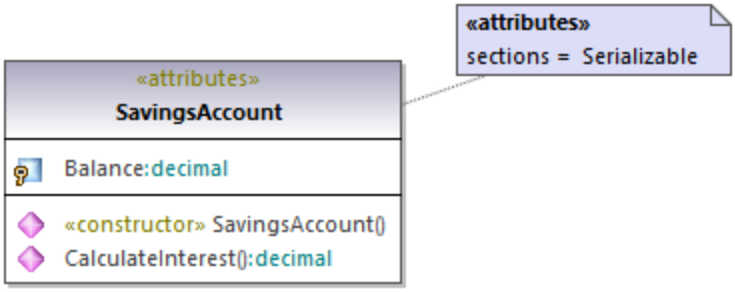
Alternativ dazu können Sie Eigenschaftswerte auch direkt über das Diagramm hinzufügen, indem Sie mit der rechten Maustaste auf einen Wert klicken und im Kontextmenü den Befehl **Neu | Eigenschaftswert** auswählen.



 © 2016-2022 Altova GmbH

5.4.3 Anzeigen oder Ausblenden von Eigenschaftswerten

Wenn ein Element Eigenschaftswerte hat, können Sie alle jeweiligen Eigenschaftswerte entweder in einem eigenen Kasten oder inline, als Bereich, anzeigen. Eigenschaftswerte können komplett ausgeblendet werden. Um auszuwählen, wie Eigenschaftswerte angezeigt werden sollen, klicken Sie mit der rechten Maustaste auf das Element im Diagramm und wählen Sie die Option **Eigenschaftswerte | <Anzeigeoption>** aus. Um z.B. alle Eigenschaftswerte außerhalb der Klasse anzuzeigen, klicken Sie im Diagramm mit der rechten Maustaste auf die Klasse und wählen Sie **Eigenschaftswerte | alle**. Um alle Eigenschaftswerte einer Klasse auszublenden, klicken Sie mit der rechten Maustaste auf die Klasse im Diagramm und wählen Sie den Befehl **Eigenschaftswerte | Keine**.

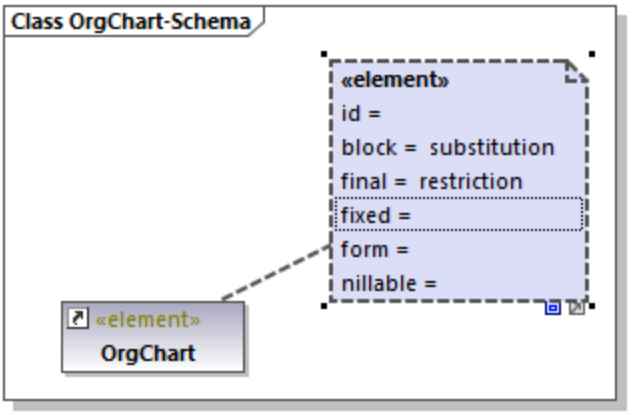


Außerhalb einer Klasse angezeigte Eigenschaftswerte

Ein- und Ausschalten des Kompaktmodus

Wenn einige Werte in einem Eigenschaftswertekasten leer sind, können Sie nur die leeren Werte folgendermaßen ausblenden:

1. Wählen Sie einen Eigenschaftswertekasten (mit sowohl leeren als auch nicht leeren Werten) im Diagramm aus.



2. Klicken Sie in der rechten unteren Ecke des Kastens auf den Ziehpunkt **Kompaktmodus ein-/ausschalten**.

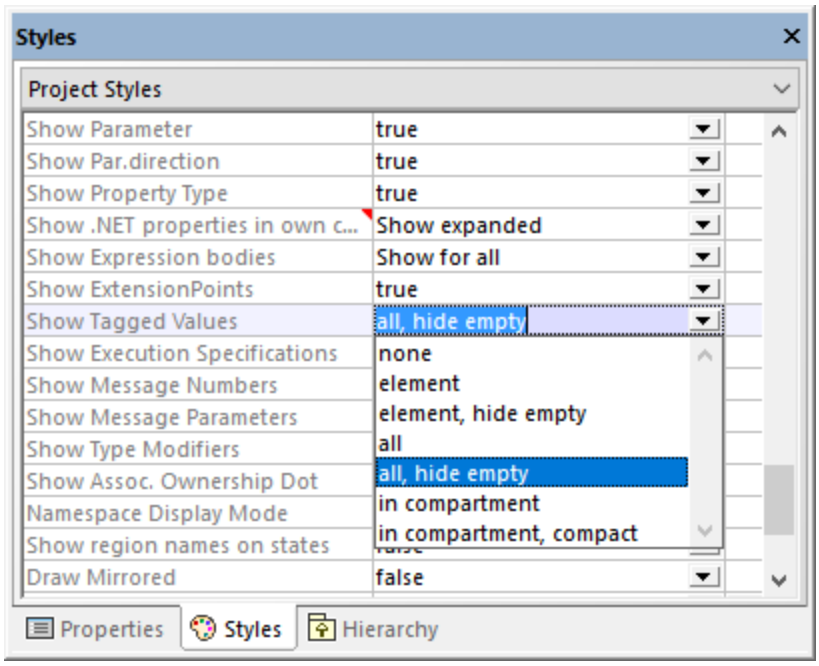
Im erweiterten Zustand werden auch die leeren Werte angezeigt. Im reduzierten Zustand werden die leeren Werte ausgeblendet.

Globale Änderung der Anzeige von Eigenschaftswerten

Sie können die Anzeigen von Eigenschaftswerten entweder, wie oben gezeigt, für einzelne Elemente oder global auf Projektebene ändern.

So ändern Sie die Anzeige von Eigenschaftswerten auf Projektebene:

1. Wählen Sie aus der Liste am oberen Rand des Fensters "Stile" 87 **Projektstile** aus.
2. Scrollen Sie hinunter bis zur Eigenschaft **Eigenschaftswerte anzeigen** und wählen Sie die gewünschte Option aus der Liste aus (z.B. **alle, leere ausblenden**).



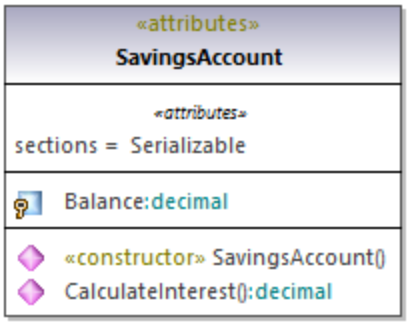
Informationen darüber, wie Sie Stile auf verschiedenen Ebenen ändern, finden Sie unter Ändern des Stils von Elementen 119.

Mögliche Anzeigeoptionen

In der Tabelle unten sehen Sie eine Liste möglicher Optionen für die Anzeige von Eigenschaftswerten. Diese Optionen sind ähnlich, wenn Sie Eigenschaftswerte global oder nur für einzelne Elemente ändern.

Keines	Blendet alle Eigenschaftswerte aus.
Alle	Zeigt die Eigenschaftswerte eines Elements (z.B. einer Klasse) sowie diejenigen von Elemente im Eigentum der Klasse wie z.B. Attributen und Operationen an.
Alle, leere ausblenden	Zeigt nur die Eigenschaftswerte, für die ein Wert vorhanden ist, an.

 © 2016-2022 Altova GmbH

Element	Zeigt die Eigenschaftswerte eines Elements (z.B. einer Klasse) an, nicht aber diejenigen von Attributen, Operationen, usw. im Eigentum der Klasse.
Element, leere ausblenden	Zeigt nur die Eigenschaftswerte eines Elements, für die ein Wert vorhanden ist, an.
Im Bereich	Zeigt die Eigenschaftswerte in einem separaten Bereich an. So hat etwa die unten gezeigte Klasse einen Bereich *«attributes»*, der Eigenschaftswerte enthält. 
Im Bereich, leere ausblenden	Zeigt nur die Eigenschaftswerte, für die ein Wert vorhanden ist, in einem Bereich an.
Im Bereich, kompakt	Wie oben.

6 Projekte und Code Engineering

Dieses Kapitel enthält Informationen zur Erstellung von UModel-Projekten (entweder von neuen Projekten oder durch Import von Daten aus Quellcode oder Binärdateien). Außerdem werden darin verschiedene Operationen im Zusammenhang mit Code Engineering mit UModel beschrieben. Dazu zählen:

- Forward Engineering (Generieren von Code anhand eines UModel-Projekts)
- Reverse Engineering (Importieren von Quellcode in ein UModel-Projekt)
- Roundtrip Engineering (Synchronisieren von Modell und Code in beide Richtungen je nach Bedarf)

Die Menübefehle zum Code Engineering stehen im Menü **Projekt** zur Verfügung. So können Sie etwa mit dem Menübefehl **Projekt | Quellprojekt importieren** C#- oder VB.NET Visual Studio-Lösungen oder Java-Code importieren und anhand dieses Codes UModel-Diagramme generieren. Wenn keine Projektlösung zur Verfügung steht, verwenden Sie den Menübefehl **Projekt | Quellverzeichnis importieren**, siehe Importieren von Quellcode (Reverse Engineering) 194. Auch Java-, C#- und VB.NET-Binärdateien können importiert werden, vorausgesetzt, es werden einige grundlegende Voraussetzungen erfüllt, siehe Importieren von Java-, C#- und VB.NET-Binärdateien 206.

Die oben erwähnten Code Engineering-Operationen können nicht nur auf Programmiersprachen, sondern auch auf Datenbanken und XML-Schemas angewendet werden. So können Sie etwa mit dem Menübefehl **Projekt | XML-Schema-Datei importieren** ein Reverse Engineering eines vorhandenen XML-Schemas durchführen und automatisch ein Klassendiagramm anhand dieser Datei generieren.

Unter UModel-Elementzuordnungen 228 finden Sie eine Liste von Zuordnungen zwischen UModel-Elementen und Elementen in den einzelnen unterstützten Sprachprofilen (darunter auch Datenbanken und XML-Schemas).

 © 2016-2022 Altova GmbH

6.1 Verwalten von UModel-Projekten

Ein UModel-Projekt dient als Container für UML-Modellierungselemente, Diagramme und verschiedene von Ihnen definierbare Einstellungen zum Projekt. UModel-Projekte werden mit der Dateierweiterung .ump (UModel-Projektdatei) gespeichert.
UModel zwingt Sie nicht, sich bei der Modellierung an eine bestimmte vordefinierte Reihenfolge zu halten. Sie können jeden Modellelementtyp zum Projekt hinzufügen: UML-Diagramm, Paket, Akteur usw. - und zwar in jeder beliebigen Reihenfolge bzw. an jeder beliebigen Stelle. Alle Modellelemente können auf dem Register "Modell-Struktur" eingefügt, umbenannt und gelöscht werden, d.h. Sie müssen diese Elemente nicht unbedingt als Teil eines Diagramms erstellen.

6.1.1 Erstellen, Öffnen und Speichern von Projekten

Wenn Sie UModel zum ersten Mal starten, wird automatisch ein neues Projekt geöffnet. Bei späteren Ausführungen wird das zuletzt verwendete Projekt geöffnet.

Anmerkung: UModel enthält eine Reihe von Beispielprojekten, anhand welcher Sie die Grundlagen der Modellierung erlernen und die grafische Benutzeroberfläche von UModel kennenlernen können. Diese Projekte befinden sich im folgenden Ordner: **C:\Benutzer\<Benutzername>\Dokumente\Altova\UModel2023\UModelExamples**.

So erstellen Sie ein neues Projekt:

- Klicken Sie im Menü **Datei** auf **Neu** (oder klicken Sie auf die Symbolleistenschaltfläche **Neu**).

Daraufhin wird ein neues Projekt mit dem Standardnamen **NeuesProjekt1** erstellt. Außerdem werden die folgenden Pakete automatisch zum Projekt hinzugefügt und im Fenster "Modell-Struktur" angezeigt.

- Root
- Component View

Diese beiden Pakete haben einen speziellen Zweck und sind die einzigen, die, wie im Tutorial unter Forward Engineering (Modell zu Code) 59 erläutert, nicht umbenannt oder gelöscht werden können.

Nachdem Sie das Projekt erstellt haben, können Sie Modellierungselemente wie UML-Pakete und Diagramme dazu hinzufügen, siehe Erstellen von Elementen 106 und Erstellen von Diagrammen 122.

So fügen Sie ein neues Paket hinzu:

1. Rechtsklicken Sie auf das Paket, unter dem das neue Paket angezeigt werden soll (in einem neuen Projekt entweder "Root" oder "Component View").
2. Wählen Sie im Kontextmenü den Befehl **Neues Element | Paket**.

Beachten Sie, dass Paket sowie andere Modellierungselemente auch über UML-Diagramme hinzugefügt werden können. Sie werden in diesem Fall automatisch im Fenster "Modell-Struktur" angezeigt.

So fügen Sie ein neues Diagramm hinzu:

- Klicken Sie mit der rechten Maustaste auf ein Paket in der Modell-Struktur und wählen Sie **Neues Diagramm**.

So fügen Sie Elemente zu einem Diagramm hinzu:

- Wählen Sie eine der folgenden Methoden:
  - Klicken Sie mit der rechten Maustaste auf das Diagramm und Wählen Sie **Neues Element | <Elementart>** aus dem Kontextmenü aus.
  - Ziehen Sie das gewünschte Element aus der Symbolleiste.

Ein Arbeitsbeispiel zur Erstellung eines Projekts und zum Generieren von Programmcode anhand dieses Projekts finden Sie unter Forward Engineering (Modell zu Code) 59.

So öffnen Sie ein vorhandenes Projekt:

- Klicken Sie im Menü **Datei** auf **Öffnen** und navigieren Sie zur .ump-Projektdatei.

Anmerkung: Externe Änderungen, die an der Projektdatei oder inkludierten Dateien extern vorgenommen wurden, werden automatisch erkannt und es erscheint eine entsprechende Meldung, ob das Projekt neu geladen werden soll. Diese Funktion kann über **Extras | Optionen** | Register **Datei** deaktiviert werden.

So speichern Sie ein Projekt:

- Klicken Sie im Menü Datei auf Speichern (oder **Speichern unter**).

Alle projektrelevanten Daten werden in der UModel-Projektdatei (Dateierweiterung *.**ump** (UModel-Projektdatei) gespeichert.

Anmerkung: Die *.ump-Datei ist ein XML-Dateiformat, auf das beim Speichern die Option "Pretty Print" aus dem Menü **Extras | Optionen |** Register **Datei** angewendet werden kann.

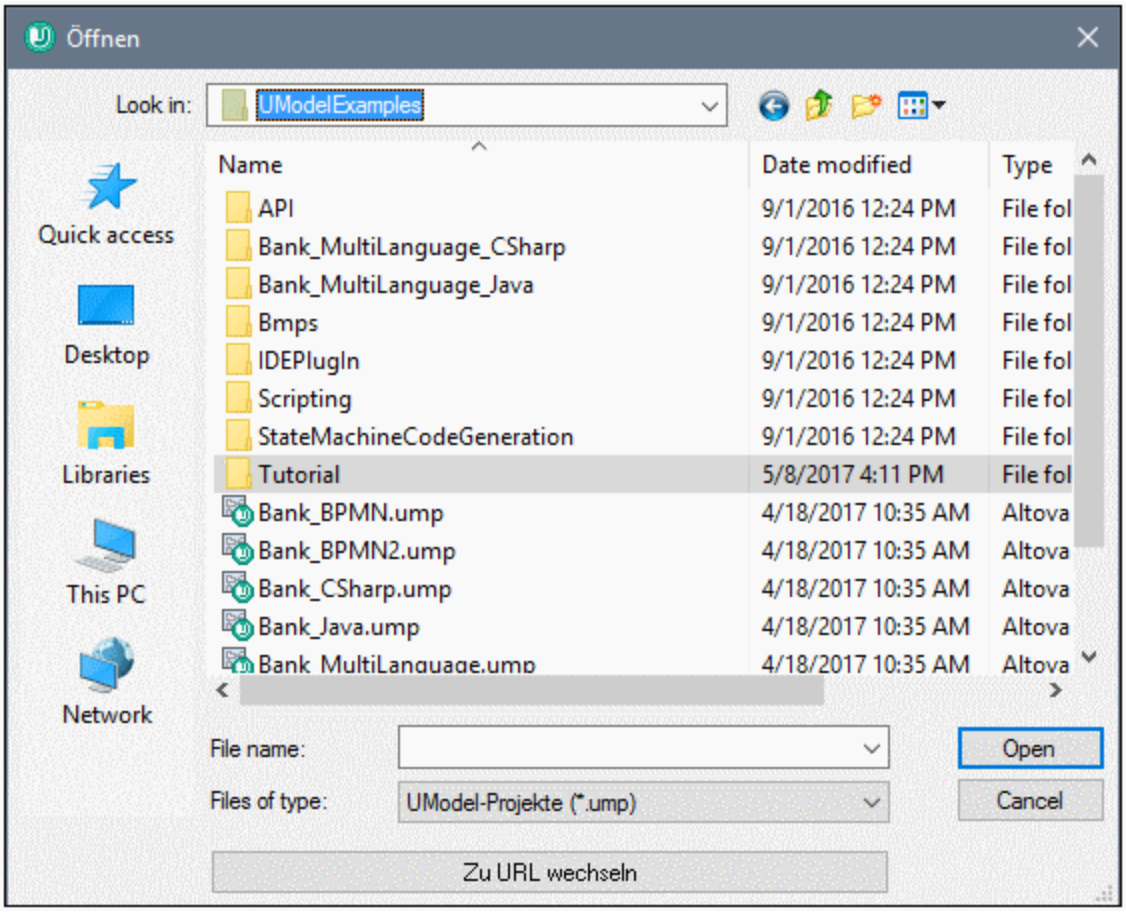
6.1.2 Öffnen von Projekten über eine URL

Zusätzlich zu lokalen Projekt-Dateien können Sie Dateien auch über eine URL öffnen. Es werden die Protokolle HTTP, HTTPS und FTP unterstützt. Beachten Sie, dass von URLs geladene Dateien nicht wieder unter ihrem ursprünglichen Pfad gespeichert werden können (d.h. Sie haben nur Lesezugriff auf die Datei), es sei denn, Sie haben diese, wie unten gezeigt, von einem Microsoft® SharePoint® Server ausgecheckt.

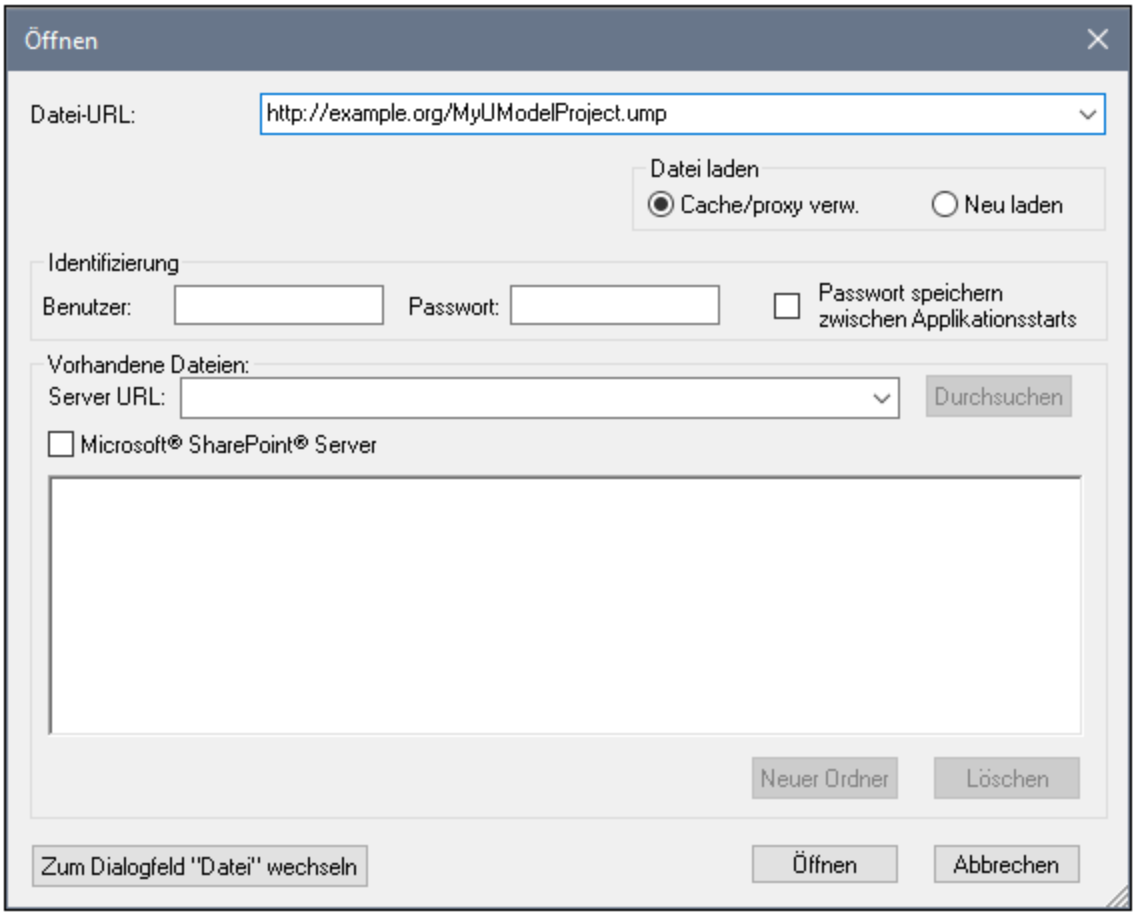
So öffnen Sie eine Datei über eine URL:

1. Klicken Sie im Dialogfeld **Öffnen** auf **Zu URL wechseln**.

 © 2016-2022 Altova GmbH



2. Geben Sie die URL der Datei in das Textfeld **Datei-URL** ein und klicken Sie auf **Öffnen**.



Falls für den Server eine Passwort-Authentifizierung erforderlich ist, werden Sie aufgefordert, den Benutzernamen und das Passwort einzugeben. Wenn Sie möchten, dass sich UModel den Benutzernamen und das Passwort für das nächste Mal merkt, geben Sie diese in das Dialogfeld "Öffnen" ein und aktivieren Sie das Kontrollkästchen **Passwort speichern zwischen Applikationsstarts**.

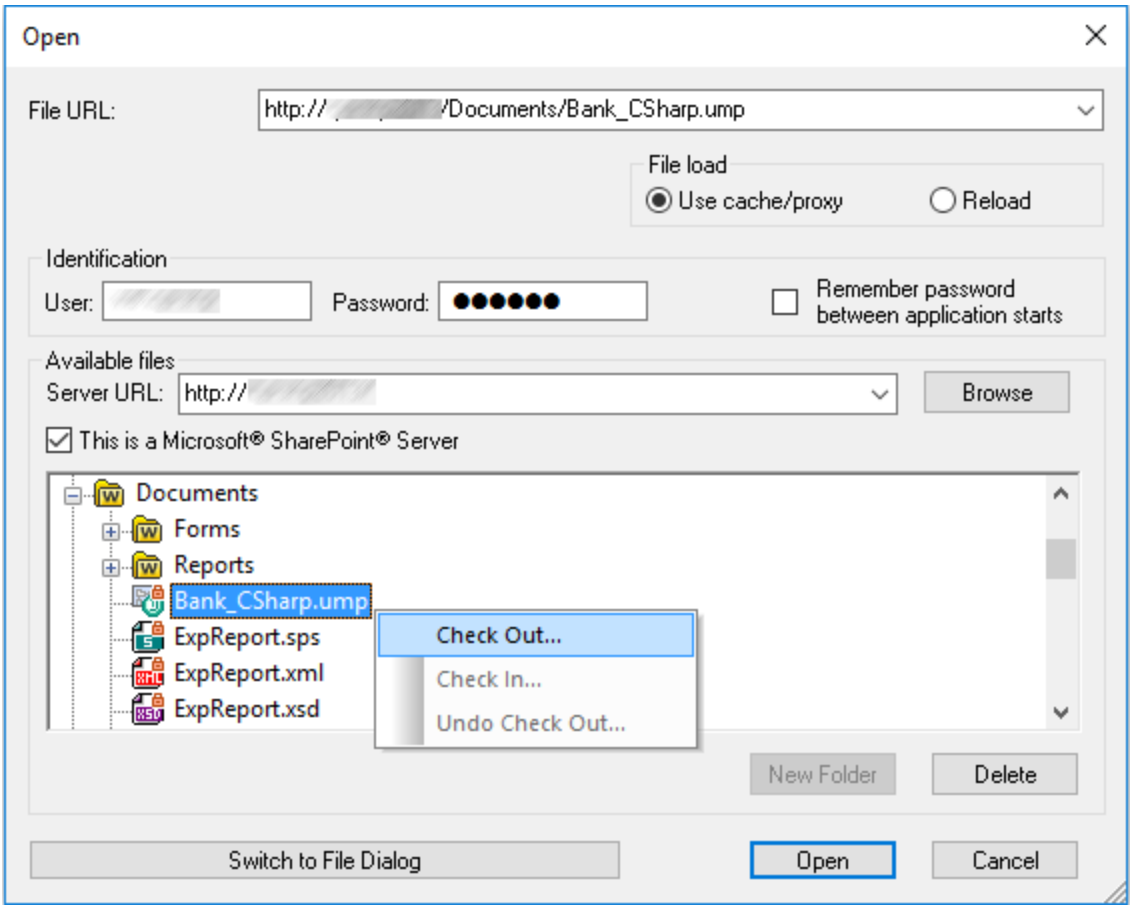
Wenn sich die zu ladende Datei wahrscheinlich nicht ändern wird, aktivieren Sie die Option **Cache/proxy verwenden,** damit die Daten im Cache gespeichert werden und die Datei schneller geladen wird. Wenn die Datei bei jedem Start von UModel neu geladen werden soll, wählen Sie **Neu laden**.

Für Server mit Unterstützung für Web Distributed Authoring and Versioning (WebDAV) können Sie die Verzeichnisse nach Eingabe der **Server-URL** in das Textfeld Server URL und Klicken auf **Durchsuchen** durchsuchen.

Anmerkung: Die Funktion **Durchsuchen** steht nur auf Servern, die WebDAV unterstützen und auf Microsoft SharePoint Servern zur Verfügung.

Wenn es sich beim Server um einen Microsoft® SharePoint® Server handelt, aktivieren Sie das Kontrollkästchen **Microsoft® SharePoint® Server**. Daraufhin wird im Vorschaubereich angezeigt, ob die Datei eingecheckt oder ausgecheckt ist.

 © 2016-2022 Altova GmbH



Dateien können einen der folgenden Status haben:

	Eingecheckt. Die Datei kann ausgecheckt werden.
	Von einem anderen Benutzer ausgecheckt. Kann nicht ausgecheckt werden.
	Lokal ausgecheckt. Kann bearbeitet und eingecheckt werden.

Um die Datei in UModel ändern zu können, klicken Sie mit der rechten Maustaste auf die Datei und wählen Sie den Befehl **Auschecken**. Wenn eine Datei von Microsoft® SharePoint® ausgecheckt wurde, werden die Änderungen in der Datei beim Speichern der Datei in UModel zurück an den Server gesendet. Um die Datei am Server wieder einzuchecken, klicken Sie im Dialogfeld oben mit der rechten Maustaste auf die Datei und wählen Sie im Kontextmenü den Befehl **Einchecken** (Melden Sie sich alternativ dazu am Server an und führen Sie diese Operation direkt vom Browser aus aus). Um Änderungen, die seit dem letzten Check-out an der Datei vorgenommen wurden, zu verwerfen, klicken Sie mit der rechten Maustaste auf die Datei und wählen Sie den Befehl **Auschecken rückgängig** (oder führen Sie diese Operation über den Browser aus).

Beachten Sie dazu Folgendes:

- Wenn eine Datei bereits von einem anderen Benutzer ausgecheckt wurde, steht Sie nicht zum Auschecken zur Verfügung.
- Wenn Sie eine Datei in einer Altova-Applikation auschecken, können Sie sie nicht in einer anderen

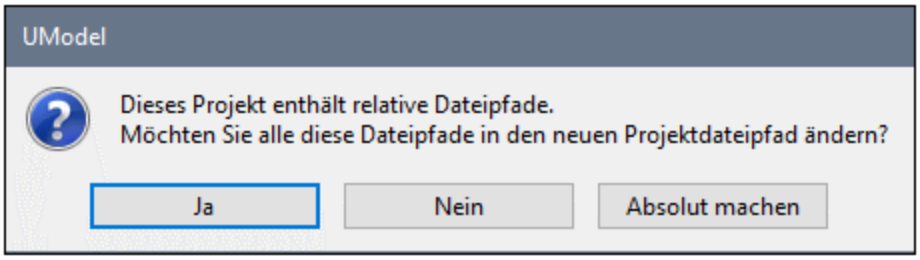
Altova-Applikation auschecken. Die Datei gilt bereits als für Sie ausgecheckt.

6.1.3 Verschieben von Projekten in ein neues Verzeichnis

UModel-Projekte und generierter Code können ganz einfach in ein anderes Verzeichnis (oder auf einen anderen Computer) verschoben und dort erneut synchronisiert werden.

Dazu gibt es zwei Methoden:

- Wählen Sie die Menüoption **Datei | Speichern unter...** und klicken Sie auf **Ja**, wenn Sie gefragt werden, ob die Dateipfade an den neuen Projektordner angepasst werden sollen.

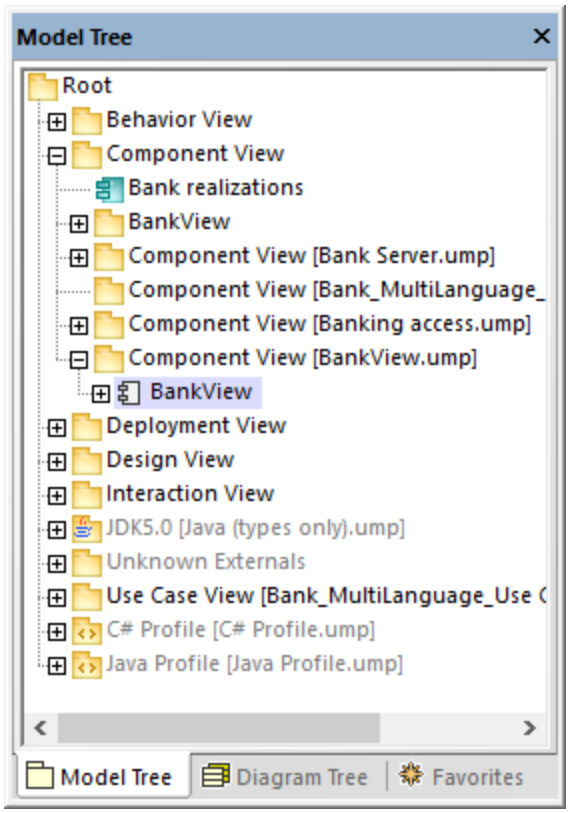


- Kopieren Sie das UModel-Projekt (*.ump) in einen neuen Ordner und passen Sie die Codegenerierungspfade für die einzelnen beteiligten Komponenten an.

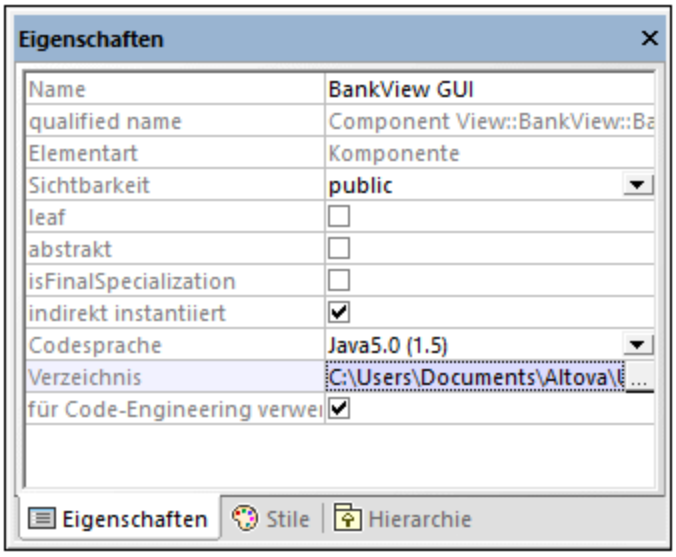
Als Beispiel für die zweite Methode öffnen Sie das folgende Beispielprojekt: **C:\Benutzer\<Benutzername>\Dokumente\Altova\UModel2023\UModelExamplesBank_Multilanguage.ump**.

1. Suchen Sie in der Modell-Struktur die Komponente `BankView`.

 © 2016-2022 Altova GmbH



2. Suchen Sie im Fenster "Eigenschaften" die Eigenschaft **Verzeichnis**, und geben Sie dort den neuen Pfad an.



3. Synchronisieren Sie das Modell erneut mit dem Code.

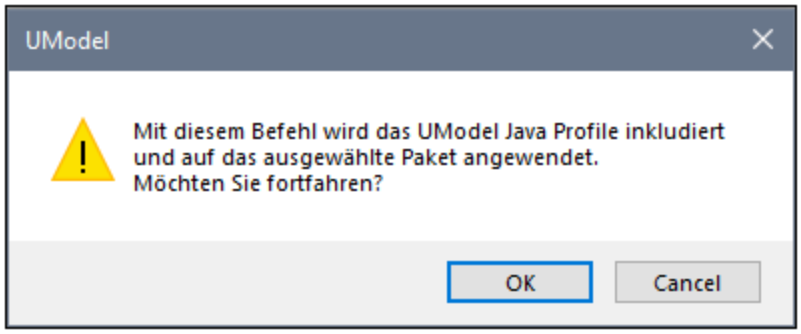
6.1.4 Anwenden von UModel-Profilen

Wenn Sie in UModel ein neues Modellierungsprojekt beginnen, sind im Projekt standardmäßig noch keine Geschäftsanwendungen oder Code Engineering-Sprachen definiert, die Sie benötigen werden. Um daher für Ihr UML-Projekt eine Domain oder Sprache einzustellen, müssen Sie ein *Profil darauf anwenden*.

Es wird zwischen zwei Profilarten unterschieden:

- In UModel vordefinierte Profile (dazu gehören C#, VB.NET, Java, usw.).
- Benutzerdefinierte Profile, die Sie erstellen können, um UML für Ihre spezifische Domain bzw. Ihre Bedürfnisse zu erweitern.

Über den Menübefehl **Projekt | Unterprojekt inkludieren** können Sie jedes der vordefinierten Profile zu Ihrem Projekt hinzufügen. Außerdem werden Sie, immer, wenn Sie eine Aktion durchführen, für die ein bestimmtes Profil benötigt wird, von UModel aufgefordert, ein vordefiniertes Profil anzuwenden. Wenn Sie z.B. mit der rechten Maustaste auf ein neues Paket klicken und im Kontextmenü die Option **Code Engineering | Als Java Namespace Root definieren** auswählen, werden Sie aufgefordert, das Java-Profil darauf anzuwenden.



Um die vollständige Liste der vordefinierten UModel-Profile anzuzeigen oder diese manuell zu Ihrem Modell hinzuzufügen, wählen Sie den Menübefehl **Projekt | Unterprojekt inkludieren**. Siehe auch Inkludieren von Unterprojekten 164.

Eine Anleitung zum Erstellen von benutzerdefinierten Profilen zur Erweiterung oder Anpassung von UML finden Sie unter Erstellen und Anwenden von benutzerdefinierten Profilen 422.

6.1.5 Aufteilen von UModel-Projekten

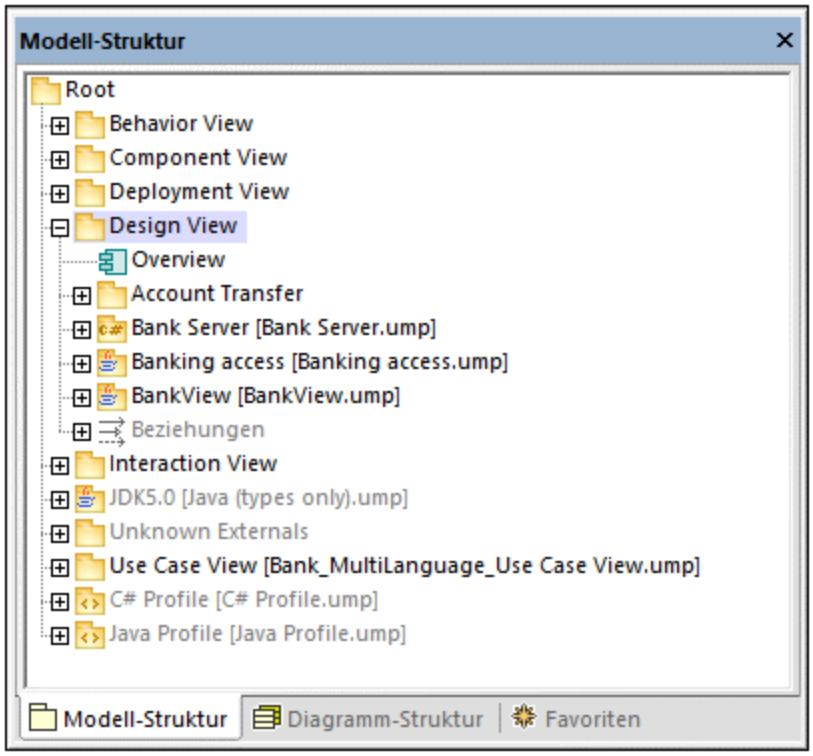
UModel-Projekte können in mehrere Unterprojekte aufgeteilt werden, sodass mehrere Entwickler unterschiedliche Teile eines einzigen Projekts gleichzeitig bearbeiten können. Unterprojekte sind genau wie UModel-Standardprojektdateien und haben dieselbe *.ump-Erweiterung. Die einzelnen Unterprojekte können zu einem Versionskontrollsystem hinzugefügt werden. Das Projekt der obersten Ebene wird als als Hauptprojekt bezeichnet.

Unterprojekte können anhand beinahe jeden Pakets im Hauptprojekt erstellt werden. Sie können auswählen, ob das Unterprojekt vom Hauptprojekt aus bearbeitet werden können soll oder ob es schreibgeschützt sein soll. Wenn es schreibgeschützt ist, kann das Unterprojekt nur bearbeitet werden, wenn es als eigenständiges Projekt geöffnet wird.

 © 2016-2022 Altova GmbH

Unterprojekte können beliebig strukturiert werden, in einer flachen oder einer hierarchischen Struktur oder in Form einer Kombination aus beiden. Dadurch kann praktisch jedes Paket eines Hauptprojekts in Unterprojektdateien aufgeteilt werden.

Im Fenster "Modell-Struktur" 80 werden Unterprojekte innerhalb von eckigen Klammern mit dem jeweiligen .ump-Dateinamen rechts davon angezeigt. So enthält etwa das unten gezeigte Projekt (**Bank_MultiLanguage.ump** aus dem Verzeichnis **C:\Benutzer\<Benutzername>\Dokumente\Altova\UModel2023\UModelExamples**) mehrere Unterprojekte.

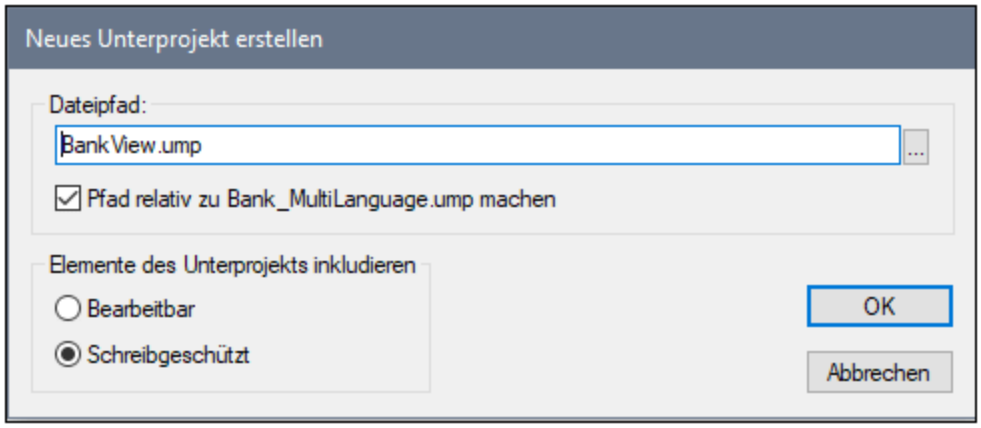


Beim Code Engineering werden alle untergeordneten Komponenten eines Unterprojekts berücksichtigt. Es besteht kein Unterschied zwischen einer einzelnen Projektdatei oder einer Projektdatei, die aus mehreren editierbaren Unterprojekten besteht. Dies gilt auch für UML-Diagramme. Auch diese Diagramme können auf der Haupt- oder Unterprojektebene bearbeitet werden.

Anmerkung: Sie können Pakete und darin enthaltene UML-Diagramme auch in verschiedenen Projekten gemeinsam verwenden. Nähere Informationen dazu finden Sie unter Freigeben von Paketen und Diagrammen 166.

Erstellen von Unterprojekten

Um ein Unterprojekt zu erstellen, klicken Sie mit der rechten Maustaste auf ein Paket und wählen Sie im Kontextmenü den Befehl **Unterprojekt | Neues Unterprojekt erstellen**.



Klicken Sie auf die Schaltfläche **Durchsuchen** und wählen Sie das Verzeichnis, in dem das Unterprojekt gespeichert werden soll, aus.

Aktivieren Sie das Optionsfeld **Bearbeitbar**, damit das Unterprojekt vom Hauptprojekt aus bearbeitet werden kann. (Wenn Sie "Schreibgeschützt" auswählen, kann es im Hauptprojekt nicht bearbeitet werden.)

Anmerkung: Der Dateipfad des Unterprojekts kann jederzeit durch Rechtsklick auf das Unterprojekt und Auswahl des Befehls **Unterprojekt | Dateipfad bearbeiten** geändert werden.

Öffnen und Bearbeiten von Unterprojektdateien

Sie können ein Unterprojekt direkt vom Hauptprojekt aus als eigenes UModel-Projekt öffnen. Dazu sollte es keine nicht aufgelösten Referenzen auf andere Elemente geben. UModel führt bei Erstellung eines Unterprojekts anhand des Hauptprojekts und beim Speichern einer Datei automatisch eine Überprüfung durch.

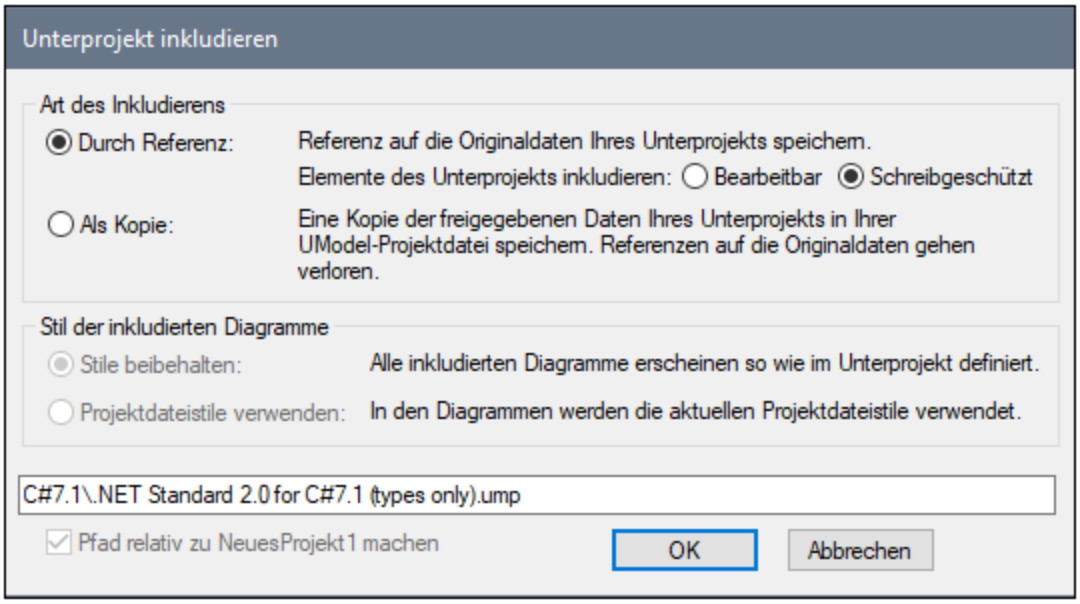
Um ein Unterprojekt als eigenständiges Projekt zu öffnen, klicken Sie mit der rechten Maustaste im Hauptprojekt auf das Unterprojektpaket und wählen Sie den Befehl **Unterprojekt | Als Projekt öffnen.** Daraufhin wird eine weitere Instanz von UModel gestartet und das Unterprojekt wird als Hauptprojekt geöffnet. Alle nicht aufgelösten Referenzen werden im Fenster "Meldungen" angezeigt.

Wiederverwendung von Unterprojekten

Unterprojekte, die von einem Hauptprojekt abgespalten wurden, können in jedem beliebigen anderen Hauptprojekt verwendet werden.

1. Öffnen Sie ein Projekt und wählen Sie den Befehl **Projekt | Unterprojekt inkludieren**.
2. Klicken Sie auf die Schaltfläche "Durchsuchen" und wählen Sie die gewünschte .ump-Datei aus.

 © 2016-2022 Altova GmbH



3. Wählen Sie aus, wie die Datei inkludiert werden soll; durch eine Referenz oder als Kopie.

Speichern von Projekten

Wenn Sie die Hauptprojektdatei speichern, werden auch alle editierbaren Unterprojektdateien gespeichert. Sie sollten daher außerhalb der freigegebenen/Unterprojektstruktur keine Daten (Komponenten) erstellen/hinzufügen, wenn das Unterprojekt in einer Hauptprojektdatei als "editierbar" definiert ist. Wenn es noch Daten außerhalb der Unterprojektstruktur gibt, wird im Fenster "Meldungen" eine Warnmeldung angezeigt.

Speichern von Unterprojektdateien

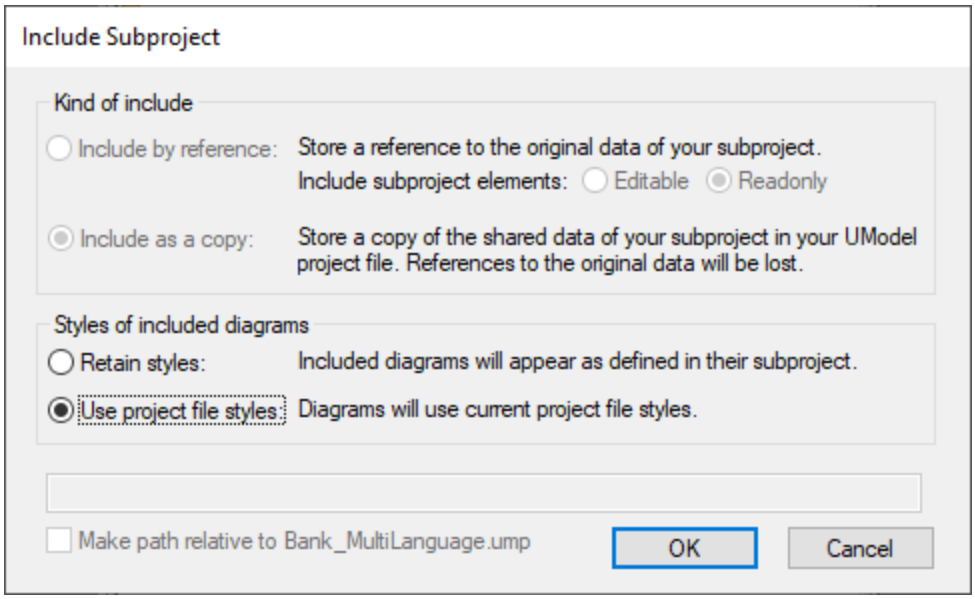
Beim Speichern von Unterprojekten (von der Hauptprojektebene aus) werden alle Referenzen auf gleichrangige und untergeordnete Unterprojekte berücksichtigt und gespeichert. Wenn z.B. zwei gleichrangige Unterprojekte, nämlich sub1 und sub2 existieren und in sub1 Elemente aus sub2 verwendet werden, dann wird sub1 so gespeichert, dass darin automatisch auch Referenzen auf sub2 enthalten sind.

Wenn sub1 als ein Hauptprojekt geöffnet wurde, so wird es als unabhängiges Projekt betrachtet und kann ohne Referenz auf das eigentliche Hauptprojekt bearbeitet werden.

Wiedereingliedern von Unterprojekten in das Hauptprojekt

Zuvor definierte Unterprojekte können wieder zurück in das Hauptprojekt kopiert werden. Wenn das Unterprojekt keine Diagramme enthält, so wird es sofort wieder eingegliedert. Falls das Unterprojekt Diagramme enthält, wird das folgende Dialogfeld angezeigt:

1. Klicken Sie mit der rechten Maustaste auf das Unterprojekt und wählen Sie den Befehl **Unterprojekt | Als Kopie inkludieren**. Daraufhin wird das Dialogfeld "Unterprojekt inkludieren" geöffnet, in dem Sie die Diagrammstile, die Sie beim Inkludieren des Unterprojekts verwenden möchten, definieren können.



2. Wählen Sie die gewünschte Stiloption aus und klicken Sie anschließend auf **OK**.

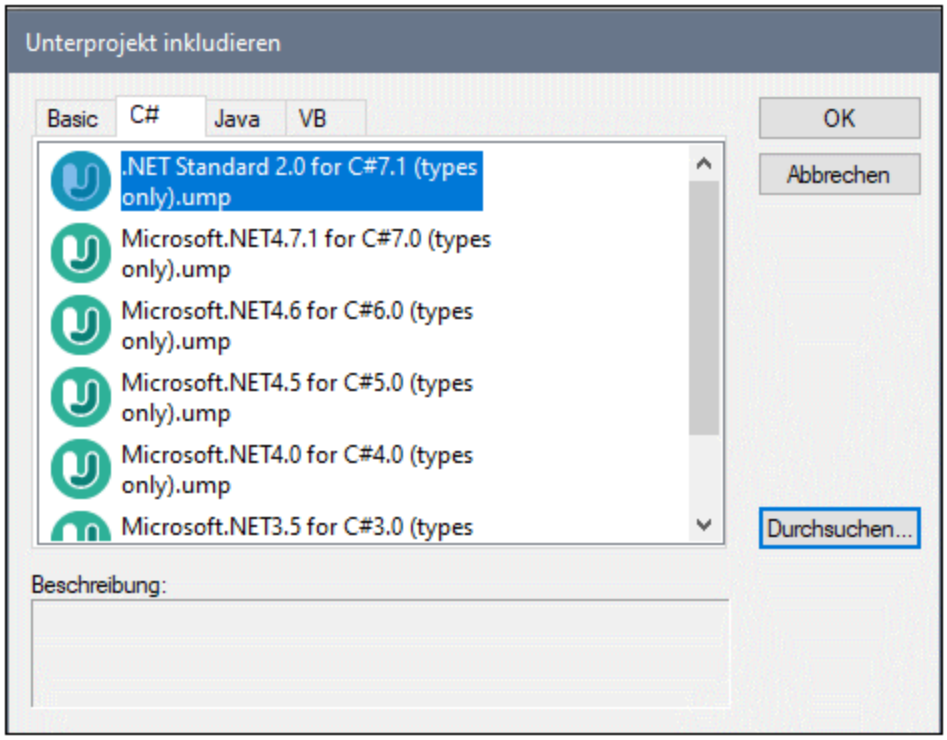
6.1.6 Inkludieren von Unterprojekten

Wenn Sie Code anhand eines Modells generieren oder Quellcode in ein Modell importieren möchten, muss ein Profilprojekt für die jeweilige Sprachen (z.B. für C#, Java, VB.NET) in Ihr UModel-Projekt inkludiert werden.

Um ein UModel-Projekt als Unterprojekt eines anderen UModel-Projekts zu inkludieren, wählen Sie den Menübefehl **Projekt | Unterprojekt inkludieren.** Wie unten gezeigt, stehen auf dem Register **Basic** diverse .ump-Unterprojekte (für das Code Engineering erforderliche Sprachprofile) zur Verfügung. Zusätzlich dazu steht eine Reihe von .ump-Unterprojekten mit C#-, Java- und VB.NET-Typen nach Version geordnet auf den entsprechenden Registern zur Verfügung.

Damit beim Code Engineering alle Typen richtig erkannt werden, müssen sowohl das Sprachprofil (z.B. das **C#-Profil**) als auch das Typprojekt der entsprechenden Sprachversion (z.B. **.NET 5 für C# 9.0**) inkludiert werden, da sonst im Projekt ein Paket namens "Unbekannte externe Elemente", das alle nicht erkannten Typen enthält, erstellt wird.

 © 2016-2022 Altova GmbH

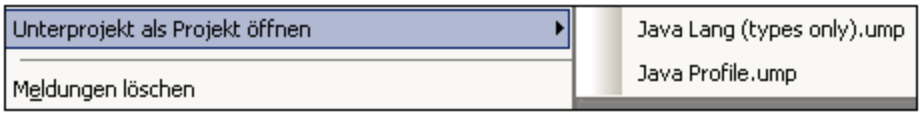


Dialogfeld "Unterprojekt inkludieren"

Die Register und UModel-Projekte im Dialogfeld "Unterprojekt inkludieren" sind konfigurierbar. UModel liest diese Informationen aus dem folgenden Pfad (relativ zum Ordner "Programme" Ihres Betriebssystems): **\Altova\UModel2023\UModelInclude**. Beachten Sie, dass sich die Projektdateien auf dem Register **Basic** direkt unterhalb des Ordners **UModelInclude** befinden, während sich die Projekte auf den verschiedenen Java-, VB- und C#-Registern in Unterordnern des Ordners **UModelInclude** befinden.

So zeigen Sie alle derzeit importierten Projekte an:

- Wählen Sie die Menüoption **Projekt | Unterprojekt separat öffnen**.
  Das Menü, das erscheint, enthält die aktuell inkludierten Unterprojekte.



So erstellen Sie im Dialogfeld "Unterprojekt inkludieren" ein benutzerdefiniertes Register:

- Navigieren Sie zum Ordner **\Altova\UModel2023\UModelInclude** (relativ zu Ihrem Ordner "Programme") und erstellen Sie Ihren benutzerdefinierten Ordner unterhalb, z.B. **UModelInclude\myfolder.** Als Name des Registers im Dialogfeld "Unterrprojekt inkludieren" wird der Name, den Sie dem Ordner geben, angezeigt.

- Kopieren Sie alle .ump-Dateien, die Sie auf dem entsprechenden Register zur Verfügung stellen möchten, in Ihren benutzerdefinierten Ordner.

So erstellen Sie beschreibenden Text zu jeder UModel Projektdatei:

- Erstellen Sie eine Textdatei mit demselben Namen wie die *.ump-Datei und platzieren Sie sie in denselben Ordner. So wird z.B. für die Datei **MyModel.ump** eine beschreibende Datei names **MyModel.txt** benötigt. Stellen Sie bitte sicher, dass die Codierung dieser Textdatei UTF-8 ist.

So entfernen Sie ein inkludiertes Projekt:

1. Klicken Sie in der Modell-Strukturansicht auf das inkludierte Paket und drücken Sie die Entf-Taste.
2. Wenn Sie gefragt werden, ob Sie mit dem Löschen fortfahren wollen, klicken Sie auf OK, um die inkludierte Datei aus dem Projekt zu löschen.

So löschen oder entfernen Sie ein Projekt aus dem Dialogfeld "Unterprojekt inkludieren":

- Löschen oder entfernen Sie die .ump-Datei (MyModel.ump) aus dem entsprechenden Ordner.

6.1.7 Freigeben von Paketen und Diagrammen

Pakete (und eventuell darin enthaltene UML-Diagramme) können freigegeben und in verschiedenen UMode-Projekten verwendet werden. Pakete können mittels Referenz oder in Form einer Kopie in andere UModel-Projekte inkludiert werden.

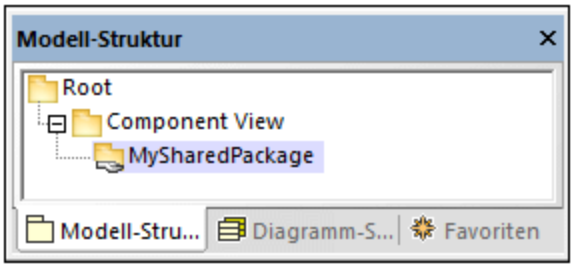
Beachten Sie bitte außerdem, dass Unterprojektdateien jederzeit von einem Haupt- oder Unterprojekt abgeteilt werden können. Die Unterprojektdateien können vom Hauptprojekt aus als editierbare oder als schreibgeschützte Projekte inkludiert werden; die einzelnen Pakete werden freigegeben und als Unterprojektdatei gespeichert und können zu einem Versionskontrollsystem hinzugefügt werden. Nähere Informationen dazu finden Sie unter Aufteilen von UModel-Projekten 160.

Anmerkungen
- Damit ein Paket freigegeben werden kann, darf es keine Links zu externen Elementen (Elementen außerhalb des freigegebenen Bereichs) enthalten.
- Wenn Sie UModel-Projektdateien erstellen, verwenden Sie nicht eine einzige Projektdatei als "Vorlage/Kopie" für eine andere Projektdatei, in der Sie ein Paket freigeben möchten. Dies kann zu Konflikten führen, da jedes Element global nur einmal vorhanden sein sollte (siehe uuid 455). Dies wäre dann nicht der Fall, da zwei Projekte dann Elemente mit identischen uuids hätten.

So verwenden Sie ein Paket in mehreren Projekten:

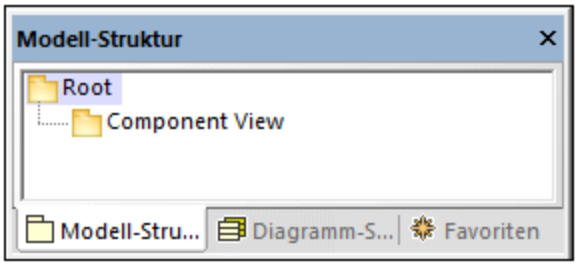
- Rechtsklicken Sie auf dem Register "Modell-Struktur" auf ein Paket und wählen Sie **Unterprojekt | Paket freigeben**. In der Modell-Struktur wird unterhalb des freigegebenen Pakets ein "freigegeben"-Symbol angezeigt. Dieses Paket kann nun in jedes beliebige andere UModel-Projekt inkludiert werden.

 © 2016-2022 Altova GmbH

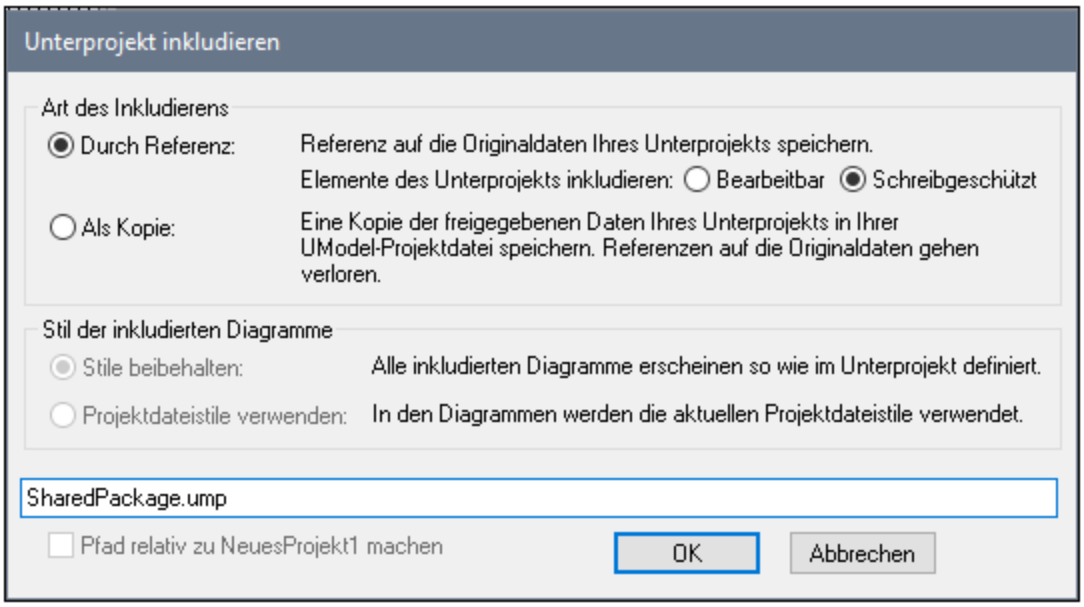


So inkludieren/importieren Sie einen freigegebenen Ordner in ein Projekt:

1. Öffnen Sie das Projekt, das das freigegebene Paket enthalten soll (in diesem Beispiel ein leeres Projekt).

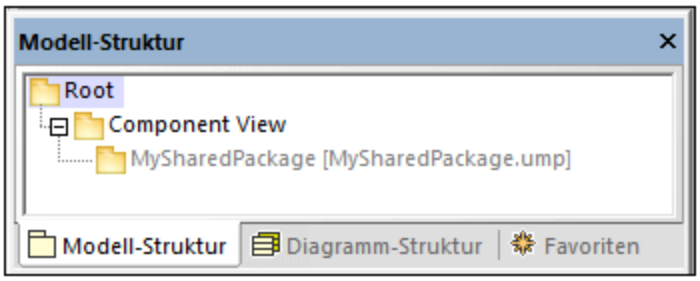


2. Wählen Sie den Menüeintrag **Projekt | Unterprojekt inkludieren...**
3. Klicken Sie auf die **Durchsuchen**-Schaltfläche, wählen Sie das Projekt, das das freigegebene Paket enthält, aus und klicken Sie auf **Öffnen**. Im Dialogfeld "Unterprojekt inkludieren" können Sie das Paket/Projekt in Form einer Referenz oder als Kopie inkludieren.



4. Wählen Sie die gewünschte Option (In diesem Beispiel "Durch Referenz") und klicken Sie auf OK.

Im neuen Paket ist nun das Paket "Deployment View" zu sehen. Das Quellprojekt des Pakets wird in Klammern angezeigt (in diesem Beispiel **SharedPackage.ump**).

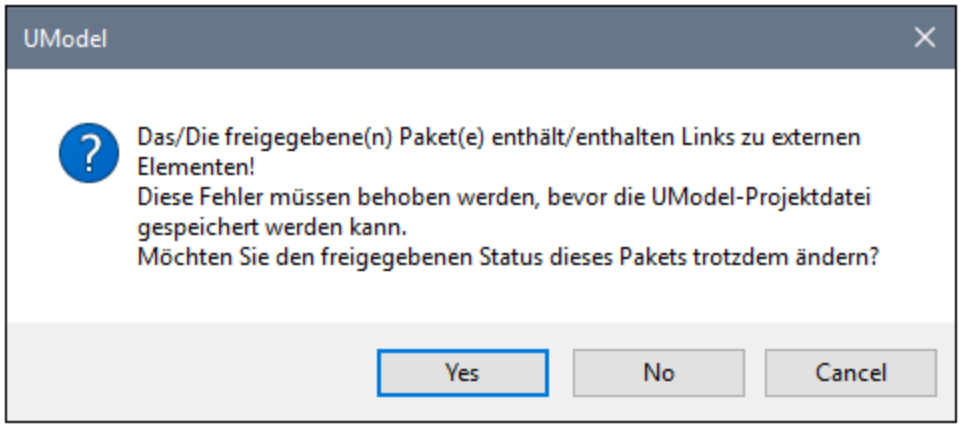


Anmerkungen:

- Wenn Sie ein Quellprojekt inkludieren, das Unterprojekte enthält, werden auch alle Unterprojekte des Quellprojekts in das Zielprojekt inkludiert.
- Freigegebene Ordner, die in Form einer Referenz inkludiert wurden, können jederzeit durch Rechtsklick auf den Ordner und Auswahl von **Unterprojekt | Als Kopie** in "als Kopie" geändert werden.

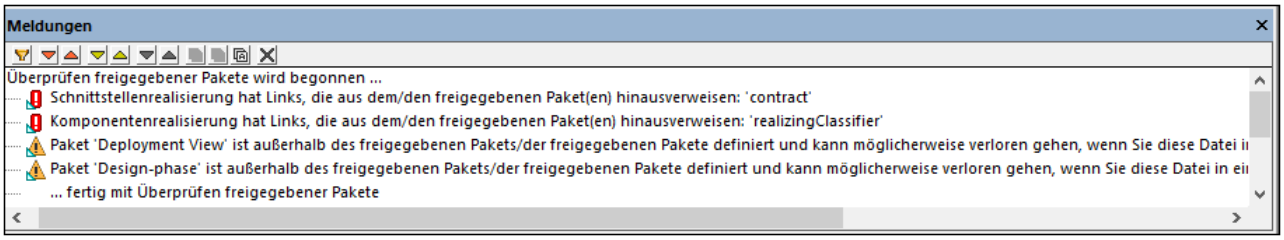
Auflösen von Links zu externen Elementen

Wenn Sie versuchen ein Paket freizugeben, das Links zu externen Elementen aufweist, erscheint ein Dialogfeld mit einer Warnmeldung. So wird etwa die folgende Meldung angezeigt, wenn Sie versuchen, das Paket "Deployment View" des Beispielprojekts **C:\Benutzer\<Benutzername>\Dokumente\Altova\UModel2023\UModelExamples\Tutorial\BankView-start.ump** freizugeben.



Klicken Sie auf **Ja**, um das Paket trotz der Fehler freizugeben. Das Fenster "Meldungen" enthält Informationen über die einzelnen externen Links.

© 2016-2022 Altova GmbH



Klicken Sie im Fenster "Meldungen" auf einen Eintrag, um das entsprechende Element im Fenster "Modell-Struktur" anzuzeigen.

6.1.8 Verbesserung der Performance

Da Modellierungsprojekte manchmal relativ groß werden können, gibt es Möglichkeiten, die Performance beim Modellieren zu verbessern:

- Stellen Sie sicher, dass Sie die neuesten Treiber für Ihre jeweilige Grafikkarte verwenden (tun Sie dies, bevor Sie die folgenden Tipps befolgen)
- Deaktivieren Sie die Syntaxfärbung - **Register "Stile" | Syntaxfarben verwenden** = false.
- Deaktivieren Sie "gradient" als Hintergrundfarbe für Diagramme, verwenden Sie eine einheitliche Farbe (Setzen Sie z.B. im Fenster "Stile" die Eigenschaft **Diag. Hintergrundfarbe** auf eine einheitliche Farbe, z.B. Weiß).
- Die automatisch aktivierte Autokomplettierung kann über **Extras | Optionen | Diagrambearbeitung** und Deaktivieren des Kontrollkästchens **Automatische Eingabehilfe aktivieren** deaktiviert werden.

6.2 Generieren von Programmcode

Nachdem Sie das Modell Ihrer Applikation in UModel erstellt haben (z.B. ein oder mehrere Klassendiagramme), können Sie schnell ein Prototyp-Projekt generieren, das alle definierten Schnittstellen, Klassen, Operationen, usw. in der Sprache Ihrer Wahl enthält. Sie können mit UModel auf Basis der in Ihrem UModel-Projekt gefundenen Elemente (wie z.B. Schnittstellen, Klassen, Operationen, usw.) anhand Ihres Modells C#-, VB.NET- oder Java-Programmcode generieren. Dieser Vorgang wird auch als "Forward Engineering" bezeichnet. Im generierten Code werden alle Objekte genauso generiert, wie sie im Modell definiert wurden, sodass Sie zu ihrer eigentlichen Implementierung übergehen können.

Die Codegenerierung kann auch auf XML-Schemas und Datenbanken angewendet werden*. So könnten Sie etwa ein XML-Schema oder eine Datenbank mit UModel erstellen und anschließend die entsprechende Datei (oder das entsprechende SQL-Skript im Fall von Datenbanken) anhand des Modells generieren. Konsultieren Sie dazu die Zuordnungstabellen, um herauszufinden, welche Schema- oder Datenbankelemente UModel-Elementen zugeordnet werden, siehe UModel-Elementzuordnungen 228.

** Für die Generierung von Datenbanken benötigen Sie die UModel Enterprise oder Professional Edition.*

Voraussetzungen

Damit Programmcode generiert werden kann, muss das UModel-Projekt die folgenden Mindestvoraussetzungen erfüllen:

- Eines der Pakete in Ihrem Projekt muss als Namespace Root definiert sein. Bei der Namespace Root kann es sich um einen C#-, Java-, VB.NET-, XSD- oder Datenbank-Namespace handeln. Dieses Paket muss alle Klassen und Schnittstellen enthalten, anhand welcher der Code generiert werden soll. Nähere Informationen dazu finden Sie unter Definieren eines Pakets als Namespace Root 170.
- Zum Projekt muss eine Code Engineering-Komponente hinzugefügt werden. Diese muss von allen Klassen oder Schnittstellen, für die Code generiert werden soll, realisiert werden. Nähere Informationen dazu finden Sie unter Hinzufügen einer Code Engineering-Komponente 171..

Zusätzlich dazu sollten Sie eines der vordefinierten UModel-Unterprojekte für die entsprechende Sprache (oder Sprachversion) inkludieren, siehe Inkludieren anderer UModel-Projekte 164. Wenn Ihre Applikation z.B. für eine bestimmte Version von C#, Java oder VB.NET geschrieben wird, könnten Sie dadurch beim Erstellen Ihrer UML-Klassen, Schnittstellen, usw. die entsprechenden Datentypen verwenden.

Ein Beispiel zum Erstellen eines Projekts von Grund auf und Generieren von Code anhand des Projekts finden Sie unter Beispiel: Generieren von Java-Code anhand des Modells 182.

6.2.1 Definieren eines Pakets als Namespace Root

Um anhand Ihres UModel-Projekts Programmcode zu generieren, muss ein Paket in Ihrem Modell als Namespace Root definiert werden.

So definieren Sie ein Paket als Namespace Root:

- Klicken Sie mit der rechten Maustaste auf ein Paket im Fenster-Modell-Struktur 80 und wählen Sie im Kontextmenü den Befehl **Code Engineering | Als <...> Namespace Root definieren,** wobei **<...>** für C#, Java, VB.NET, XSD steht.

 © 2016-2022 Altova GmbH

Wenn Sie ein Paket als Namespace Root definieren, informiert Sie UModel darüber, dass auch das UML-Profil der entsprechenden Sprache zum Projekt hinzugefügt und auf das ausgewählte Paket angewendet wird. Klicken Sie auf OK, um dies zu bestätigen, wenn Sie in einem Dialogfeld wie dem unten gezeigten danach gefragt werden.

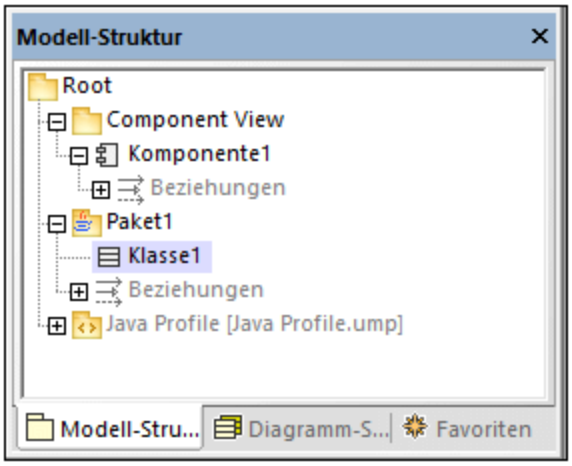
6.2.2 Hinzufügen einer Code Engineering-Komponente

Um Programmcode zu generieren, muss Ihr UModel-Projekt eine Code Engineering-Komponente enthalten, in der alle Einzelheiten zur Codegenerierung definiert sind (z.B. welche Klassen aus dem Projekt bei der Codegenerierung berücksichtigt werden sollen und in welchem Zielverzeichnis der Code gespeichert werden soll). Wie unten gezeigt, muss die Komponente die folgenden Kriterien erfüllen, damit Code generiert werden kann:

- Der Komponente muss ein physischer Ordner zugewiesen werden, in dem der Code generiert wird.
- Die Klassen oder Schnittstellen, die am Code Engineering beteiligt sind, müssen von der Komponente realisiert werden.
- Für die Komponente muss die Eigenschaft **für Code Engineering verwenden** aktiviert sein.

So fügen Sie eine Komponente hinzu, die die gewünschten Klassen oder Schnittstellen realisiert:

1. Klicken Sie in der Modell-Struktur mit der rechten Maustaste auf ein Paket und wählen Sie im Kontextmenü den Befehl **Neues Element | Komponente**. Daraufhin wird die Komponente zum Modell hinzugefügt.
2. Klicken Sie in der Modell-Struktur auf die Klasse oder Schnittstelle, die von der Komponente realisiert werden soll und ziehen Sie diese anschließend mit dem Cursor auf die Komponente (in diesem Beispiel wurde `Klasse1` aus `Paket1` auf `Komponente1` gezogen). Dadurch wird in der Modell-Struktur automatisch eine `Komponentenrealsierungsbeziehung` erstellt.



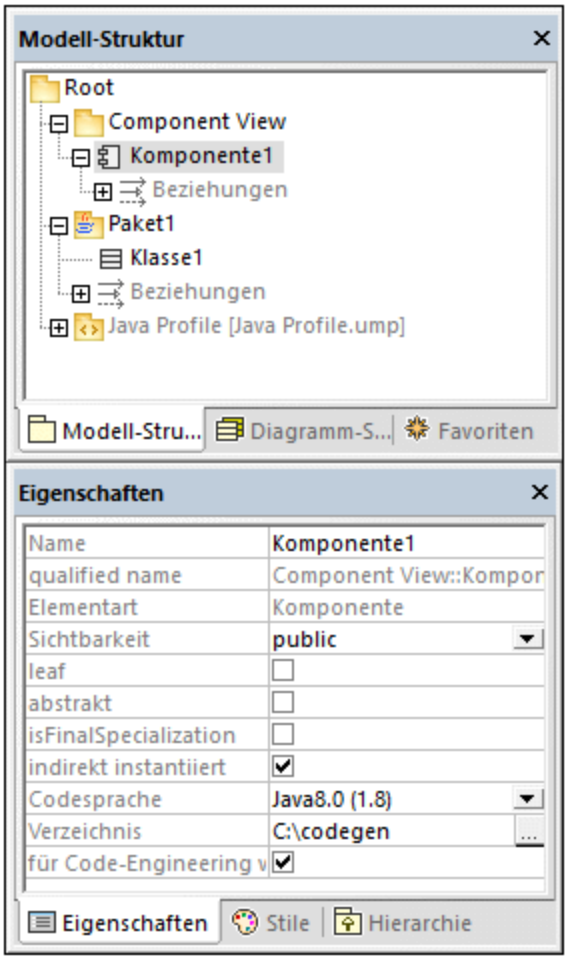
Sie können dies auch auf eine andere Art tun, nämlich indem Sie ein Komponentendiagramm erstellen und anschließend zwischen der Komponente und den Klassen oder Schnittstellen eine `Komponentenrealisierungsbeziehung` ziehen. Nähere Informationen dazu finden Sie unter Komponentendiagramme 48.

So bereiten Sie eine Komponente für das Code Engineering vor:

1. Wählen Sie die Komponente in der Modell-Struktur aus (es wird davon ausgegangen, dass diese Komponente, wie oben erläutert, bereits von mindestens einer Klasse oder Schnittstelle realisiert wird).
2. Suchen Sie im Fenster "Eigenschaften" die Eigenschaft **Verzeichnis** und definieren Sie dafür den Pfad, unter dem Sie Code generieren möchten.
3. Aktivieren Sie im Fenster "Eigenschaften" das Kontrollkästchen **für Code Engineering verwenden**.

So sehen Sie etwa in der Abbildung unten, dass die Komponente **Komponente1** aus dem Paket **Component View** so konfiguriert ist, dass damit Java 8.0 Code im Verzeichnis **C:\codegen**: generiert wird.

 © 2016-2022 Altova GmbH



6.2.3 Überprüfen der Projektsyntax

Es ist wichtig, dass Sie die Syntax des Projekts überprüfen, bevor Sie anhand des Modells Code generieren. Bei der Syntaxüberprüfung werden Sie über alle Probleme, durch die eine Codegenerierung verhindert würde, informiert. Die Projektsyntaxüberprüfung erfolgt über den Menübefehl **Projekt | Projektsyntax überprüfen** (oder drücken Sie alternativ dazu **F11**). Auch bevor Code anhand des Modells aktualisiert wird, wird automatisch eine Syntaxüberprüfung durchgeführt. Die Ergebnisse (Fehler, Warnungen und Informationsmeldungen) werden im Fenster "Meldungen" angezeigt.

Bei der Syntaxüberprüfung wird die Projektdatei, wie aus den Tabellen unten ersichtlich, auf mehreren Ebenen überprüft. Beachten Sie die folgenden Punkte:

- Informationen zur Behebung häufiger Syntaxfehler finden Sie im Kapitel Codegenerierung "Voraussetzungen" 170.
- Die unten aufgelisteten Überprüfungen werden nur dann für Komponenten durchgeführt, wenn für die Komponente im Fenster "Eigenschaften" die Eigenschaft **für Code Engineering verwenden** aktiviert ist.

- Für Klassen, Schnittstellen und Enumerationen werden die unten angeführten Überprüfungen nur durchgeführt, wenn die Klasse, Schnittstelle, Enumeration im Codesprachen-Namespace enthalten ist, d.h. sie muss sich unterhalb eines Pakets befinden, das als Namespace Root definiert wurde.
- Constraints für Modellelemente werden nicht überprüft, da sie nicht Teil der Codegenerierung bilden, siehe Einschränken von Elementen 114.

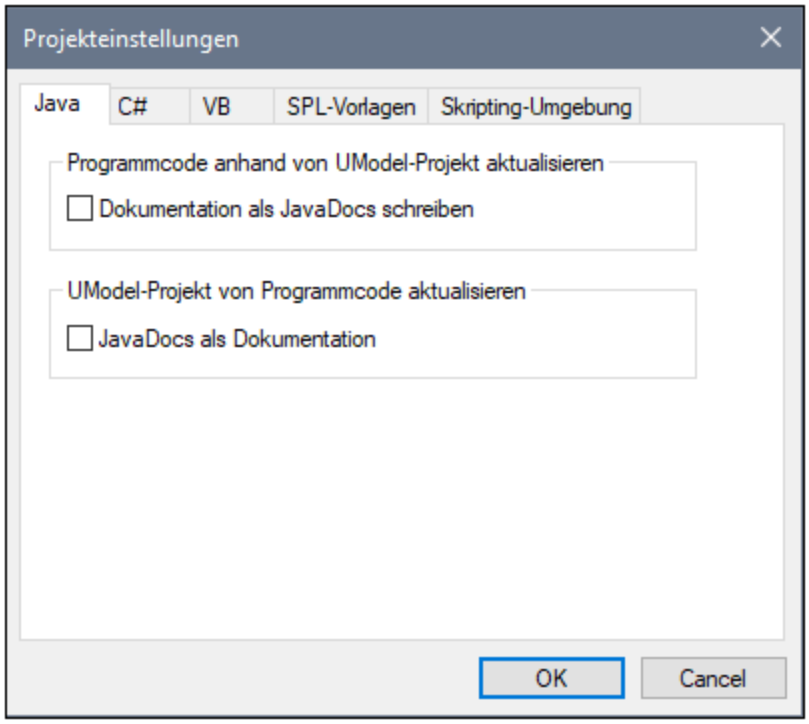
Ebene	Überprüfung	Fehlerschweregrad, falls die Überprüfung fehlschlägt
Projekt	...ob mindestens ein Namespace Root-Paket vorhanden ist.	Fehler
Komponente	...ob die Projektdatei oder das Verzeichnis definiert ist.	Fehler
	...ob diese Komponente eine `Komponentenrealisierungsbeziehung` zu mindestens einer Klasse oder Schnittstelle aufweist.	Fehler
Klasse	...ob der Codedateiname definiert ist. **Anmerkung:** Diese Überprüfung gilt nicht für verschachtelte Klassen.	*Fehler,* wenn die Option **Fehlende Codedateinamen generieren** auf dem Register **Extras | Optionen | Code Engineering** nicht aktiviert wurde. *Warnung,* wenn die Option aktiviert ist.
	...ob für Operationsparameter der Typ definiert ist.	Fehler
	...ob für Eigenschafte der Typ definiert ist.	Fehler
	...ob der Operationsrückgabetyp definiert ist.	Fehler
	...ob Operationen (Namen + Parametertypen) doppelt vorhanden sind.	Fehler
	...ob eine `Komponentenrealisierungsbeziehung` zu einer Komponente besteht. **Anmerkung:** Diese Überprüfung gilt nicht für verschachtelte Klassen.	Warnung
	...ob der Name gültig ist (keine verbotenen Zeichen, Name ist kein Schlüsselwort)	Fehler
	...ob eine Mehrfachvererbung vorkommt	Fehler
Klassenoperation	...ob der Name gültig ist (keine verbotenen Zeichen, Name ist kein Schlüsselwort)	Fehler
	...ob ein Rückgabeparameter vorhanden ist.	Fehler
Klassen-operations-	...ob der Name gültig ist (keine verbotenen Zeichen, Name ist kein Schlüsselwort)	Fehler

 © 2016-2022 Altova GmbH

Ebene	Überprüfung	Fehlerschweregrad, falls die Überprüfung fehlschlägt
parameter	...ob der Typ gültig ist	Fehler
Schnittstelle	...ob der Codedateiname definiert ist.	*Fehler,* wenn die Option **Fehlende Codedateinamen generieren** auf dem Register **Extras | Optionen | Code Engineering** nicht aktiviert wurde. *Warnung,* wenn die Option aktiviert ist.
	..ob die Schnittstelle im Codesprachen-Namespace enthalten ist.	Fehler
	...ob für Eigenschaften der Typ definiert ist.	Fehler
	...ob für Operationsparameter der Typ definiert ist.	Fehler
	...ob der Operationsrückgabetyp definiert ist.	Fehler
	...ob Operationen (Namen + Parametertypen) doppelt vorhanden sind.	Fehler
	...ob Schnittstellen an einer Komponentenrealisierung beteiligt sind.	Warnung
	...ob der Name gültig ist (keine verbotenen Zeichen, Name ist kein Schlüsselwort)	Fehler
Schnittstellen-operation	...ob der Name gültig ist (keine verbotenen Zeichen, Name ist kein Schlüsselwort)	Fehler
Schnittstellen-operations-parameter	...ob der Name gültig ist (keine verbotenen Zeichen, Name ist kein Schlüsselwort)	Fehler
Schnittstellen-eigenschaften	...ob der Name gültig ist (keine verbotenen Zeichen, Name ist kein Schlüsselwort)	Fehler
Paket	...ob der Name gültig ist (keine verbotenen Zeichen, Name ist kein Schlüsselwort) **Anmerkung:** Diese Überprüfung wird angewendet, wenn sich das Paket innerhalb eines Namespace Root-Pakets befindet und darauf über das Fenster "Eigenschaften" der <<namespace>> Stereotyp angewendet wurde.	Fehler
Enumeration	...ob eine `Komponentenrealisierungsbeziehung` zu einer Komponente besteht.	Warnung

6.2.4 Codegenerierungsoptionen

Wenn Sie anhand von Programmcode ein UModel-Projekt generieren, können Sie die unten aufgelisteten Optionen definieren bzw. ändern. Diese Optionen stehen zur Verfügung, wenn Sie den Menübefehl **Projekt | Projekteinstellungen** auswählen. Sie werden zusammen mit dem Projekt gespeichert.



Die Optionen sind folgendermaßen auf Registern gruppiert.

Register	Optionen
Java	Aktivieren Sie das Kontrollkästchen **Dokumentation als JavaDocs schreiben,** um die Dokumentation von UModel-Elementen im generierten Code in die entsprechende Dokumentation für JavaDocs zu konvertieren.
C#	Aktivieren Sie das Kontrollkästchen **Dokumentation als DocComments verfassen,** um die Dokumentation von UModel-Elementen im generierten Code in Kommentare zu konvertieren.
VB	Aktivieren Sie das Kontrollkästchen **Dokumentation als DocComments verfassen,** um die Dokumentation von UModel-Elementen im generierten VB.NET-Code in Kommentare zu konvertieren.
SPL-Vorlagen	Wenn UModel SPL-Vorlagen über einen benutzerdefinierten Pfad auslesen soll, der nicht der Standardpfad ist, muss der benutzerdefinierte Pfad hier eingegeben werden. Siehe auch SPL-Vorlagen 192.

 © 2016-2022 Altova GmbH

Zusätzlich zu den oben angeführten Einstellungen gibt es noch einige weitere Einstellungen, die sich auf die Codegenerierung auswirken. Um diese aufzurufen, wählen Sie den Menübefehl **Extras | Optionen** und klicken Sie auf das Register **Code Engineering**. Die Einstellungen zum Generieren von Code anhand eines Modells befinden sich unter **Programmcode anhand von UModel-Projekt aktualisieren**. Beachten Sie, dass diese Einstellungen lokal sind (sie wirken sich nur auf die aktuelle Installation von UModel aus und werden nicht mit dem Projekt gespeichert).

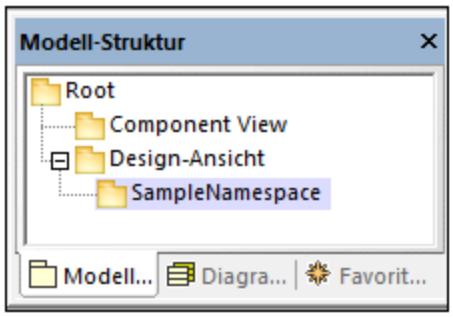
6.2.5 Beispiel: Generieren von C#-Code

In diesem Beispiel wird gezeigt, wie Sie mit UModel C#-Code generieren. Zuerst werden wir einen C#-Beispiel-Namespace, der einige Klassen enthält, erstellen, das Projekt für die Codegenerierung konfigurieren und anschließend den eigentlichen Code generieren.

Die Zielplattform in diesem Beispiel ist *.NET Standard 2.0 für C# 7.1*. Dies ist dank eines in UModel vordefinierten Profils, in dem alle Typen des *.NET Standard 2.0 für C# 7.1* vordefiniert sind, möglich. UModel enthält auch vordefinierte Profile für bestimmte .NET Framework-Versionen, siehe auch Inkludieren von Unterprojekten 164.

Erstellen eines neue Projekts und seiner Struktur

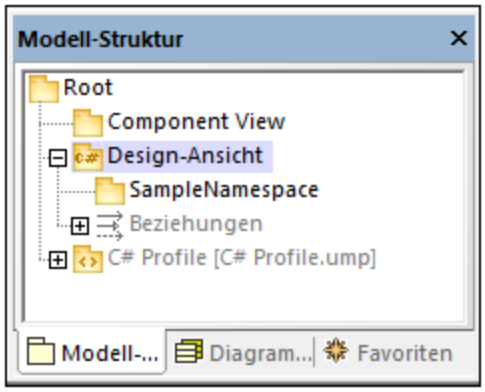
Im ersten Schritt wird ein leeres Projekt erstellt, das zwei Standardpakete hat (`Root` und `Component View`): Klicken Sie im Menü **Datei** oder in der Symbolleiste auf **Neu**. Klicken Sie als nächstes auf das `Root`-Paket und erstellen Sie einige weitere Pakete, wie unten gezeigt. Wenn Sie mit der grafischen Benutzeroberfläche von UModel noch nicht vertraut sind, lesen Sie zuerst die Kapitel UModel Tutorial 14 und Anleitung zur Modellierung von.... 104.



In diesem Beispiel dient das Paket `Design-Ansicht` als Container für den Design-Teil Ihres Modells (z.B. Klassen und Klassendiagramme), während das Paket `SampleNamespace` als Namespace für alle Klassen, die erstellt werden sollen, verwendet wird. Sie können Ihre Pakete bei Bedarf auch anders strukturieren.

Code Engineering

Im nächsten Schritt wird C# für Ihr Paket definiert. Klicken Sie mit der rechten Maustaste auf das Paket `Design-Ansicht` und wählen Sie im Kontextmenü den Befehl **Code Engineering | Als C# Namespace Root definieren**. UModel informiert Sie darüber, dass das C#-Profil auf das Paket angewendet wird. Klicken Sie auf **OK.** Das in UModel vordefinierte C#-Profil wird nun in das Projekt inkludiert (*siehe Abbildung unten*).

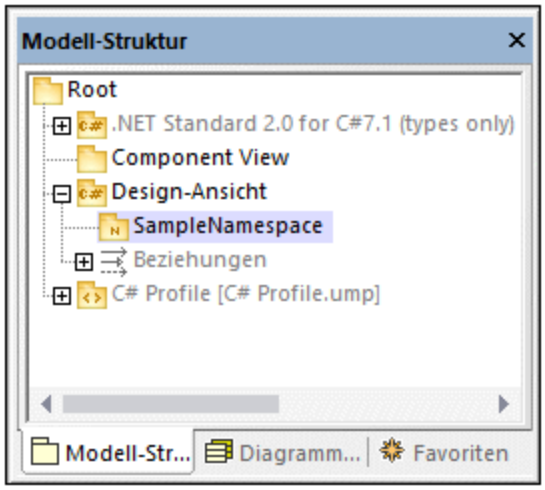


Definieren von SampleNamespace als Namespace
Klicken Sie als nächstes auf das Paket `SampleNamespace` und aktivieren Sie im Fenster **Eigenschaften** das Kontrollkästchen `<<namespace>>`. Dadurch wird das `namespace`-Stereotyp auf das Paket angewendet und sein Symbol ändert sich in . Sie können jetzt unter diesem Namespace Klassen erstellen.

Inkludieren eines Unterprojekts
Das Modell enthält bisher das C#-Profil, das die Datentypen für C# enthält. Allerdings enthält es die .NET Standard 2.0-spezifischen Datentypen noch nicht (diese stehen in einem separaten UModel-Profil zur Verfügung). Um dieses Profil zum Projekt hinzuzufügen, gehen Sie folgendermaßen vor:

1. Klicken Sie im Menü **Projekt** auf **Unterprojekt inkludieren**.
2. Wechseln Sie zum Register **C#** und wählen Sie den Eintrag *.NET Standard 2.0 for C# 7.1 (types only)* aus.
3. Wählen Sie im Dialogfeld **Unterprojekt inkludieren** *Durch Referenz* aus und klicken Sie auf **OK**.

Das zusätzliche Profil wurde nun zum Projekt hinzugefügt (*siehe unten*).



 © 2016-2022 Altova GmbH

Erstellen von C#-Klassen

Im nächsten Schritt werden Klassen erstellt. Sie können dies entweder direkt über das Fenster **Modell-Struktur** oder über ein Klassendiagramm tun. In diesem Beispiel haben wir uns für die zweite Option entschieden. Gehen Sie folgendermaßen vor:

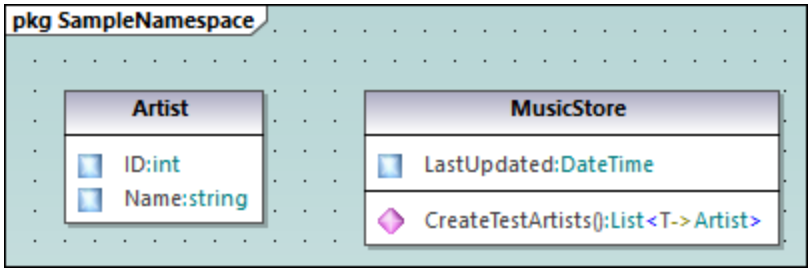
1. Öffnen Sie das Fenster **Diagramm-Struktur**.
2. Klicken Sie mit der rechten Maustaste auf **Klassendiagramme** und wählen Sie **Neues Diagramm | Klassendiagramm**.

In diesem Beispiel wird vorausgesetzt, dass alle Ihre Klassen unter dem Namespace `SampleNamespace` generiert werden müssen. Wenn Sie daher aufgefordert werden, einen Owner für das Diagramm auszuwählen, wählen Sie das Paket `SampleNamespace` aus. Wenn Sie ein anderes Paket auswählen, gehören alle Elemente, die Sie zum Diagramm hinzufügen zum selben Paket wie das Diagramm (was nicht notwendigerweise beabsichtigt ist).

Erstellen von Klassen und ihrer Struktur

Erstellen Sie als nächstes Klassen, Typen und andere in Ihrem Modell erforderlichen Elemente. Sie können für unser Beispiel ein einfaches Diagramm, das eine `Artist`- und eine `MusicStore`-Klasse enthält, erstellen (*siehe Abbildung unten*). Gehen Sie folgendermaßen vor:

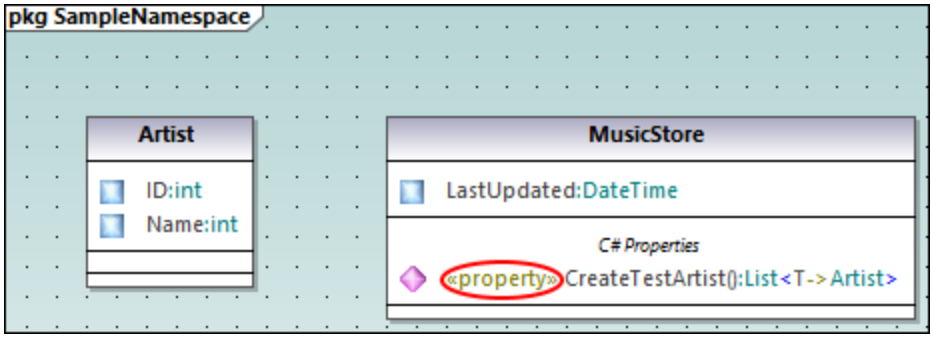
1. Klicken Sie mit der rechten Maustaste in das *pkg SampleNamespace*-Fenster und wählen Sie **Neu | Klasse**.
2. Geben Sie dieser Klasse den Namen `Artist`.
3. Klicken Sie mit der rechten Maustaste in das Kästchen `Artist` und erstellen Sie zwei Eigenschaften: `ID` von Typ `int` und `Name` vom Typ `string`.
4. Erstellen Sie die zweite Klasse mit dem Namen `MusicStore`.
5. Erstellen Sie eine Eigenschaft namens `LastUpdated` vom Typ `DateTime`.
6. Erstellen Sie eine Operation und geben Sie ihren Namen und ihre Definition, wie unten gezeigt, ein.

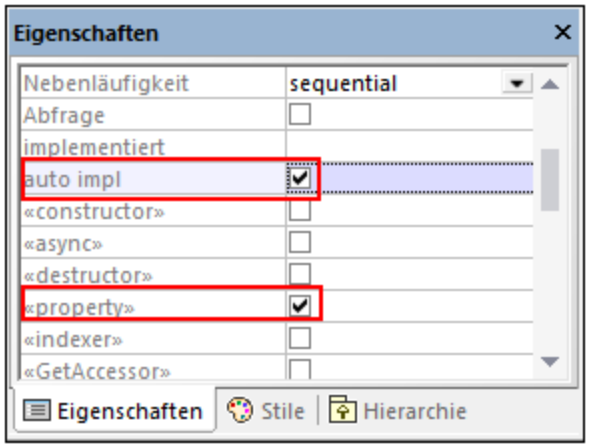


Nähere Informationen zum Erstellen von Klassen und ihren Mitgliedern finden Sie in den Kapiteln Klassendiagramme 25 und Anleitung zur Modellierung von... 104.

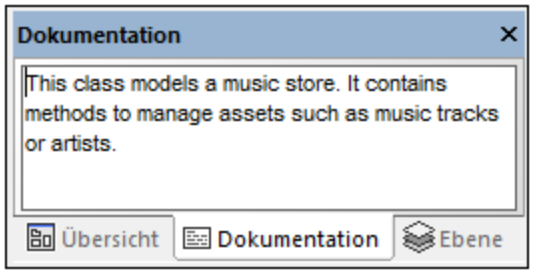
Über automatisch implementierte C#-Eigenschaften

Sie sehen in UModel, ob C#-Eigenschaften automatisch implementiert wurden. Die Autoimplementierungsoption steht zur Verfügung, nachdem im Fenster **Eigenschaften** das Kontrollkästchen `property` (in unserem Beispiel für `CreateTestArtist()`) aktiviert wurde (*siehe Abbildungen unten*).





Hinzufügen von Dokumentation (optional)
Klicken Sie optional im Diagramm auf die Klasse `MusicStore` und fügen Sie durch Eingabe von Text in das Fenster "Dokumentation" 91 Dokumentation hinzu (*siehe Abbildung unten*). Auf diese Art können Sie Codekommentare zu dieser Klasse generieren.



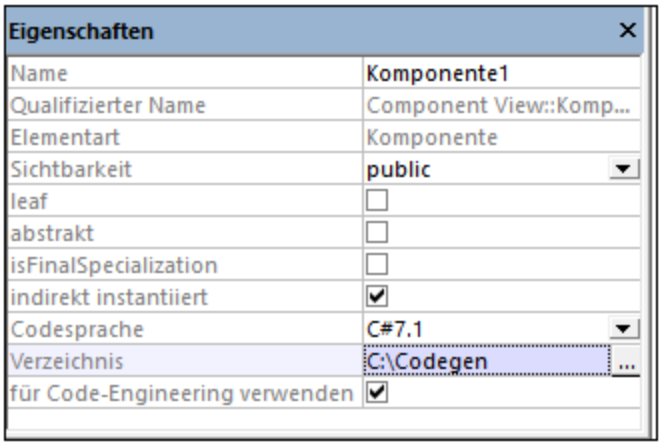
Konfigurieren des Projekts für das Code Engineering

Im nächste Schritt müssen wir nun Code Engineering-Einstellungen definieren. Gehen Sie folgendermaßen vor:

1. Speichern Sie das Projekt in einem Verzeichnis.
2. Klicken Sie anschließend im **Fenster Modell-Struktur** mit der rechten Maustaste auf das Paket `Component View` und fügen Sie eine neue **Komponente** (d.h. eine Software-Komponente) hinzu.

3. Klicken Sie auf die neue Software-Komponente und definieren Sie im Fenster **Eigenschaften** die folgenden Eigenschaften (*siehe Abbildung unten*):

   - Setzen Sie die Codesprache der Komponente z.B. auf C# 7.1.
   - Wählen Sie das Verzeichnis für die Codegenerierung aus (`c:\codegen` in unserem Beispiel).
   - Aktivieren Sie das Kontrollkästchen *für Code Engineering verwenden*.



Erstellen einer Komponentenrealisierungsbeziehung

Erstellen Sie als nächstes zwischen den Klassen, anhand welcher C#-Code generiert werden muss, eine `Komponentenrealisierungsbeziehung`. Sie können dies folgendermaßen tun: Klicken Sie im Fenster **Modell-Struktur** auf die Klasse, die von der Komponente realisiert werden soll (in diesem Beispiel `Artist`) und ziehen Sie sie mit der Maus in die Code Engineering-Komponente (`Komponente1`) (*siehe Abbildung unten*). Führen Sie denselben Schritt auch für die Klasse `MusicStore` durch.

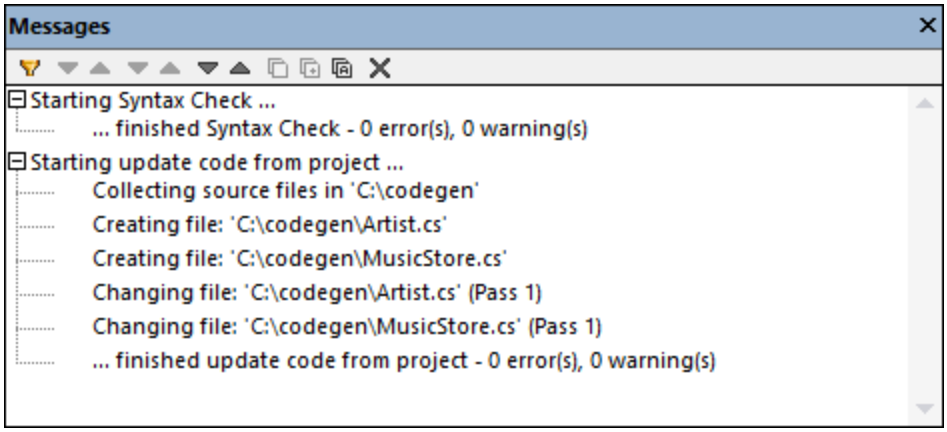
Anmerkung: Falls Sie vergessen haben, für eine Klasse eine `Komponentenrealisierungsbeziehung` zu erstellen, generiert UModel dennoch die entsprechende Codedatei, gibt im Fenster **Meldungen** aber Warnungen aus. Diese Einstellung kann über **Extras | Optionen | Register Code Engineering** konfiguriert werden (Kontrollkästchen *Fehlende Komponentenrealisierungen generieren*).

Generieren von C#-Code

Im letzten Schritt wird nun der eigentliche C#-Code generiert. Gehen Sie folgendermaßen vor:

1. Klicken Sie im Menü **Projekt** auf **Merge Programmcode aus UModel-Projekt**. Daraufhin wird ein Dialogfeld angezeigt, in dem Sie einstellen können, ob Änderungen im Modell mit denjenigen im Code zusammengeführt oder diese gegebenenfalls überschrieben werden sollen. Wählen Sie in diesem Beispiel **...überschreiben**, da ein neues Projekt generiert wird.
2. Damit die Klassendokumentation im generierten Code in Form von Kommentaren inkludiert wird, klicken Sie auf **Projekt | Projekteinstellungen** und aktivieren Sie das Kontrollkästchen **Dokumentation als DocComments verfassen**. Nähere Informationen dazu finden Sie unter Codegenerierungsoptionen 176.

3. Klicken Sie auf **OK.** Das Ergebnis des Code Engineering wird im Fenster **Meldungen** angezeigt (*siehe unten*).



Wenn Sie zur Klasse `MusicStore` Dokumentation hinzugefügt haben, sehen Sie diese im generierten Code in Form von Codekommentaren:

```
using System;
using System.Collections.Generic;
namespace SampleNamespace
{
   /// This class models a music store. It contains methods to manage assets such as
music tracks or artists.
   public class MusicStore
   {
      public DateTime LastUpdated;
      public List<Artist> CreateTestArtists()
      {
         // TODO add implementation
      }
   }

}
```

6.2.6 Beispiel: Generieren von Java-Code

In diesem Beispiel wird gezeigt, wie Sie ein neues UModel-Projekt erstellen und Programmcode anhand dieses Projekts generieren (ein Prozess, der als "Forward Engineering" bezeichnet wird). Der Einfachheit halber besteht das Projekt aus nur einer Klasse. Außerdem lernen Sie, wie Sie das Projekt für die Codegenerierung vorbereiten und sicher stellen, dass im Projekt die richtige Syntax verwendet wird. Nachdem Sie Programmcode generiert haben, werden Sie diesen außerhalb von UModel ändern, indem Sie eine neue Methode zur Klasse hinzufügen. Zum Abschluss werden Sie die Codeänderungen wieder im ursprünglichen UModel-Projekt zusammenführen (ein als "Reverse Engineering" bekannter Prozess).

In diesem Tutorial wird als Codegenerierungssprache Java verwendet. Die Vorgangsweise ist aber auch bei anderen Codegenerierungssprachen ähnlich.

 © 2016-2022 Altova GmbH

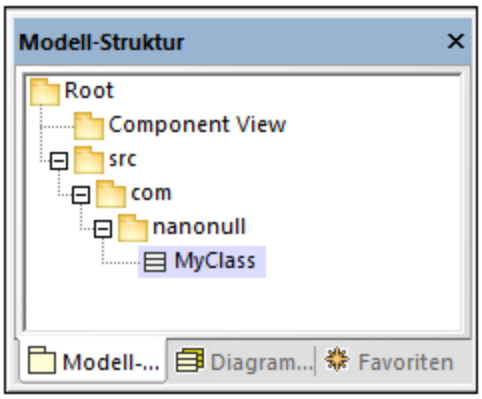
Erstellen eines neue UModel-Projekts

Sie können folgendermaßen ein neues UModel-Projekt erstellen:

- Klicken Sie im Menü **Datei** auf **Neu**. (Oder drücken Sie alternativ dazu **Strg+N** oder klicken Sie auf die Symbolleisten-Schaltfläche "Neu" .)

Das Projekt enthält zu diesem Zeitpunkt nur die Standardpakete "Root" und "Component View". Diese beiden Pakete können nicht gelöscht oder umbenannt werden. "Root" bildet die oberste Hierarchieebene für alle anderen Pakete und Elemente im Projekt. "Component View" wird für das Code Engineering benötigt; normalerweise enthält es eine oder mehrere UML-Komponenten, die von den Klassen oder Schnittstellen Ihres Projekts realisiert werden; wir haben zu diesem Zeitpunkt jedoch noch keine Klassen erstellt. Erstellen wir also zuerst die Struktur unseres Programms. Gehen Sie dazu folgendermaßen vor:

1. Klicken Sie mit der rechten Maustaste im Fenster "Modell-Struktur" auf das Paket "Root" und wählen Sie im Kontextmenü den Befehl **Neues Element | Paket**. Benennen Sie das neue Paket in "src" um.
2. Klicken Sie mit der rechten Maustaste auf "src" und wählen Sie im Kontextmenü den Befehl **Neues Element | Paket.** Benennen Sie das neue Paket in "com" um.
3. Klicken Sie mit der rechten Maustaste auf "com" wählen Sie im Kontextmenü den Befehl **Neues Element | Paket**. Benennen Sie das neue Paket in "nanonull" um.
4. Klicken Sie mit der rechten Maustaste auf "nanonull" wählen Sie im Kontextmenü den Befehl **Neues Element | Klasse**. Benennen Sie die neue Klasse in "MyClass" um.



Vorbereiten des Projekts für die Codegenerierung

Um Code anhand eines UModel-Modells zu generieren, müssen die folgenden Voraussetzungen gegeben sein:

- Es muss ein C#- oder VB.NET-Namespace-Root-Pakte definiert sein.
- Es muss eine Komponente geben, die von allen Klassen oder Schnittstellen, für die Code generiert werden muss, realisiert wird.
- Der Komponente muss ein physischer Pfad (Verzeichnis) zugewiesen worden sein. Der Code wird in diesem Verzeichnis generiert.
- Für die Komponente muss die Eigenschaft **für Code Engineering verwenden** aktiviert sein.

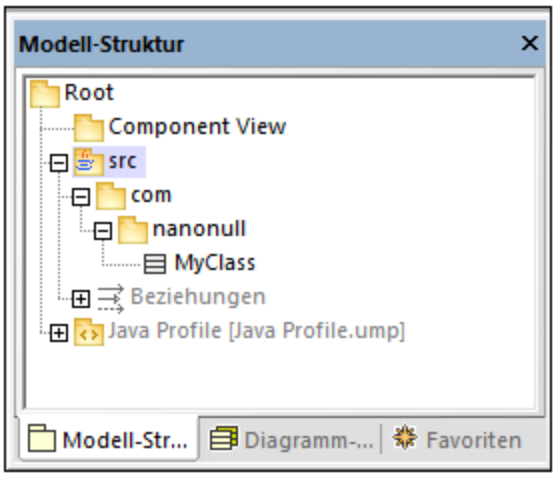
Alle diese Anforderungen werden weiter unten ausführlicher beschrieben. Beachten Sie, dass Sie jederzeit überprüfen können, ob das Projekt alle Voraussetzungen für die Codegenerierung erfüllt, indem Sie es validieren:

- Klicken Sie im Menü **Projekt** auf **Projektsyntax überprüfen**. (Drücken Sie alternativ dazu **F11**.)

Wenn Sie das Projekt in dieser Phase validieren, wird im Fenster "Meldungen" ein Validierungsfehler angezeigt ("*Es wurde keine Namespace Root gefunden! Verwenden Sie bitte das Kontextmenü in der Modell-Struktur, um ein Paket als Namespace Root zu definieren*"). Um diesen Fehler zu beheben, weisen Sie das Paket "src" als Namespace Root zu:

- Klicken Sie mit der rechten Maustaste auf das Paket "src" und wählen Sie im Kontextmenü den Befehl **Code Engineering | Als Java Namespace Root definieren**.
- Wenn Sie informiert werden, dass das UModel Java-Profil inkludiert wird, klicken Sie auf **OK**.

Beachten Sie, dass sich das Paketsymbol nun in das Symbol geändert hat, was bedeutet, dass es sich beim Paket um eine Java Namespace Root handelt. Zusätzlich dazu wurde ein Java-Profil zum Projekt hinzugefügt.



Der eigentliche Namespace kann folgendermaßen definiert werden:

1. Wählen Sie das Paket "com" im Fenster **Modell-Struktur** aus.
2. Aktivieren Sie im Fenster **Eigenschaften** die Eigenschaft **<<namespace>>**.

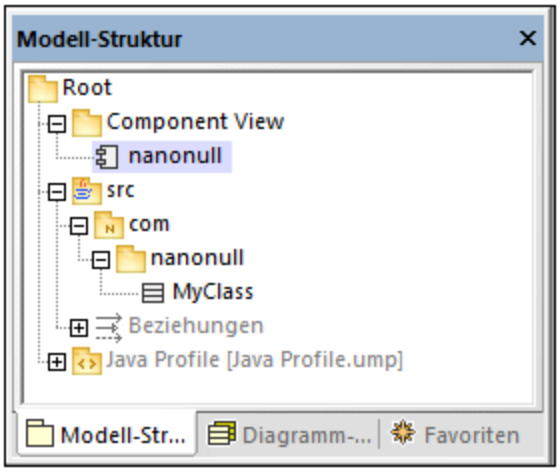
 © 2016-2022 Altova GmbH

3. Wiederholen Sie die obigen Schritte für das Paket "nanonull".

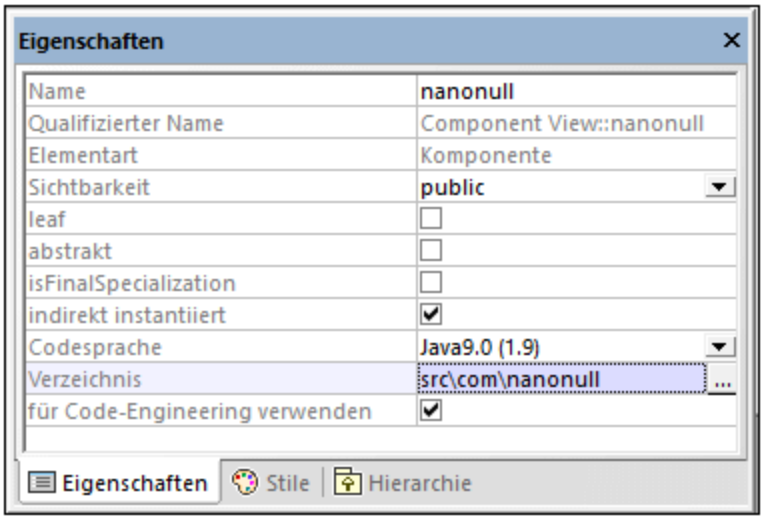
Beachten Sie, dass sich das Symbol der beiden Pakete "com" und "nanonull" nun in geändert hat, was bedeutet, dass es sich nun um einen Namespace handelt.

Eine weitere Voraussetzung für die Codegenerierung ist, dass eine Komponente von mindestens einer Klasse oder Schnittstelle realisiert werden muss. Eine Komponente ist in UML ein Teilstück des Systems. In UModel können Sie über die Komponente das Verzeichnis für die Codegenerierung und andere Einstellungen definieren. Andernfalls wäre die Codegenerierung nicht möglich. Wenn Sie das Projekt zu diesem Zeitpunkt validieren, wird im Fenster **Meldungen** eine Warnung angezeigt: "*MyClass hat keine Komponentenrealisierung zu einer Komponente - es wird kein Code generiert*". Um diesen Fehler zu beheben, muss eine Komponente zum Projekt hinzugefügt werden. Gehen Sie folgendermaßen vor:

1. Klicken Sie mit der rechten Maustaste im Fenster "Modell-Struktur" auf "Component View" und wählen Sie im Kontextmenü den Befehl **Neues Element | Komponente**.
2. Benennen Sie die neue Komponente in "nanonull" um.



3. Ändern Sie im Fenster **Eigenschaften** die Eigenschaft **Verzeichnis** in ein Verzeichnis, in dem der Code generiert werden soll (in diesem Beispiel, "src\com\nanonull"). Beachten Sie dass die Eigenschaft **für Code Engineering verwenden** aktiviert ist, was eine weitere Vorbedingung für die Codegenerierung ist.



4. Speichern Sie das UModel-Projekt in einem Verzeichnis und geben Sie ihm einen beschreibenden Namen (in diesem Beispiel **C:\UModelDemo\Tutorial.ump**).

Anmerkung: Der Pfad für die Codegenerierung kann absolut oder relativ zum .ump-Projekt sein. Wenn er, wie in diesem Beispiel, relativ ist, würden mit dem Pfad **src\com\nanonull** alle Verzeichnisse im selben Verzeichnis, in dem auch das UModel-Projekt gespeichert ist, erstellt werden.

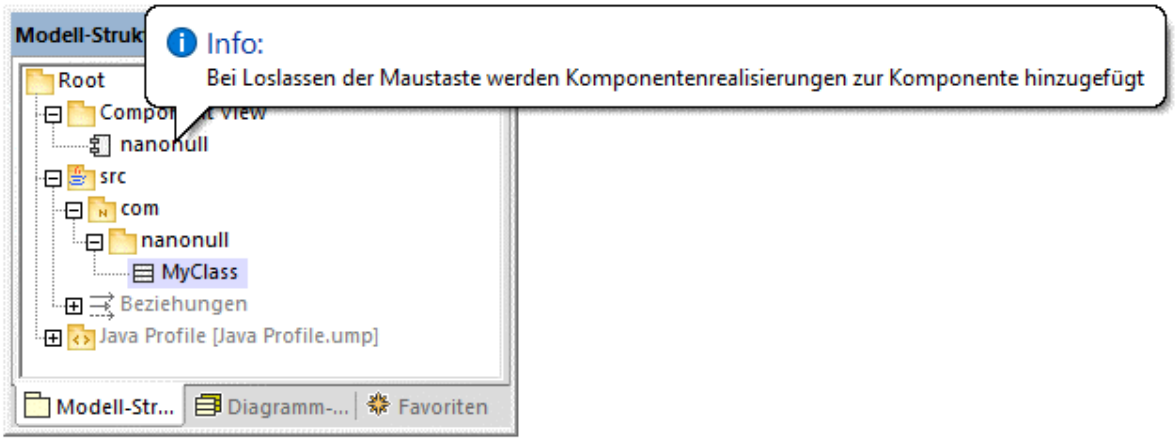
Wir haben uns absichtlich entschieden, Code in einem Verzeichnis zu generieren, das den Namespace-Namen enthält; andernfalls würde es zu Warnungen kommen. UModel zeigt standardmäßig Projektvalidierungswarnungen an, wenn die Komponente so konfiguriert ist, das Java-Code in einem Verzeichnis generiert wird, das nicht denselben Namen wie der Namespace hat. Die Komponente "nanonull" in diesem Beispiel hat den Pfad "C:\UModelDemo\src\com\nanonull", sodass es zu keinen Validierungswarnungen kommt. Wenn Sie eine ähnliche Überprüfung für C# oder VB.NET implementieren möchten oder wenn Sie die Namespace-Validierung für Java deaktivieren möchten, gehen Sie folgendermaßen vor:

1. Klicken Sie im Menü **Extras** auf **Optionen**.
2. Klicken Sie auf das Register **Code Engineering**.
3. Aktivieren Sie unter **Namespace für Codedateipfad verwenden** das entsprechende Kontrollkästchen.

Die Komponentenrealisierungsbeziehung kann folgendermaßen erstellt werden:

- Ziehen Sie die Maus im Fenster Modell-Struktur bei gedrückter Maustaste von der zuvor erstellten Klasse `MyClass` auf die Komponente `nanonull`.

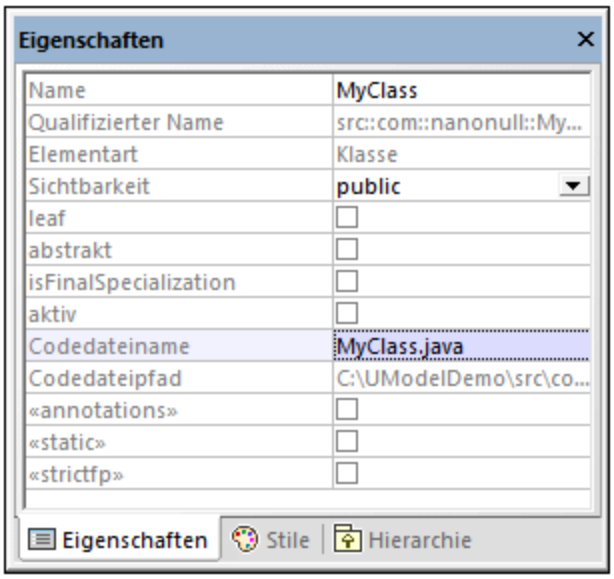
 © 2016-2022 Altova GmbH



Die Komponente wird nun von der einzigen Klasse des Projekts, nämlich `MyClass` realisiert. Beachten Sie, dass die oben beschriebene Methode nun eine von mehreren Möglichkeiten ist, die Komponentenrealisierung zu erstellen. Eine weitere Methode ist, sie, wie im Tutorialabschnitt Komponentendiagramme 48 beschrieben, von einem Komponentendiagramm aus zu erstellen.

Als nächstes wird empfohlen, den an der Codegenerierung beteiligten Klassen oder Schnittstellen einen Dateinamen zu geben. Andernfalls generiert UModel die entsprechende Datei mit einem Standarddateinamen und im Fenster **Meldungen** wird eine Warnung angezeigt ("*'Codedateiname wurde nicht definiert - es wird ein Standardname generiert.*"). Um diese Warnung zu entfernen, gehen Sie folgendermaßen vor:

1. Wählen Sie die Klasse `MyClass` im Fenster **Modell-Struktur** aus.
2. Ändern Sie die Eigenschaft **Codedateiname** im Fenster **Eigenschaften** in den gewünschten Datennamen (in diesem Beispiel, `MyClass.java`).

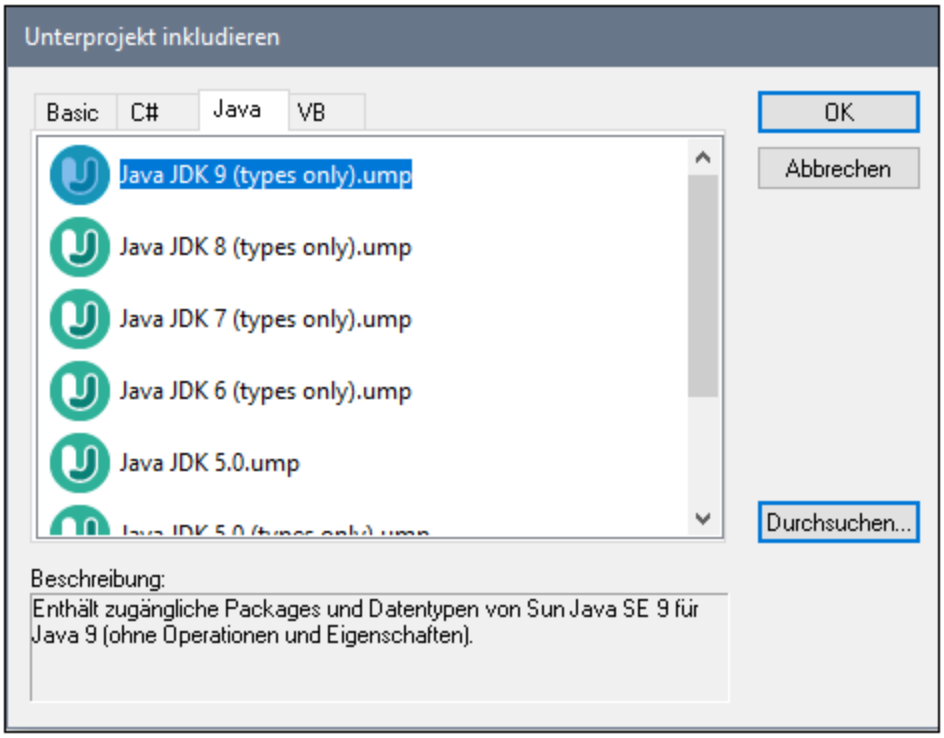


Inkludieren der JDK-Typen

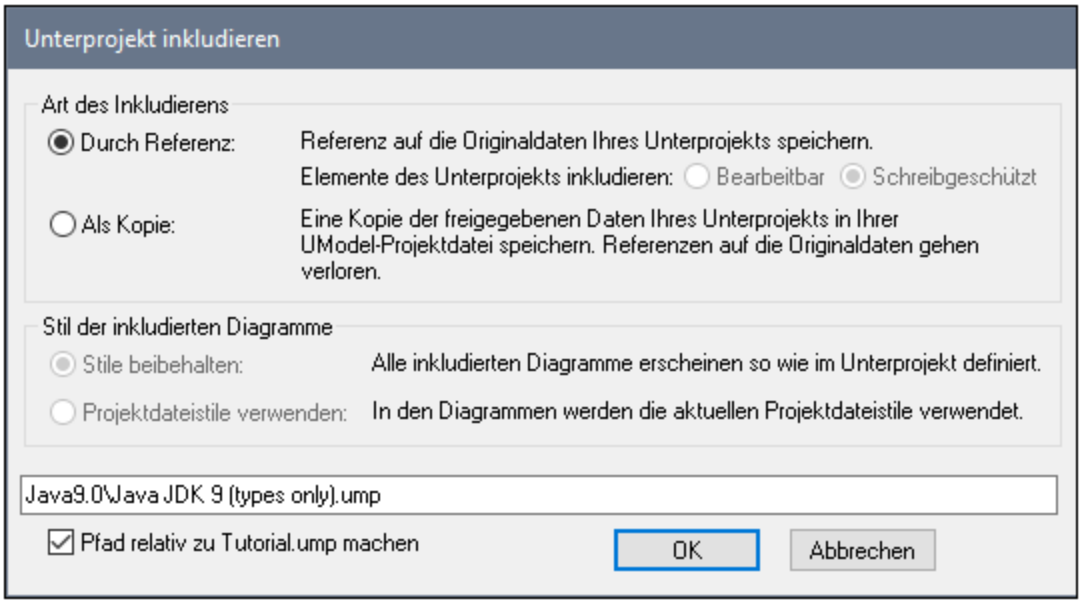
Dieser Schritt ist zwar optional, dennoch wird empfohlen, die Java Development Kit (JDK)-Sprachentypen als

Unterprojekt Ihres aktuellen UModel-Projekts zu inkludieren. Andernfalls stehen die JDK-Typen beim Erstellen von Klassen oder Schnittstellen nicht zur Verfügung. Gehen Sie dazu folgendermaßen vor (die Vorgangsweise ist für C# und VB.NET ähnlich):

1. Klicken Sie im Menü **Projekt** auf **Unterprojekt inkludieren**.
2. Klicken Sie auf das Register **Java** und wählen Sie das Projekt **Java JDK 9 (types only)** aus.



3. Wenn Sie gefragt werden, ob das Projekt über eine Referenz oder als Kopie inkludiert werden soll, wählen Sie **Durch Referenz**.

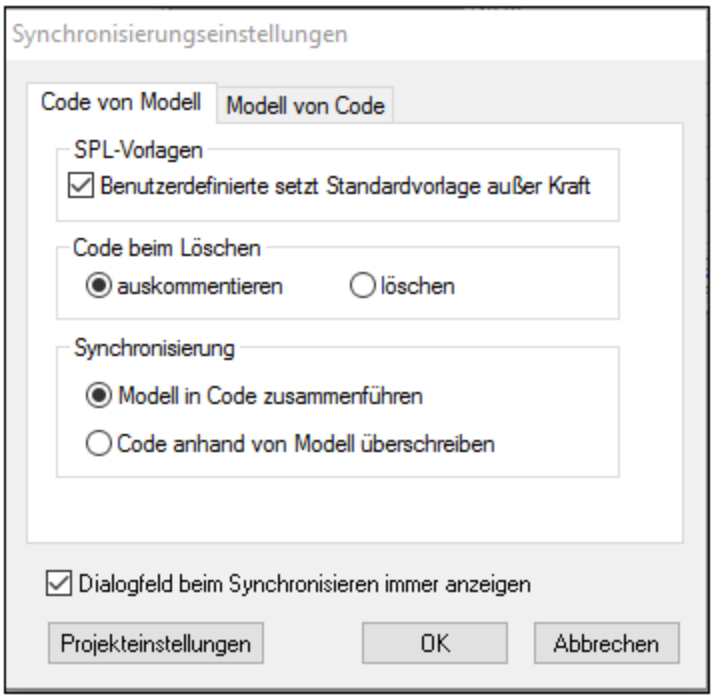


 © 2016-2022 Altova GmbH

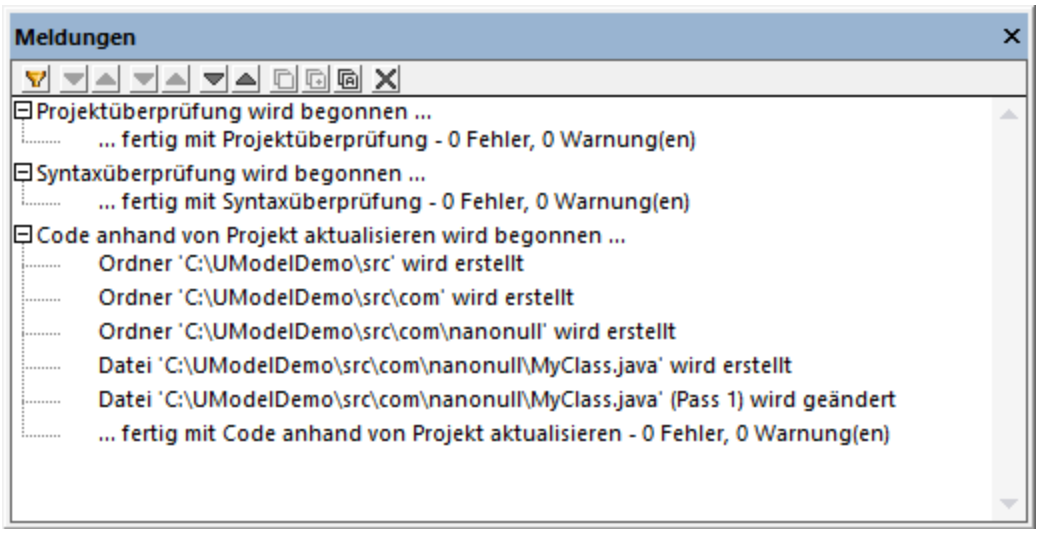
Generieren von Code

Nachdem nun alle Vorbedingungen erfüllt werden, kann Code folgendermaßen generiert werden:

1. Klicken Sie im Menü **Projekt** auf **Merge Programmcode aus UModel-Projekt**. (Drücken Sie alternativ dazu **F12**). Beachten Sie, dass dieser Befehl den Namen **Überschreibe Programmcode aus UModel-Projekt** hat, wenn zuvor im unten gezeigten Dialogfeld "Synchronisierungseinstellungen" die Option **Code anhand von Modell überschreiben** aktiviert wurde.



2. Lassen Sie die Standardsynchronisierungseinstellungen unverändert und klicken Sie auf **OK**. Es wird automatisch eine Projektsyntaxüberprüfung durchgeführt, deren Ergebnis im Fenster **Meldungen** angezeigt wird:



Ändern von Code außerhalb von UModel

Die Generierung von Programmcode ist erst der erste Schritt bei der Entwicklung Ihrer Software-Applikation oder Ihres Systems. In einem realen Szenario würde der Code mehrmals verändert, bevor das ein Programm mit allen seinen Features fertig entwickelt ist. Öffnen Sie die generierte Datei **MyClass.java** zu diesem Zweck in einem Text-Editor und fügen Sie eine neue Methode zur Klasse hinzu, wie unten gezeigt. Die Datei **MyClass.java** sollte nun folgendermaßen aussehen:

```
package com.nanonull;
public class MyClass{
   public float sum(float num1, float num2){
    return num1 + num2;
  }
}
```

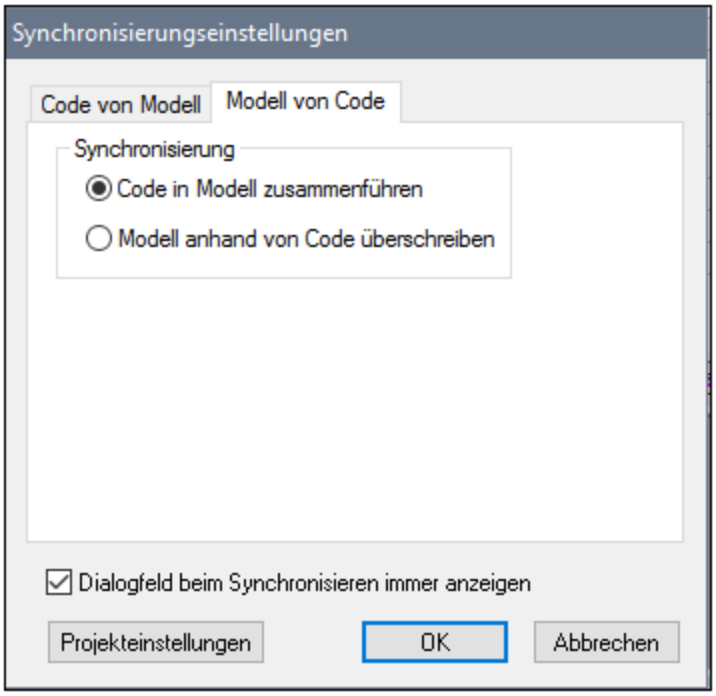
MyClass.java

Zusammenführen der Codeänderungen im Modell

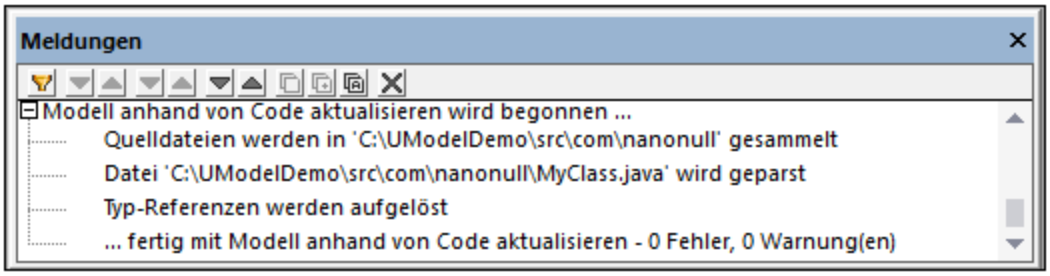
Sie können den geänderten Code nun wieder im Modell zusammenführen. Gehen Sie folgendermaßen vor:

1. Klicken Sie im Menü **Projekt** auf **Merge UModel-Projekt aus Programmcode** (Drücken Sie alternativ dazu **Ctrl + F12**).

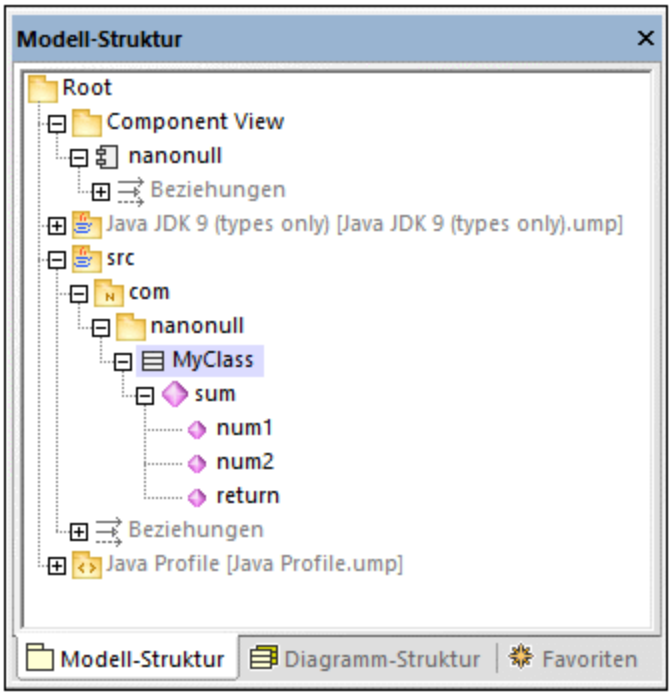
 © 2016-2022 Altova GmbH



2. Lassen Sie die Standardsynchronisierungseinstellungen unverändert und klicken Sie auf **OK**. Es wird automatisch eine Codesyntaxüberprüfung durchgeführt, deren Ergebnis im Fenster **Meldungen** angezeigt wird:



Die Operation `sum` (die mit Reverse Engineering anhand des Codes generiert wurde) wird nun im Fenster "Modell-Struktur" angezeigt.



6.2.7 SPL-Vorlagen

Bei der Generierung von Java-, C#- oder VB.NET-Code sowie von XSD-Schemas verwendet UModel eine Vorlagensprache namens SPL (Spy Programming Language). Die SPL-Vorlagen geben die Syntax der generierten Codedateien vor. Die SPL-Vorlagen können angepasst werden, z.B. um die Syntax des generierten Codes geringfügig zu modifizieren. Das Bearbeiten von SPL-Vorlagen ist nur bei Codesprachen sinnvoll, die von UModel unterstützt werden. Wenn Sie völlig neue SPL-Vorlagen für andere Sprachen erstellen möchten, wäre es zwar möglich neuen Code zu generieren, doch wäre es nicht möglich, vorhandenen Code zu aktualisieren (da die Sprachsyntax UModel nicht bekannt wäre).

Die SPL-Standardvorlagen befinden sich relativ zum Programminstallationsverzeichnis im Verzeichnis **UModelSPL**.

> Ändern Sie die vorhandenen SPL-Standardvorlagen nicht, da diese sich direkt auf die Standardcodegenerierung auswirken. Falls Sie die Codegenerierung anpassen müssen, erstellen Sie stattdessen, wie unten beschrieben, benutzerdefinierte Vorlagen.

SPL-Vorlagen werden nur verwendet, wenn neuer Code generiert wird (d.h. wenn neue Klassen, Operationen, usw. zum Modell hinzugefügt wurden und anschließend Code generiert wird). Vorhandener Code wird von SPL-Vorlagen nicht beeinflusst.

Eine Einführung in SPL finden Sie unter SPL-Referenz 543.

 © 2016-2022 Altova GmbH

So ändern Sie bereitgestellte SPL-Vorlagen:

1. Gehen Sie im UModel-Installationsverzeichnis ("Programme") zu den bereitgestellten SPL-Vorlagen, z.B.: **...\UModel2023\UModelSPL\Java\Default**.
2. Kopieren Sie die SPL-Dateien, die Sie ändern möchten, in das übergeordnete Verzeichnis. Wenn Sie z.B. das Auftreten der Java-Klasse im generierten Code ändern möchten, kopieren Sie die Datei **Class.spl** aus **...\UModel2023\UModelSPL\Java\Default** in **...\UModel2023\UModelSPL\Java**.
3. Nehmen Sie die Änderungen an der/den .spl-Datei(en) vor und speichern Sie diese.

So verwenden Sie die benutzerdefinierten SPL-Vorlagen:

1. Wählen Sie die Menüoption **Projekt | Synchronisierungseinstellungen**.
2. Aktivieren Sie in der Gruppe "SPL-Vorlagen" das Kontrollkästchen **Benutzerdefinierte setzt Standardvorlage außer Kraft**.

6.3 Importieren von Quellcode

Vorhandener Java-, C#- und VB.NET-Programmcode kann in UModel importiert werden (dieser Vorgang wird auch als "Reverse Engineering") bezeichnet. Die folgenden Projekttypen können in UModel importiert werden:

- Java-Projekte (Eclipse-.project-Dateien, NetBeans-project.xml-Dateien und JBuilder-.jpx-Dateien)
- C#- und VB.NET-Projekte (Visual Studio sln, csproj, csdprj..., vbproj, vbp sowie Borland .bdsproj-Projektdateien)

Sie können nicht nur Quellcode aus einem Quellprojekt, sondern auch aus einem Quellverzeichnis importieren. Der Import aus einem Quellverzeichnis funktioniert ähnlich und ist vor allem dann nützlich, wenn in Ihrem Code keiner der oben aufgelisteten Projekttypen verwendet wird. Ein Beispiel zum Importieren eines Quellverzeichnisses finden Sie unter Reverse Engineering (Code zu Modell) 69.

Quellcode kann außerdem entweder in ein neues leeres UModel-Projekt oder in ein vorhandenes UModel-Projekt importiert werden. Sie können beim Import angeben, ob die importierten Elemente diejenigen im Modell selbst (falls vorhanden) überschreiben sollen, oder ob sie in das Modell zusammengeführt werden sollen. Optional können Klassen- und Paketdiagramme beim Importieren von Code automatisch generiert werden.

Der Importassistent enthält eine Reihe von Importoptionen für die einzelnen Plattformen (Java, .NET). Wenn der importierte Java/C#/VB.NET-Code etwa Kommentare enthält, können diese optional in UModel-Dokumentation konvertiert werden. Eine vollständige Liste aller Optionen finden Sie unter Optionen für den Code-Import 196.

Nachdem Ihr C#-, VB.NET- oder Java-Code in UModel importiert wurde, können Sie das Modell ändern (z.B. durch Hinzufügen neuer Klassen oder Umbenennen von Eigenschaften und Operationen) und es optional wieder mit dem Originalcode rücksynchronisieren, sodass ein vollständiges Round-Trip-Engineering durchgeführt wird, siehe Synchronisieren von Modell und Quellcode 221.

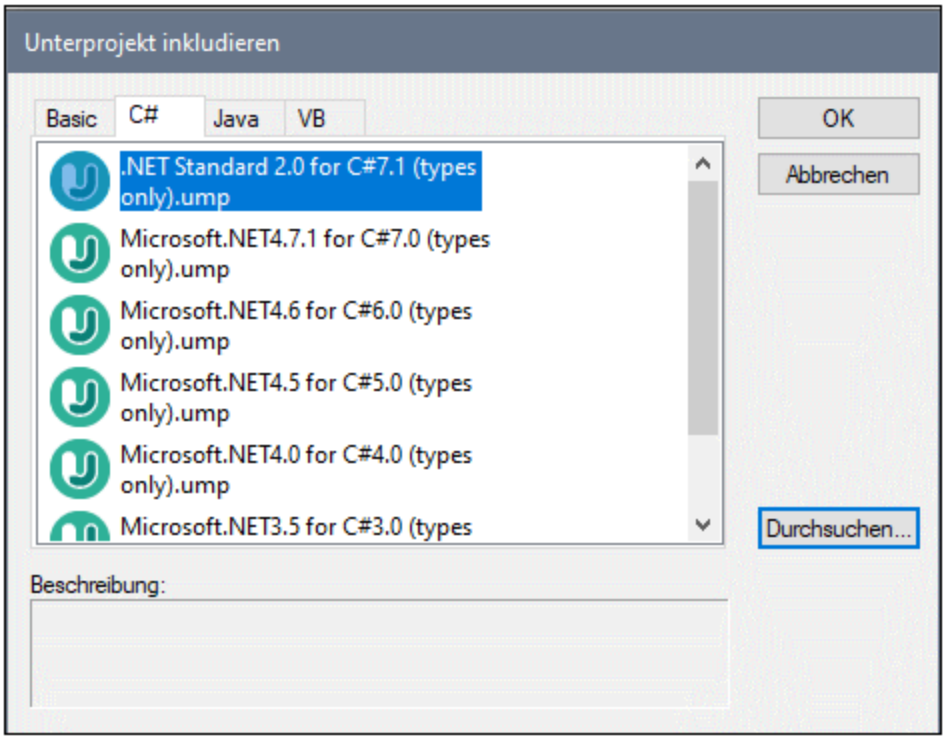
Voraussetzungen

UModel enthält mehrere vordefinierte Unterprojekte, die speziell für das Code Engineering erstellt wurden und die die Datentypen für die jeweilige unterstützte Sprache und Plattform enthalten. Bevor Sie versuchen, Quellcode in ein UModel-Projekt zu importieren, sollten Sie das vordefinierte UModel-Unterprojekt für die entsprechende Programmiersprache und Plattform inkludieren, siehe Inkludieren von Unterprojekten 164. Andernfalls werden bestimmte Datentypen nicht erkannt und werden nach dem Import in ein separates Paket mit dem Namen "Unbekannte externe Elemente" platziert.

So inkludieren Sie ein Unterprojekt mit den erforderlichen Sprachdatentypen:

1. Klicken Sie im Menü **Projekt** auf **Unterprojekt inkludieren**.
2. Klicken Sie auf das Register für die entsprechende Quellsprache und Plattform (z.B., Java 8.0, C# 6.0, VB 9.0) und klicken Sie anschließend auf OK.

 © 2016-2022 Altova GmbH



Beachten Sie die folgenden Punkte:

- Wenn Sie ein Datentyp-Unterprojekt für eine bestimmte Sprache inkludieren, fügt UModel automatisch auch das Profil dieser Sprache zu Ihrem Projekt hinzu. Das Profil-Unterprojekt (.ump) enthält nur die wichtigsten Typen und ist nicht mit dem Datentyp-Unterprojekt (auch .ump), welches ausführlichere Typdefinitionen enthält, identisch.
- Wenn Sie den Import ohne Inkludierung eines Datentyp-Unterprojekts durchführen, wird der Import dennoch durchgeführt und UModel inkludiert auch das Profil dieser Sprache in das Projekt. Alle unbekannten Typen werden jedoch in das Paket "Unbekannte externe Elemente" platziert. Um dies zu vermeiden, sollten Sie sicherstellen, dass das Datentyp-Unterprojekt für die erforderliche Sprache und Plattform, wie oben erklärt, inkludiert wird.

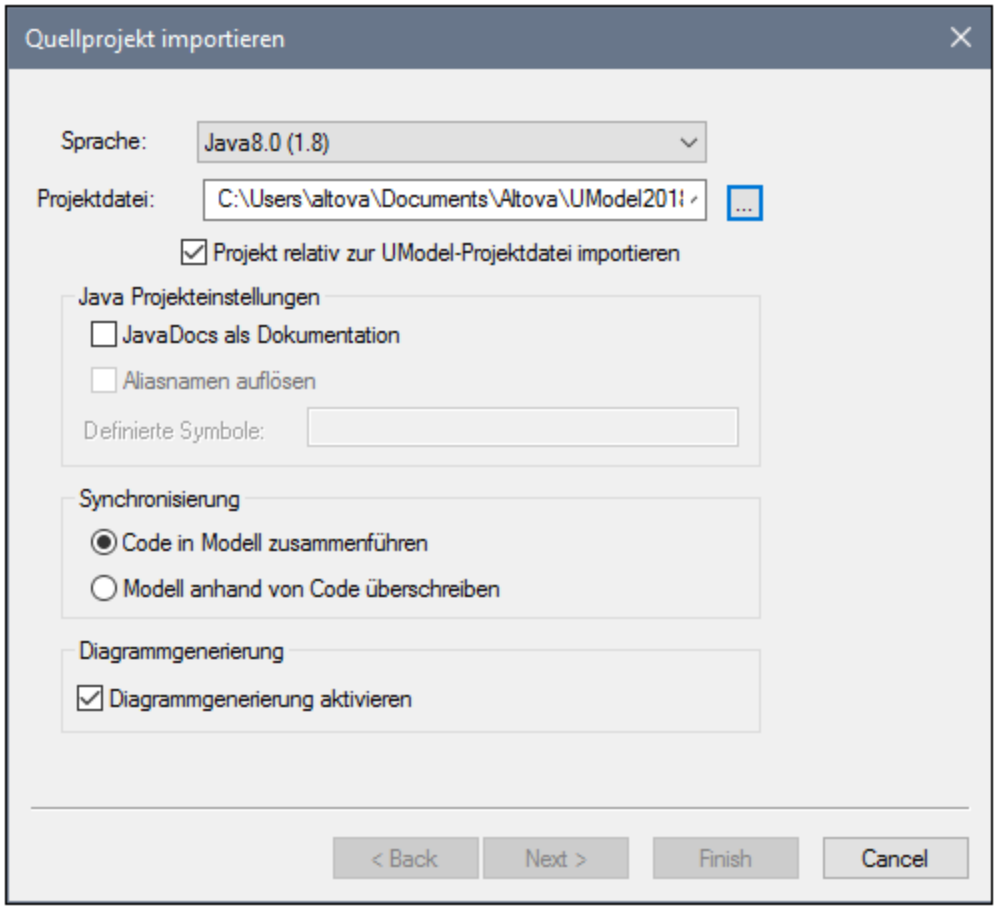
Importieren von Quellcode aus einem Projekt

1. Klicken Sie im Menü **Projekt** auf **Quellprojekt importieren**. (Wenn Sie alternativ dazu Code aus einem bestehenden Verzeichnis importieren möchten, wählen Sie den Befehl **Quellverzeichnis importieren**.)
2. Wählen Sie die Sprachversion des Quellprojekts (z.B. Java 8.0, C# 6.0).
3. Klicken Sie auf **Durchsuchen** ... und wählen Sie die Quellprojektdatei aus.
4. Definieren Sie die erforderlichen Importoptionen oder ändern Sie diese, siehe auch Optionen für den Code-Import 196 (Beachten Sie, dass diese Optionen von der in Schritt 2 ausgewählten Sprache abhängig sind).
5. Klicken Sie auf **Fertig stellen**, um den Assistenten zu beenden.

Ein Schritt-für-Schritt-Beispiel dazu finden Sie unter Beispiel: Importieren eines C#-Projekts 199.

6.3.1 Optionen für den Code-Import

Wenn Sie Programmcode in ein UModel-Projekt importieren, müssen Sie die unten aufgelisteten Optionen eventuell definieren oder ändern. Diese Optionen stehen in dem Dialogfeld, das bei Aufruf des Menübefehls **Projekt | Quellprojekt importieren** oder **Projekt | Quellverzeichnis importieren** angezeigt wird, zur Verfügung.



Dialogfeld "Quellprojekt importieren"

Die meisten der Optionen im Dialogfeld können auch später jederzeit geändert werden, siehe Codesynchronisierungseinstellungen 225.

Die folgenden Optionen gelten unabhängig von Sprache oder Plattform für alle Projekttypen:

Option	Beschreibung
Projekt relativ zur UModel-Projektdatei importieren	Standardmäßig ist diese Option aktiviert, d.h. zwischen dem UModel-Projekt und dem importierten Quellcodeprojekt wird ein relativer Pfad definiert.

 © 2016-2022 Altova GmbH

Option	Beschreibung
	Nach Import des Quellcodes wird im UModel-Projekt automatisch eine UML-Komponente generiert (sie steht in der Modell-Struktur als Child von "Component View" zur Verfügung). Diese Komponente realisiert die zu erzeugenden Schnittstellen oder Klassen; Sie definiert auch die Optionen für das Code Engineering, einschließlich Pfad zum Quellprojekt oder -verzeichnis. Wenn **Projekt relativ zur UModel-Projektdatei importieren** aktiviert ist, handelt es sich hierbei um einen relativen Pfad, andernfalls um einen absoluten.
Code in Modell zusammenführen / Modell anhand von Code überschreiben	Wenn **Code ...zusammenführen** ausgewählt ist, werden potenzielle Namenskonflikte (wie z.B. Paket- oder Klassennamen) durch Anhängen einer Nummer an das zu importierende Element gelöst. Wenn **...überschreiben** ausgewählt ist und es Namenskonflikte gibt, hat das importierte Element Vorrang vor einem im Projekt bereits vorhandenen (und überschreibt dieses).
Diagrammgenerierung aktivieren	Aktivieren Sie dieses Kontrollkästchen optional, wenn Sie anhand der importierten Klassen Klassen- und Paketdiagramme generieren möchten. Wenn dieses Kontrollkästchen aktiviert ist, inkludiert der Importassistent zusätzliche Schritte, über die sie das Aussehen der generierten Diagramme anpassen können.

Die folgenden Optionen gelten nur für C#- und VB.NET-Projekte:

Option	Beschreibung
DocComments als Dokumentation	Aktivieren Sie dieses Kontrollkästchen, um im C#-Code gefundene Kommentare in UModel-Elementdokumentation zu konvertieren (siehe auch Dokumentation 91).

Aliasnamen auflösen	Dieses Kontrollkästchen ist standardmäßig aktiviert. Wenn Ihr C#- oder VB.NET-Code Namespaces oder Klassenaliasse wie im Codefragment unten enthält, wird empfohlen, dieses Kontrollkästchen aktiviert zu lassen. Andernfalls werden Assoziationen und Abhängigkeiten im Zusammenhang mit Klassen und Namespaces in Ihrem Code beim Import von UModel eventuell nicht automatisch erkannt (und wären daher im Modell auch nicht vorhanden).
	
	`using Q = System.Collections.Generic.Queue<String>;`
	`Q myQueue;`
	
	Beispiel für ein Alias in C#-Code
	
	Potenziell Konflikte verursachende Aliasnamen werden beim Import zum Paket "Unbekannte externe Elemente" des UModel-Projekts hinzugefügt, wenn ihre Verwendung nicht klar ist.

Option	Beschreibung
	Wenn Sie den Code anhand des Modells wieder aktualisieren (Round-Trip Engineering), werden die Aliasse, so wie sie im generierten Code vorhanden sind, beibehalten. Die Option **Alialsnamen auflösen** kann später jederzeit wieder geändert werden, siehe Codesynchronisierungseinstellungen 225. Wenn Sie diese Option nach (nicht aber vor) dem Import aktivieren, fordert Sie UModel auf, das Projekt wieder anhand des Codes zu aktualisieren, da die Option sich auch auf das Forward Engineering auswirkt.

Definierte Symbole	Wenn Ihr C#- oder VB.NET-Code Symbole enthält, die über Vorverarbeitungsanweisungen wie `#if`, `#endif` definiert sind, können Sie UModel anweisen, diese beim Reverse Engineering von Code zu berücksichtigen.
	
	`#if DEBUG`
	`  static void DisplayMessage()`
	`  {`
	`  Console.WriteLine("Please wait...");`
	`  }`
	`#endif`
	
	Beispiel für ein Symbol für bedingte Kompilierung in C#-Code
	
	Wenn Sie z.B. am obigen Code ein Reverse Engineering durchführen, wird die Methode `DisplayMessage()` nur dann in das Modell importiert, wenn Sie das `DEBUG`-Symbol definiert haben.
	
	Um Symbole für bedingte Kompilierung zu definieren, geben Sie diese, getrennt durch ein Semikolon, in das Textfeld "Definierte Symbole" ein.
	
	Beim Reverse Engineering werden alle im Quellcode verwendeten Symbole von UModel im Fenster "Meldungen" ausgegeben.

Die folgenden Optionen gelten nur für Java-Projekte:

Option	Beschreibung
JavaDocs als Dokumentation	Aktivieren Sie dieses Kontrollkästchen, um im Code gefundene Kommentare im JavaDocs-Stil in UModel-Elementdokumentation zu konvertieren (siehe auch Dokumentation 91). **Anmerkung:** Nur Kommentare zu Java-Klassen, Schnittstellen, Operationen und Eigenschaften werden konvertiert.

© 2016-2022 Altova GmbH

6.3.2 Beispiel: Importieren eines C#-Projekts

In diesem Beispiel wird gezeigt, wie Sie eine mit Visual Studio erstellte C#-Beispiellösung in UModel importieren. Die Lösung steht als .zip-Archiv unter dem folgenden Pfad zur Verfügung: **C:\Benutzer\<Benutzername>\Dokumente\Altova\UModel2023\UModelExamples\Tutorial\Anagram_CSharp.zip**. Sie müssen die Lösung vor dem Import nicht mit Visual Studio kompilieren, das Archiv **Anagram_CSharp.zip** jedoch in einen Ordner ihrer Wahl entpacken, bevor Sie mit den unten beschriebenen Schritten fortfahren.

Ziel in diesem Beispiel ist die Erstellung eines Reverse Engineering der C#-Lösung und die Erzeugung von eines UModel-Projekts anhand dieses Codes. Bei Importieren des Codes wählen wir die Option zum automatischen Generieren von Klassen- und Paketdiagrammen.

Schritt 1: Erstellen eines neuen Projekts

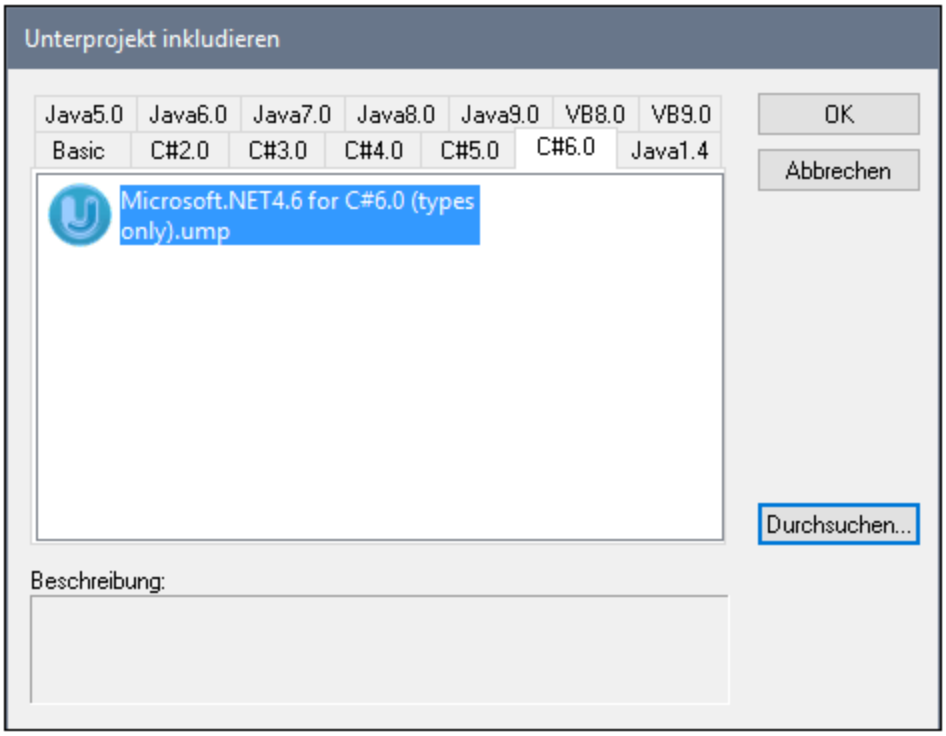
Sie können Quellcode entweder in ein vorhandenes oder in ein neues UModel-Projekt importieren. In diesem Beispiel werden wir Code in ein neues UModel-Projekt importieren.

- Wählen Sie im Menü **Datei** den Befehl **Neu** (Drücken Sie alternativ dazu **Strg + N** oder klicken Sie in der Symbolleiste auf die Schaltfläche **Neu**).

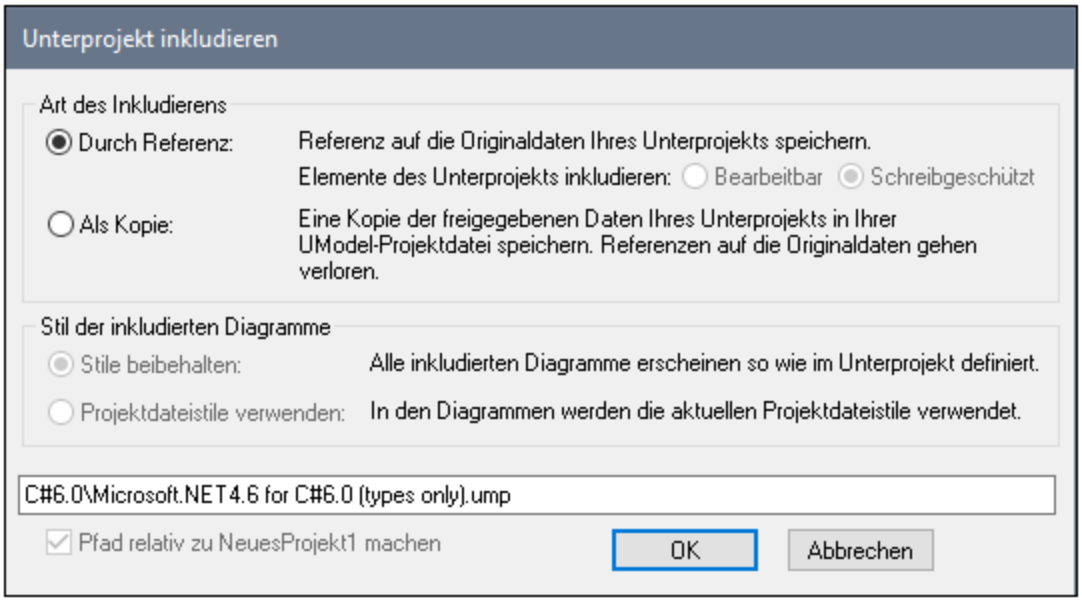
Schritt 2: Inkludieren der C#-Sprachtypen

Das Quellprojekt wurde mit Visual Studio 2015 in C# erstellt, daher inkludieren wir ein vordefiniertes UModel-Projekt, das die C# 6.0-Sprachtypen enthält (da die C#-Sprachversion für Visual Studio 2015 Version 6.0 ist). Wahrscheinlich funktionieren auch frühere Versionen von C# mit unserer C#-Beispiellösung.

1. Klicken Sie im Menü **Projekt** auf **Unterprojekt inkludieren**.
2. Klicken Sie auf das Register **C#**.

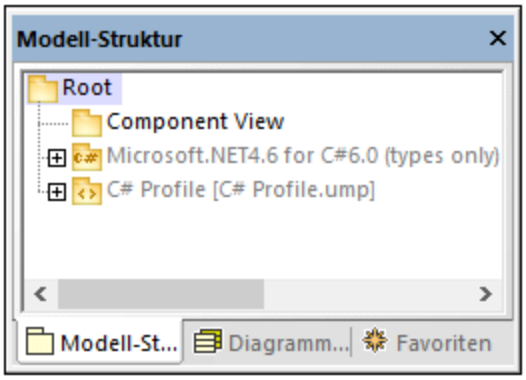


3. Wählen Sie das Projekt **Microsoft .NET 4.6 for C# 6.0 (types only).ump** aus und klicken Sie auf **OK**.
4. Wenn Sie aufgefordert werden, die Art der Inkludierung auszuwählen (durch Referenz oder als Kopie) belassen Sie die Standardoption unverändert.



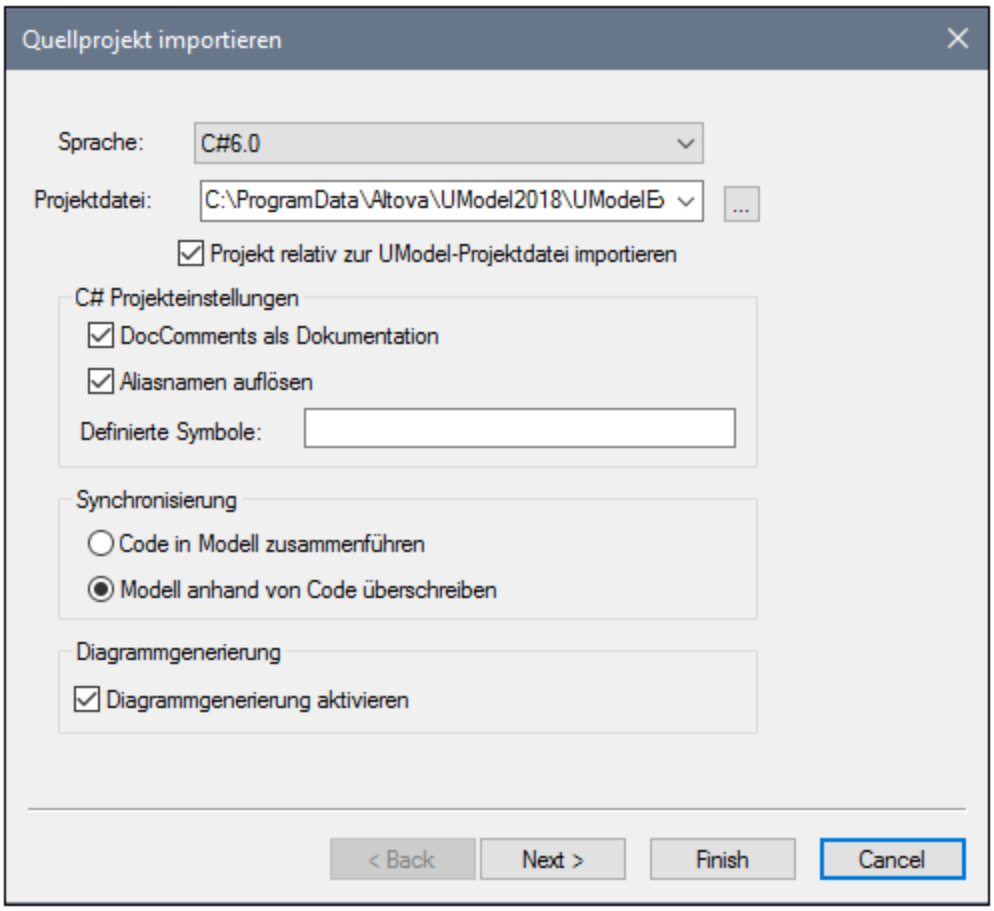
Dadurch werden sowohl die C#-Sprachtypen als auch das C#-Sprachprofil inkludiert und in der Modell-Struktur angezeigt:

 © 2016-2022 Altova GmbH



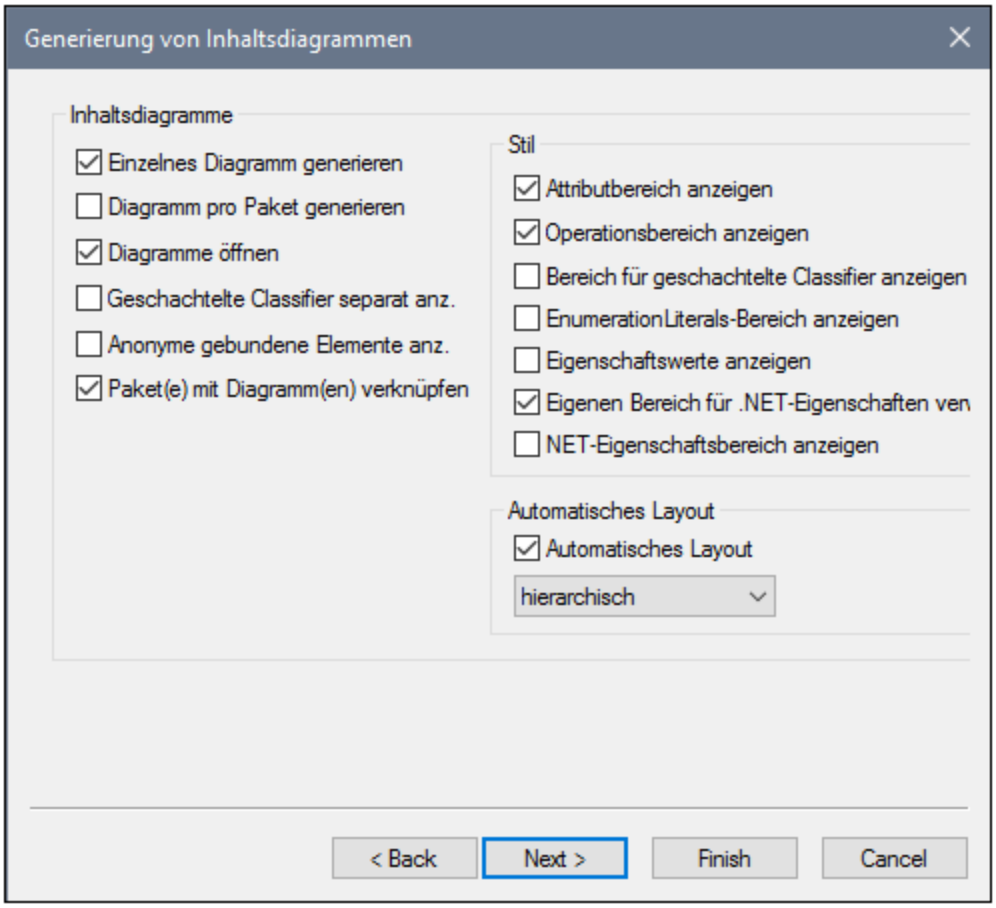
Schritt 3: Importieren der C#-Lösung

1. Klicken Sie im Menü **Projekt** auf **Quellprojekt importieren**.



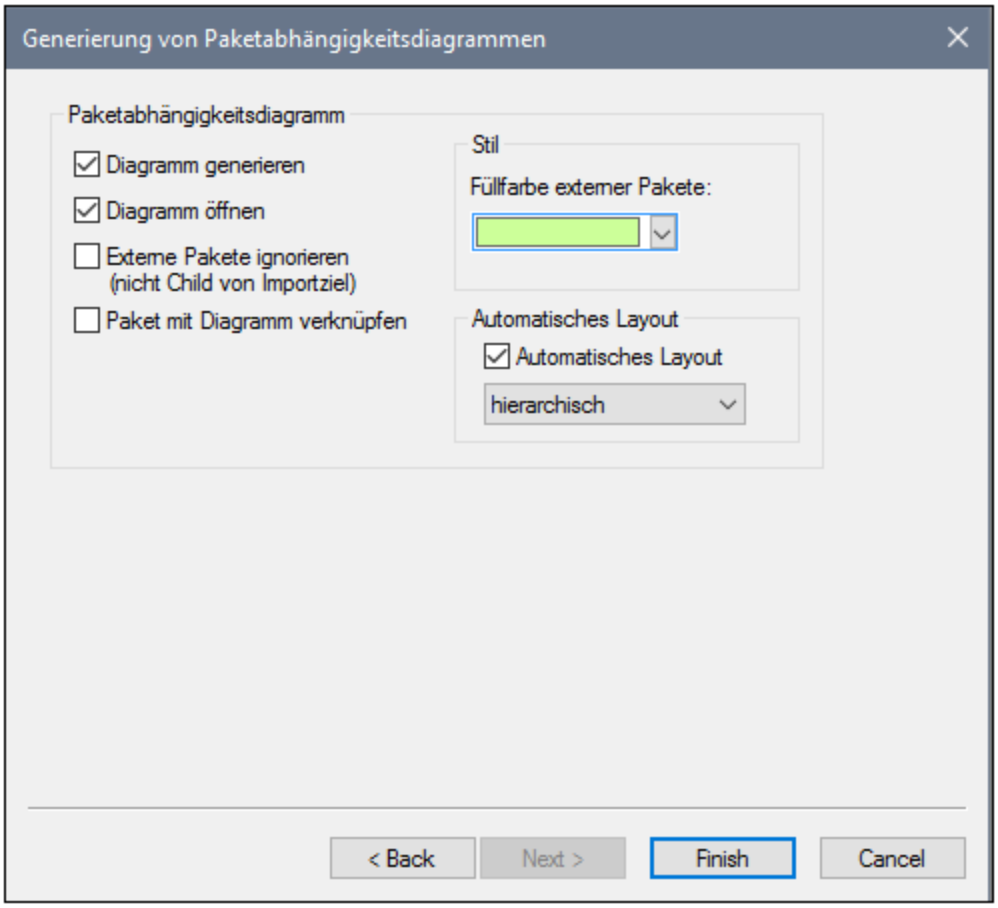
2. Wählen Sie als Sprache **C# 6.0** aus.
3. Klicken Sie neben **Projektdatei** auf **Durchsuchen** ... und navigieren Sie zur .sln-Lösungsdatei.
4. Aktivieren Sie das Kontrollkästchen **DocComments als Dokumentation** (Dadurch werden Codekommentare zu Operationen oder Eigenschaften in das Modell importiert).

5. Da wir Code in ein neues UModel-Projekt importieren, aktivieren Sie die Option **Modell anhand von Code überschreiben** (die andere Option **Code in Modell zusammenführen** eignet sich für den Import in ein vorhandenes Projekt).
6. Klicken Sie auf **Weiter**.
7. Aktivieren Sie die Diagrammgenerierungsoptionen, wie unten gezeigt, und klicken Sie auf **Weiter**. (Diese Optionen gelten für automatisch beim Codeimport generierte Klassendiagramme.)

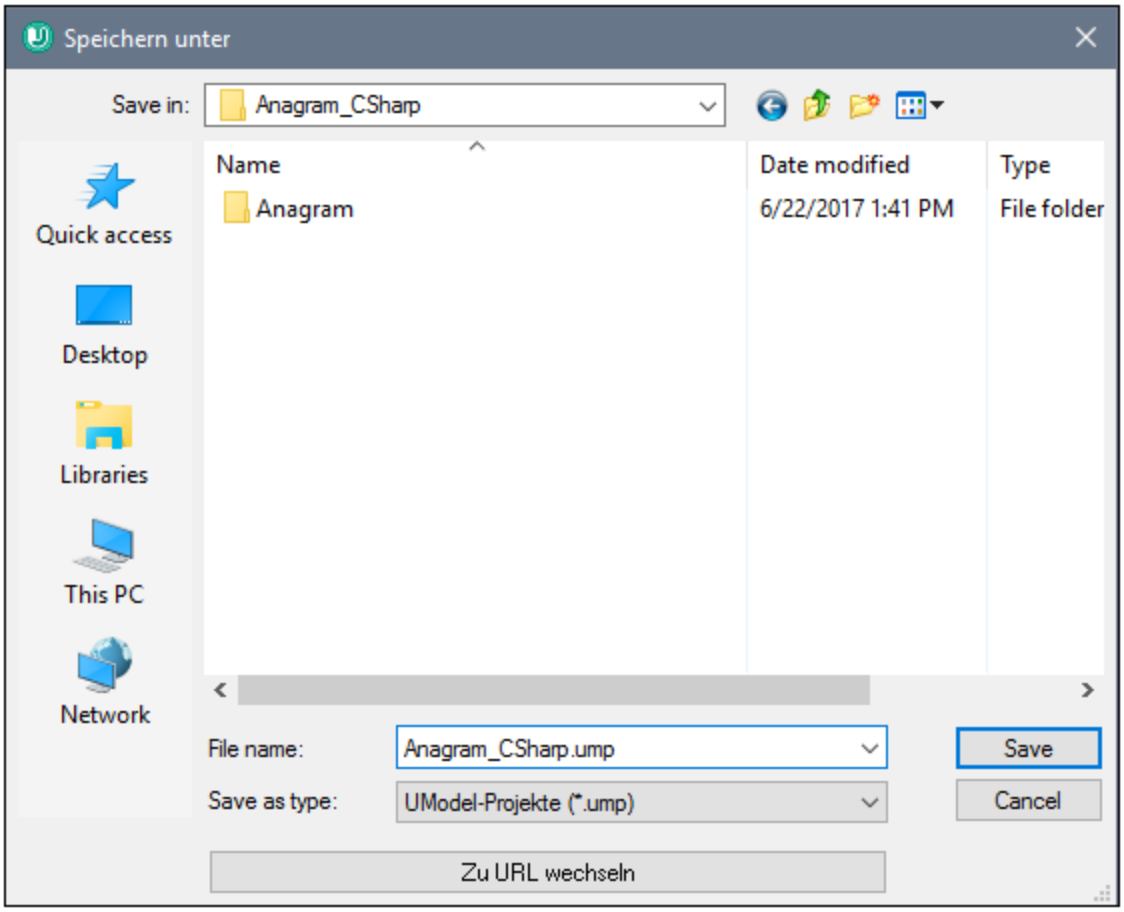


8. Wählen Sie die Diagrammgenerierungsoptionen aus, wie unten gezeigt, und klicken Sie auf **Fertig stellen**. (Diese Optionen gelten für automatisch beim Codeimport generierte Paketdiagramme.)

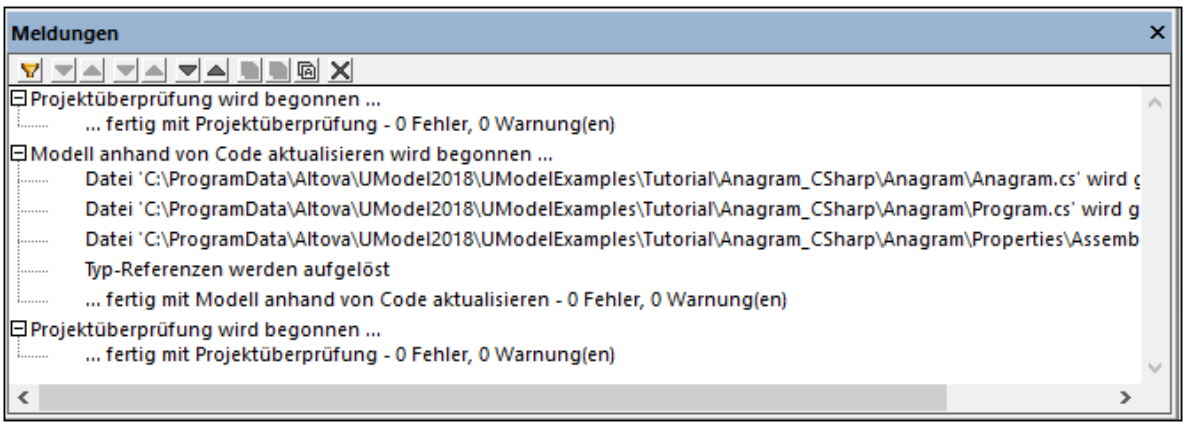
© 2016-2022 Altova GmbH



9. Geben Sie einen Namen ein, wählen Sie einen Zielordner für das neue UModel-Projekt aus und klicken Sie auf **Speichern** (Standardmäßig ist der in diesem Dialogfeld angezeigte Ordner derselbe Ordner, wie der aus, dem die Lösung importiert wurde).

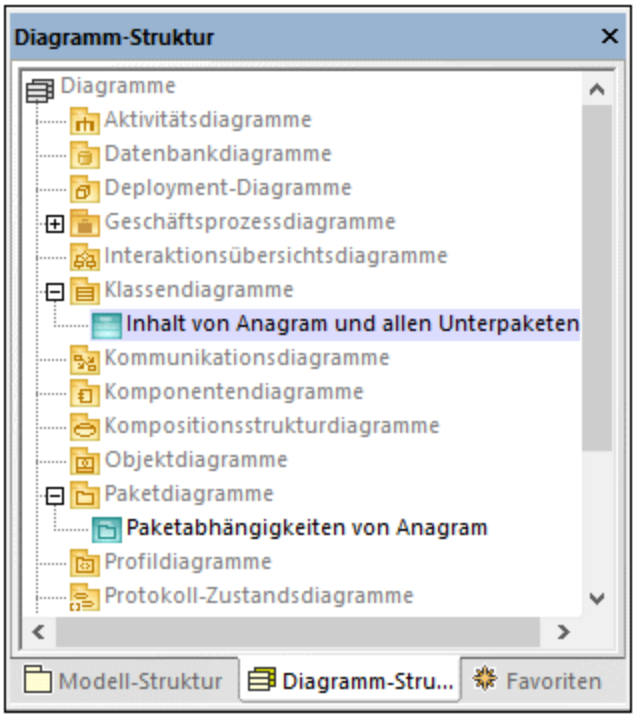


Im Fenster "Meldungen" sehen Sie die Fortschrittsanzeige für das Reverse Engineering.

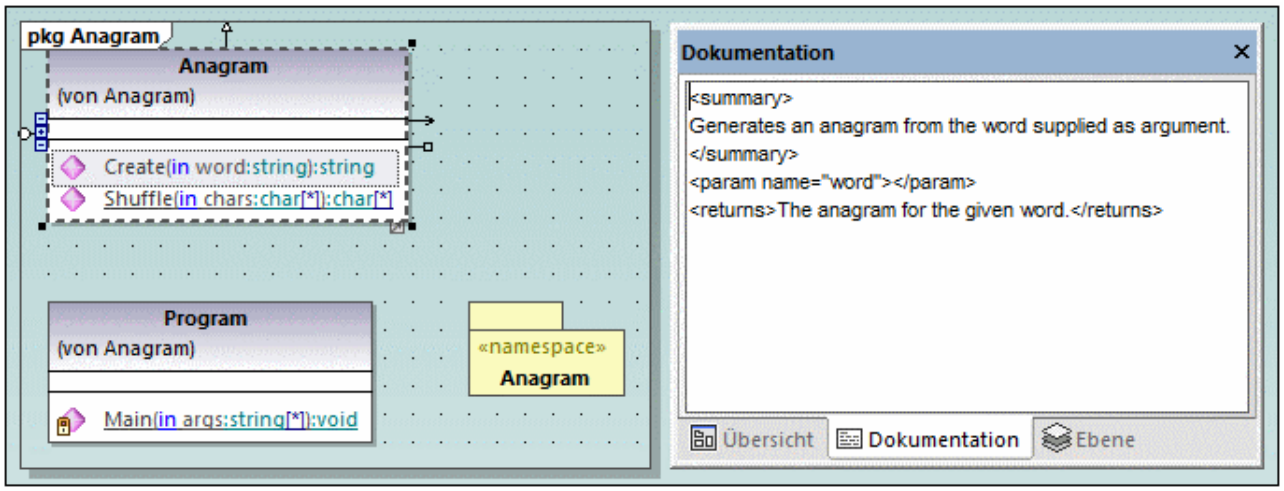


Nach Abschluss des Codeimports werden alle Diagramme automatisch geöffnet, da diese Option vor der Codegenerierung ausgewählt wurde. Alle generierten Diagramme stehen in der Diagramm-Struktur zur Verfügung:

 © 2016-2022 Altova GmbH



Da wir die Option zum Generieren von Dokumentation anhand des Quellcodes ausgewählt haben, wird die importierte Dokumentation im Fenster **Dokumentation** angezeigt, z.B. wenn Sie in der Klasse `Anagram` auf die Operation `create` klicken:



Anmerkung: Die Dokumentation wird nur hinzugefügt, wenn beim Import der C#-Lösung die Option **DocComments als Dokumentation** aktiviert war (siehe "Schritt 3: Importieren der C#-Lösung weiter oben).

6.4 Importieren von Java-, C#- und VB.NET-Binärdateien

UModel unterstützt den Import von C#-, Java- und VB.NET-Binärdateien. Diese Funktion erweist sich vor allem beim Arbeiten mit Binärdateien aus externen Quellen oder, wenn der Quellcode nicht mehr zur Verfügung steht, als besonders nützlich. Beachten Sie dazu Folgendes:

- Um Java-Binärdateien zu importieren, muss eine unterstützte Version [11] von Java Runtime Environment (JRE) oder Development Kit (JDK) installiert sein. Der Import von Typen wird für Java .class-Dateien oder .jar Class Archives, die den Java Virtual Machine Spezifikationen entsprechen, unterstützt. Dazu gehören Java Virtual Machines wie OpenJDK, SapMachine, Liberica JDK und andere, siehe Hinzufügen von benutzerdefinierten Java Runtimes [207].
- Um C#- oder VB.NET-Binärdateien zu importieren, muss je nach Bedarf .NET Framework, .NET Core, .NET 5 oder .NET 6 installiert sein. Die besten Ergebnisse erzielen Sie bei Auswahl der Option **beliebige (Disassembler verwenden)** aus dem Import-Dialogfeld. Alle nicht erkannten Typen werden nach dem Import in das Paket "Unbekannte externe Elemente" platziert. Damit keine (oder weniger") unbekannte externe Elemente importiert werden, wenden Sie *vor dem Import* das entsprechende UModel-Profil für die jeweilige Version Ihrer Code Engineering-Sprache an (z.B. ."NET 5 für C# 9.0"). Siehe auch Anwenden von UModel-Profilen [160].
- Der Import von Binärdateien, die mit Hilfe eines Obfuscators unleserlich gemacht wurden, wird nicht unterstützt.

In der unten stehenden Tabelle finden Sie eine Liste der verfügbaren Methoden für den Import von Binärdateitypen in ein UModel-Projekt.

C#, VB.NET	Java
Import einer Assembly-Datei (.dll, .exe)	Import eines Class File Archive (.jar, .zip)
Import einer Assembly aus dem Global Assembly Cache (GAC)	Import einer Klassendatei (.class) aus einem Paket-Root-Ordner
Import einer Assembly aus Visual Studio .NET-Referenzen	Import von Class Archives aus dem Klassenpfad
	Import von Class Archives aus Java Runtime (Nur für Java-Versionen bis einschließlich Java 8)

Binärdateien können mit dem Menübefehl **Projekt | Binärtypen importieren** importiert werden. Sie können UModel optional Klassen- und Paketdiagramme anhand der importierten Typen generieren lassen. Beispiele dazu finden Sie unter Beispiel: Import von .NET GAC Assemblys [211] und Beispiel: Import von Java .class-Dateien [214].

Zusätzlich dazu können Binärtypen auch über die Befehlszeile (siehe UModel Befehlszeilenschnittstelle [97]) importiert werden.

Wenn Sie Binärdateien in ein UModel-Projekt importieren, können Sie verschiedene Importoptionen, darunter die folgenden, definieren:

- Zusätzlich zu den in der Binärdatei definierten Typen können Sie jeden beliebigen referenzierenden Typ importieren. Außerdem können Sie den Import von referenzierenden Typen auf bestimmte Java-Pakete und .NET-Namespaces einschränken.

 © 2016-2022 Altova GmbH

- Typmember können beim Import übersprungen werden. So können Sie etwa Klassen und Schnittstellen ohne deren Eigenschaften und Methoden importieren.
- Typen können entsprechend ihren Zugriffsmodifizierern (wie z.B. private oder public) importiert werden. So können Sie z.B. nur öffentliche Klassen importieren und private, geschützte (protected) und interne Klassen überspringen.

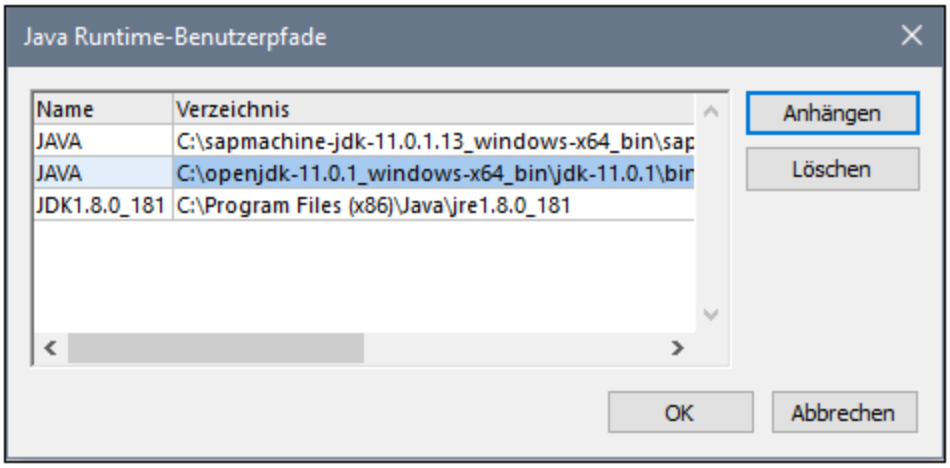
Nähere Informationen zu allen Optionen finden Sie unter Optionen für den Import von Binärdateien 208.

6.4.1 Hinzufügen von benutzerdefinierten Java Runtimes

UModel ermittelt standardmäßig JDKs und JREs, falls diese auf dem lokalen Rechner *installiert* sind. Diese werden folglich beim Start des Binärdateiimportassistenten in der Liste der Java Runtimes angezeigt. Dies ist bei JDKs und JREs von Oracle, die mit einem Installationsprogramm geliefert werden und sich bei der Installation im System selbst registrieren, der Fall. Andere Java Virtual Machine Distributionen hingegen, zu denen es kein Installationsprogramm gibt, müssen manuell zu UModel hinzugefügt werden. Dazu gehören Oracle OpenJDK, SapMachine und andere.

So fügen Sie benutzerdefinierte Java Runtimes zu UModel hinzu:

1. Klicken Sie im Menü **Projekt** auf **Binärtypen importieren**.
2. Wählen Sie als Sprache **Java** aus.
3. Erweitern Sie die **Runtime** Dropdown-Liste und klicken Sie auf **Java Runtime-Benutzerpfade bearbeiten...**.
4. Klicken Sie auf **Anhängen** und navigieren Sie zum Verzeichnis für das jeweilige JDK.



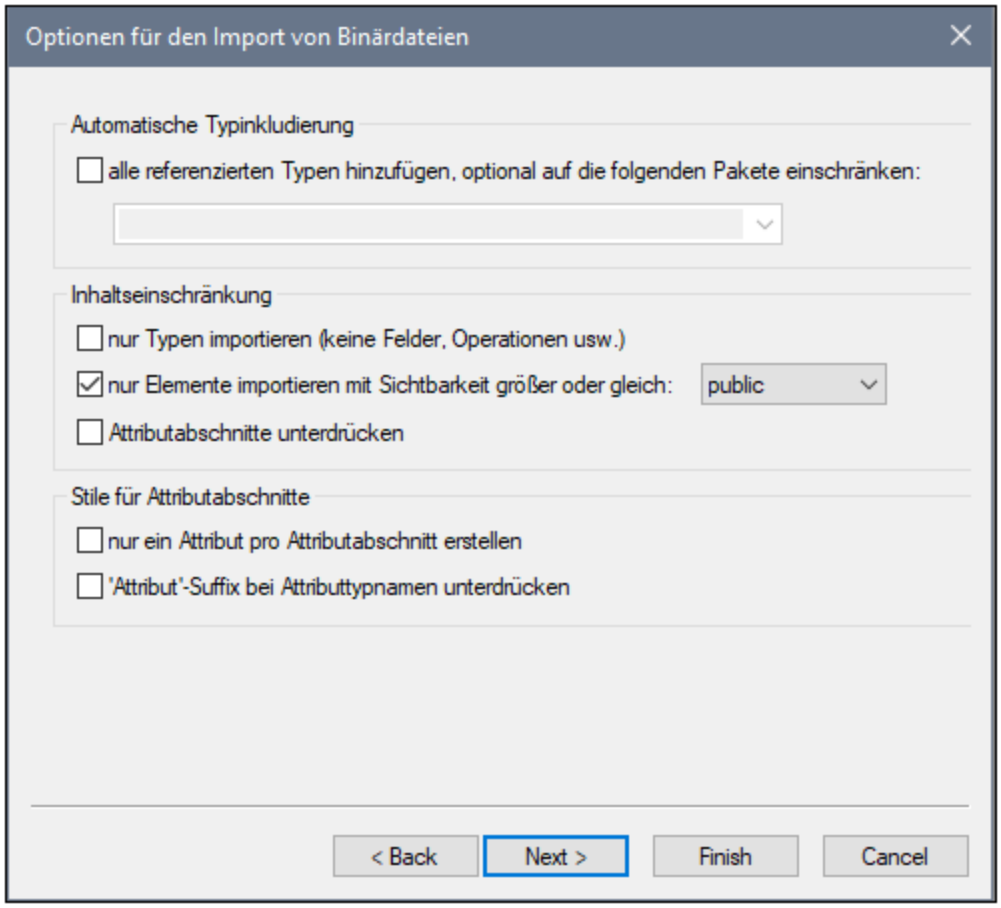
5. Klicken Sie auf **OK**.

Die ausgewählte Runtime wird nun in der **Runtime-**Liste angezeigt und steht zur Auswahl bereit, wenn Sie Binärdateien für diese Runtime importieren müssen.

Beachten Sie, dass sich diese Einstellungen nur auf den Import von Binärdateien auswirken. Informationen über das Hinzufügen eines Java Virtual Machine-Pfads für JDBC-Verbindungen und die Generierung und den Import von Java-Code finden Sie unter Java Virtual Machine-Einstellungen 531.

6.4.2 Optionen für den Import von Binärdateien

Bei Aufruf des Menübefehls **Projekt | Binärtypen importieren** werden Sie in einem der Schritte des Assistenten aufgefordert, Optionen für den Import von Binärdateien zu definieren. Im Folgenden wird beschrieben, welche Optionen Sie definieren können. Beachten Sie, dass die Dialogfeldoptionen je nachdem, ob Sie .NET- oder Java-Binärdateien importieren, etwas unterschiedlich sein können.



Dialogfeld "Optionen für den Import von Binärdateien"

Automatische Typinkludierung

.NET- oder Java-Binärdateien können verschiedene externe Assemblys oder Pakete referenzieren. Aktivieren Sie die Option **alle referenzierten Typen hinzufügen...** , wenn alle Typen, die von den in der Binärdatei inkludierten Typen referenziert werden, importiert werden sollen.

Um referenzierte Typen nur für bestimmte Java-Pakete oder .NET-Namespaces zu importieren, geben Sie die entsprechenden Pakete oder Namespaces in das Textfeld daneben ein. Mehrere Pakete oder Namespaces können durch Kommas, Semikola oder Leerzeichen voneinander getrennt werden.

Angenommen, die Quell-.NET .dll-Datei referenziert Typen aus den Namespaces `System.Reflection` und `System.Data`. Wenn Typen aus dem Namespace `System.Reflection`, nicht aber solche aus dem

 © 2016-2022 Altova GmbH

Namespace `System.Data` importiert werden sollen, aktivieren Sie die Option **alle referenzierten Typen hinzufügen, optional auf die folgenden Pakete einschränken:** und geben Sie in das Textfeld "System.Reflection" ein.

Inhaltseinschränkung

Aktivieren Sie die Option **Nur Typen importieren,** um Member wie Felder, Operationen, Eigenschaften usw. zu überspringen.

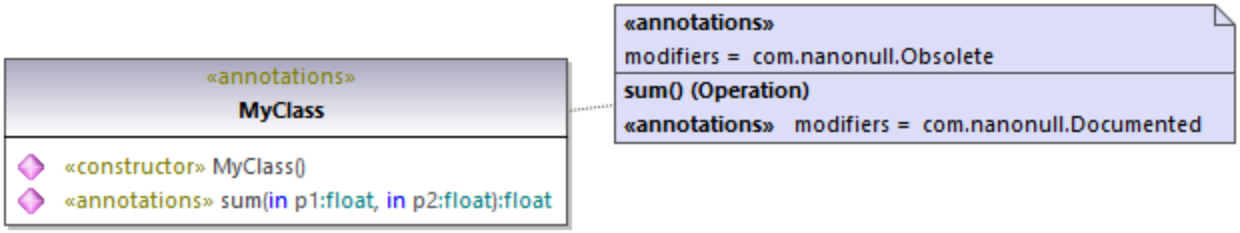
Aktivieren Sie die Option **nur Elemente importieren mit Sichtbarkeit größer oder gleich:**, um Typen und Typmember nach ihrer Sichtbarkeit zu importieren. In der Tabelle unten ist die Sichtbarkeit von Typen beginnend mit Typen mit der geringsten Sichtbarkeit aufgelistet. Wenn Sie z.B. "private" auswählen, werden alle Typen importiert, während bei Auswahl von "public" nur öffentliche Typen und Typmember importiert werden.

Anmerkung: Wenn das Kontrollkästchen deaktiviert ist, werden alle Typen unabhängig von ihrer Sichtbarkeit importiert.

.NET	Java
private	private
internal	package (Standardsichtbarkeit, wenn kein expliziter Modifier vorhanden ist)
protected	protected
public	public

Die Option **Attributabschnitte unterdrücken** ist auf .NET-Binärdateien anwendbar. Standardmäßig importiert UModel die in der Binärdatei gefundenen C#- oder VB.NET-Attribute. Aktivieren Sie die Option **Attributabschnitte unterdrücken,** wenn Attribute nicht importiert werden sollen. Andernfalls wird auf Member, die in Originalquellcode mit Attributen ausgestattet waren, nach Import der Binärdatei in das Modell das Stereotyp `<<attributes>>` angewendet. Wenn Attribute importiert werden, können Sie diese im Diagramm als Eigenschaftswerte anzeigen, Klicken Sie dazu mit der rechten Maustaste im Diagramm auf die Klasse und wählen Sie im Kontextmenü den Befehl **Eigenschaftswerte | Alle**. Nähere Informationen dazu finden Sie unter Stereotype und Eigenschaftswerte 145.

Die Option **Attributabschnitte unterdrücken** kann auch auf Java-Binärdateien angewendet werden. Standardmäßig importiert UModel die in der Binärdatei gefundenen Java-Annotationen, vorausgesetzt ihre Beibehaltungsrichtlinie wurde als `RUNTIME` (nicht `CLASS` oder `SOURCE`) definiert. Wenn keine Annotationen importiert werden sollen, aktivieren Sie die Option **Annotations-Modifier unterdrücken**. Wenn Annotationen importiert werden, erhalten Member, die in der Originaldatei Annotationen hatten, das Stereotyp `<<annotations>>` und Annotationen werden, wie unten gezeigt, als Eigenschaftswerte angezeigt.



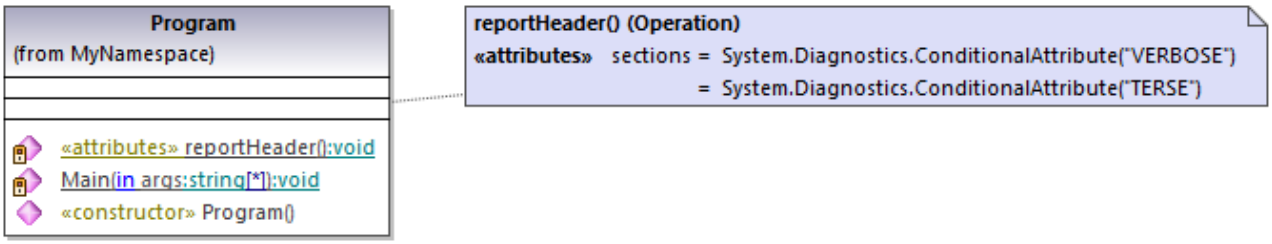
Stile für Attributabschnitte

Diese Optionen sind nur auf .NET-Binärdateien anwendbar. Wie bereits zuvor erwähnt, werden Typen oder Typmember in UModel als Eigenschaftswerte importiert, falls diese im ursprünglichen Quellcode Attribute hatten.

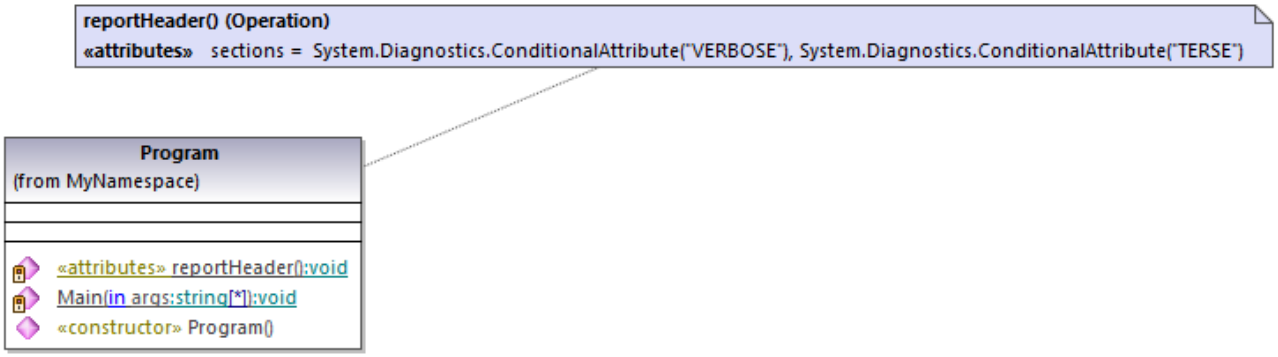
Am besten lässt sich die Option **nur ein Attribut pro Attributabschnitt erstellen** anhand eines Beispiels erläutern. Angenommen, im ursprünglichen C#-Quellcode ist eine Methode mit zwei Attributen definiert:

```
using System;
using System.Diagnostics;

namespace MyNamespace
{
    class Program
    {
        [Conditional("VERBOSE"), Conditional("TERSE")]
        static void reportHeader()
        {
            Console.WriteLine("This is the header");
        }

        static void Main(string[] args)
        {
            reportHeader();
        }
    }
}
```

Wenn beim Import der Binärdatei die Option **nur ein Attribut pro Attributabschnitt erstellen** aktiviert ist, würde jedes Attribut im Element "Eigenschaftswerte" in einer eigenen Zeile angezeigt:



Andernfalls würde die Attribute durch Kommas getrennt:



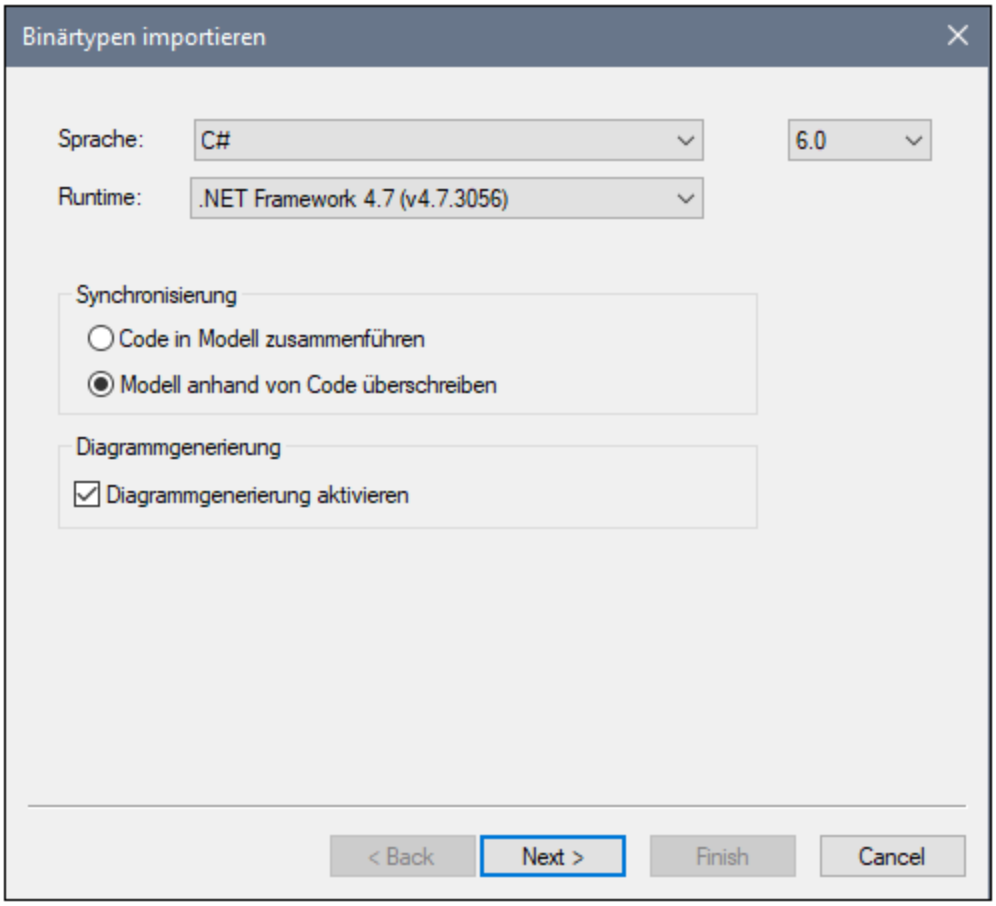
Mit der Option **'Attribut'-Suffix bei Attributtypnamen unterdrücken** schließlich wird das 'Attribute'-Suffix eines Attributtyps entfernt. So würde z.B. ein im Quellcode definierter Attributtyp `System.Xml.Serialization.XmlTypeAttribute` als `System.Xml.Serialization.XmlType` importiert, wenn diese Option aktiviert ist.

6.4.3 Beispiel: Import von .NET Assemblys

In diesem Beispiel wird gezeigt, wie Sie Binärtypen aus dem .NET Global Assembly Cache (GAC) in ein C#-UModel-Projekt importieren. Die Vorgangsweise ist ähnlich wie beim Import von Binärtypen aus einer Standalone `.dll`- oder `.exe`-Datei. Eine Anleitung zum Import von Java `.class`-Dateien finden Sie im nächsten Kapitel 214.

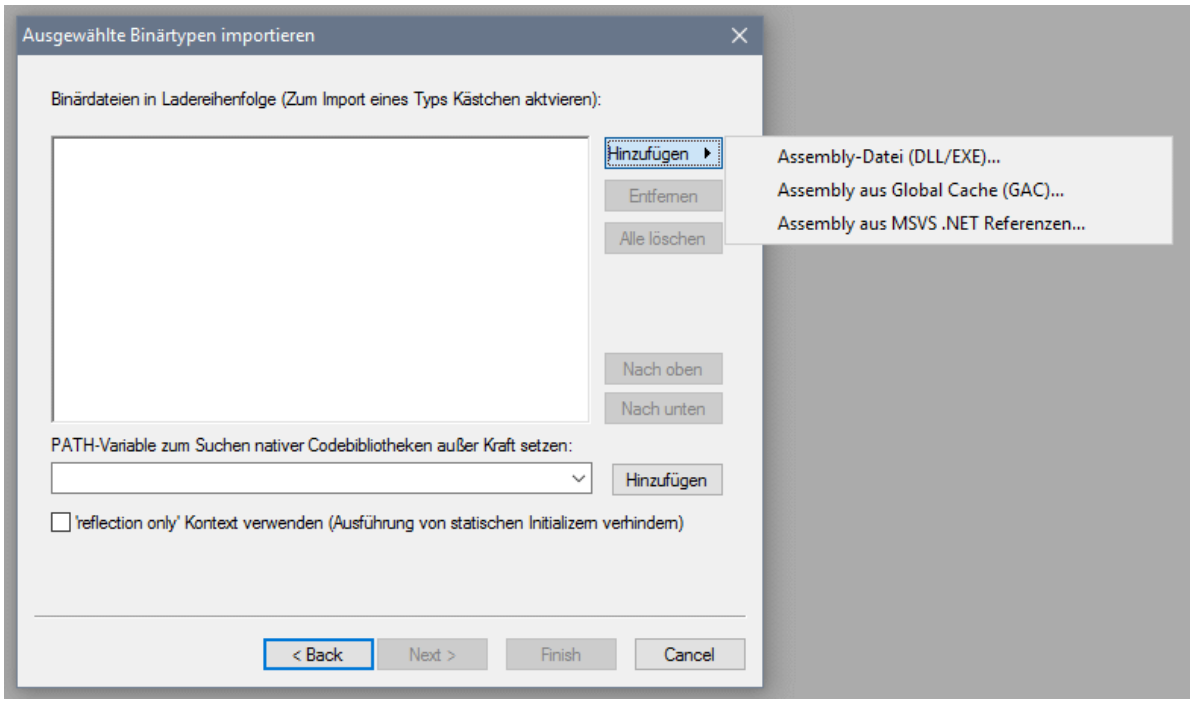
So importieren Sie Binärdateien aus dem .NET Global Assembly Cache:

1. Klicken Sie im Menü **Projekt** auf **Binärtypen importieren** (*siehe Abbildung unten*).

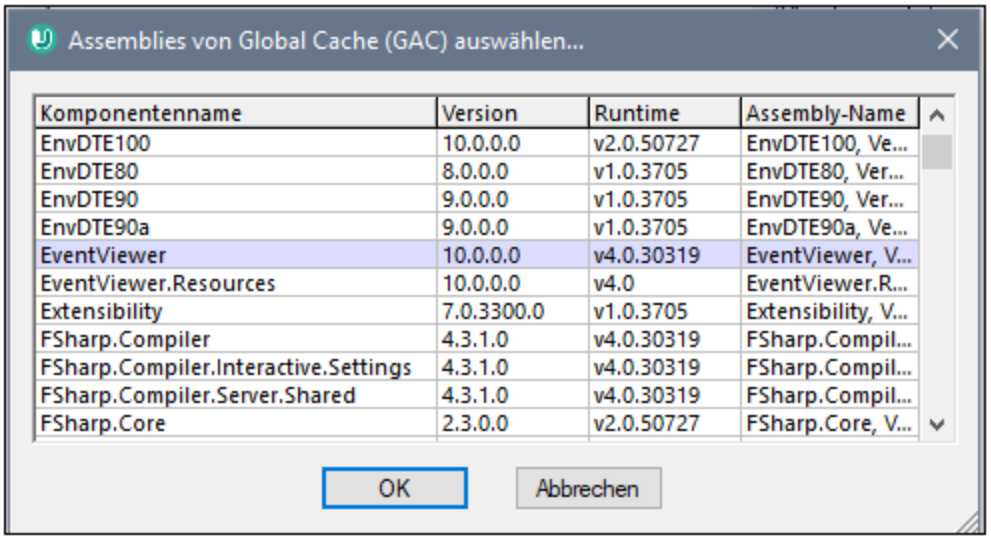


2. Wählen Sie die Zielsprache des UModel-Projekts aus (C#, VB.NET, Java). In diesem Beispiel wird C# ausgewählt, da wir eine .NET GAC Assembly importieren.
3. Wenn Sie eine bestimmte Sprachversion für das importierte UModel-Projekt festlegen möchten, wählen Sie diese aus dem benachbarten Textfeld aus. In diesem Beispiel wird C# 7.3 ausgewählt.
4. Wählen Sie optional eine .NET Runtime-Version aus der **Runtime** Dropdown-Liste aus. Die Standardoption ist *beliebige (Disassembler verwenden)*. In diesem Fall wählt UModel die für die importierte Binärdatei am besten geeignete Reflection APIs aus.
5. Wenn Sie Binärtypen in ein neues Projekt importieren, wählen Sie entweder **Code in Modell zusammenführen** oder **Modell anhand von Code überschreiben** aus.
6. Um anhand der importierten Binärtypen optional Klassen- und Paketdiagramme zu generieren, aktivieren Sie das Kontrollkästchen **Diagrammgenerierung aktivieren**. Wenn Sie diese Option auswählen, stehen in den nächsten Schritten weitere Diagrammgenerierungsoptionen zur Verfügung. Siehe auch Generieren von Klassendiagrammen 407 und Generieren von Paketdiagrammen 418.
7. Klicken Sie auf **Weiter**.
8. Klicken Sie auf **Hinzufügen | Assembly aus Global Cache (GAC)** (*siehe Abbildung unten*). Beachten Sie, dass die Option **Assembly aus Global Cache (GAC)** nur für NET Framework 2.x-4.x zur Verfügung steht. Für .NET Core, .NET 5 und spätere Versionen ist der GAC nicht relevant. Nähere Informationen dazu finden Sie in der Microsoft-Dokumentation. Um Assembly-Dateien für .NET Core, .NET 5 und .NET 6 zu importieren, müssen Sie die erforderlichen Dateien aus dem GAC extrahieren. Klicken Sie anschließend auf **Hinzufügen | Assembly-Datei (DLL/EXE)**, wählen Sie die Assembly-Dateien manuell aus und fügen Sie sie zum Projekt hinzu.

 © 2016-2022 Altova GmbH



9. Wählen Sie eine Assembly aus dem Dialogfeld aus. In diesem Beispiel wurde die Assembly *EventViewer* ausgewählt (*siehe Abbildung unten*).



Komponentenname	Version	Runtime	Assembly-Name
EnvDTE100	10.0.0.0	v2.0.50727	EnvDTE100, Ve...
EnvDTE80	8.0.0.0	v1.0.3705	EnvDTE80, Ver...
EnvDTE90	9.0.0.0	v1.0.3705	EnvDTE90, Ver...
EnvDTE90a	9.0.0.0	v1.0.3705	EnvDTE90a, Ve...
EventViewer	10.0.0.0	v4.0.30319	EventViewer, V...
EventViewer.Resources	10.0.0.0	v4.0	EventViewer.R...
Extensibility	7.0.3300.0	v1.0.3705	Extensibility, V...
FSharp.Compiler	4.3.1.0	v4.0.30319	FSharp.Compil...
FSharp.Compiler.Interactive.Settings	4.3.1.0	v4.0.30319	FSharp.Compil...
FSharp.Compiler.Server.Shared	4.3.1.0	v4.0.30319	FSharp.Compil...
FSharp.Core	2.3.0.0	v2.0.50727	FSharp.Core, V...

10. Wählen Sie die zu importierenden Typen aus und klicken Sie auf **Weiter**. Nähere Informationen zu anderen Optionen des Dialogfelds **Ausgewählte Binärtypen importieren** finden Sie in den Anmerkungen weiter unten.
11. Wählen Sie die entsprechenden Importoptionen aus. Nähere Informationen dazu finden Sie unter Optionen für den Import von Binärdateien 208.
12. Wenn Sie in Schritt 6 die Diagrammgenerierung aktiviert haben, klicken Sie auf **Weiter** und konfigurieren Sie die entsprechenden Optionen dafür. Klicken Sie andernfalls auf **Fertig stellen**.

UModel führt die Konvertierung durch und zeigt im Fenster **Meldungen** die Fortschritte an. Wenn die

Konvertierung von Binärtypen nicht durchgeführt werden kann, finden Sie in der Fehlermeldung eventuell zusätzliche Informationen dazu. So kann es z.B. vorkommen, dass die Binärdatei, die Sie versuchen zu importieren, für eine Runtime-Version gedacht ist, die neuer als die von Ihnen im Dialogfeld **Binärtypen importieren** ausgewählte ist. Wählen Sie in diesem Fall eine neuere Runtime-Version aus und versuchen Sie es erneut.

Anmerkungen:

- Das Textfeld **PATH-Variable ... außer Kraft setzen** im Dialogfeld **Ausgewählte Binärtypen importieren** gilt nur für Java. Fügen Sie optional hier alle Java-Klassenpfade ein, die zusätzlich zu den aus der Umgebungsvariablen `CLASSPATH` ausgelesenen abgefragt werden sollen. Klicken Sie alternativ dazu auf **Hinzufügen** und navigieren Sie zu den erforderlichen Ordnern.
- Das Kontrollkästchen **'reflection only'-Kontext verwenden...** im Dialogfeld **Ausgewählte Binärtypen importieren** gilt nur, wenn Sie eine C#- oder VB.NET-Binärdatei importieren. Dies ist nützlich, wenn Sie eine Bibliothek importieren, die Abhängigkeiten hat, die nicht aufgelöst oder geladen werden können. Wenn Sie dieses Kontrollkästchen aktivieren, wird statischer Initializer-Code, der beim Import zu Fehlern führen könnte, nicht ausgeführt.

6.4.4 Beispiel: Import von Java .class-Dateien

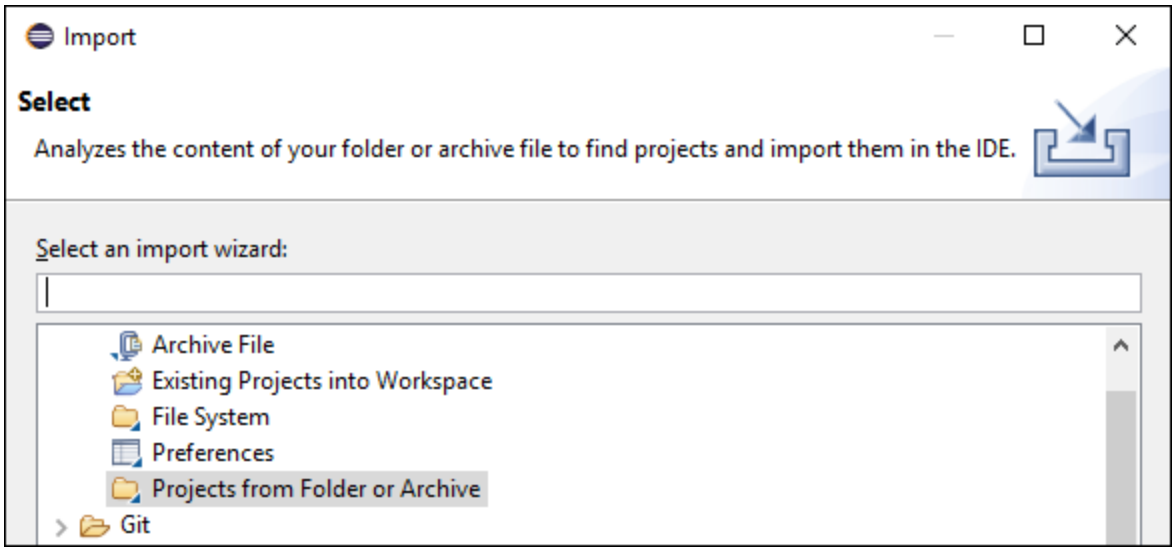
In diesem Beispiel wird gezeigt, wie Sie kompilierte Java .class-Dateien in UModel importieren. Die Java .class-Quelldateien in diesem Beispiel stammen aus einem mit UModel erstellten Java-Tutorial-Projekt, Sie können jedoch alternativ dazu auch andere .class-Dateien verwenden.

Kompilieren von mit UModel generiertem Java-Code (optional)

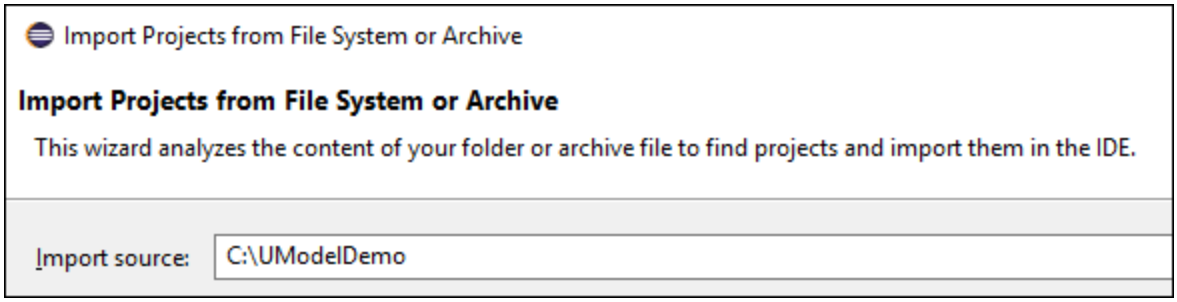
In diesem Abschnitt wird gezeigt, wie Sie ein mit UModel generiertes Java-Demo-Projekt mit Eclipse kompilieren. Beachten Sie, dass dies ein rein optionaler Schritt ist. Ziel ist es, eine kompilierte .class-Datei zu erhalten. Wenn Sie bereits Java .class-Dateien zur Verfügung haben, können Sie den Schritt auch überspringen. In diesem Beispiel wird aus praktischen Gründen Eclipse als Kompilierungsumgebung verwendet, Sie können dazu aber auch die Java-Befehlszeile oder eine Java-Entwicklungsumgebung verwenden.

1. Erstellen Sie, wie in Beispiel: Generieren von Java-Code 182 gezeigt, ein einfaches Java-Projekt mit UModel, falls Sie dies noch nicht getan haben. Es handelt sich hierbei um ein ganz einfaches Beispiel bestehend aus einem Java-Paket mit nur einer Klasse. Nach Fertigstellung des Beispiels enthält das Verzeichnis **C:\UModelDemo\src** den benötigten Java-Quellcode.
2. Starten Sie Eclipse. Klicken Sie im Menü **File** auf **Import**.

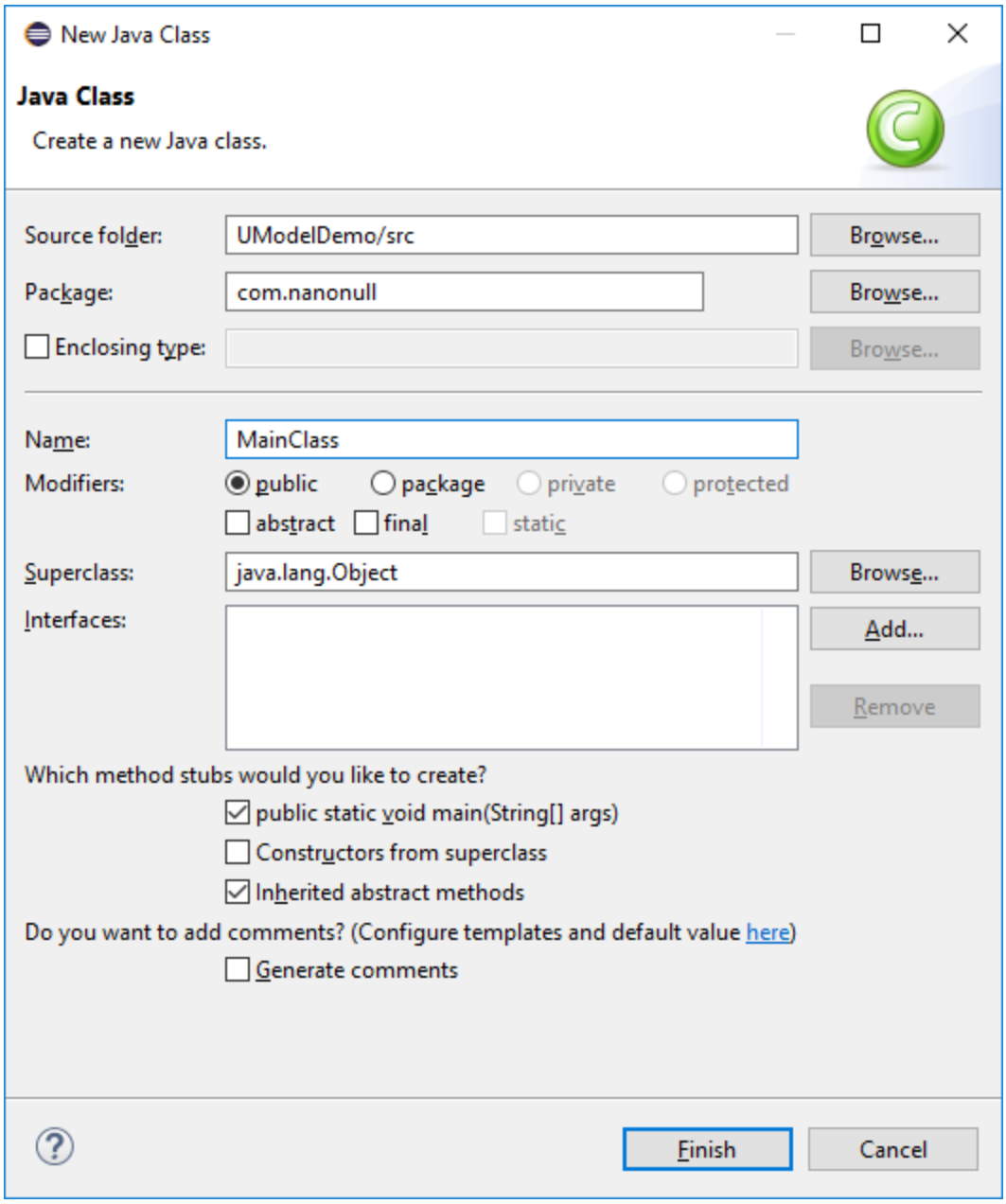
 © 2016-2022 Altova GmbH



3. Wählen Sie **Projects from Folder or Archive** und klicken Sie auf **Next**.



4. Geben Sie als Verzeichnis **C:\UModelDemo** ein und klicken Sie auf **Finish**.
5. Klicken Sie mit der rechten Maustaste im Paket-Explorer von Eclipse auf das Paket **com.nanonull** und wählen Sie im Kontextmenü den Befehl **New | Class**.
6. Geben Sie einen Klassennamen ein (In diesem Beispiel "MainClass") und aktivieren Sie das Kontrollkästchen **public static void main...**.



7. Klicken Sie im Menü **Run** auf **Run**.

Sie haben nun das mit UModel generierte Java-Projekt kompiliert. Die kompilierten .class-Dateien sollten sich im Unterverzeichnis **bin** Ihres Projektverzeichnisses befinden.

Machen Sie sich eine Notiz, welche Java-Version Sie für die Kompilierung verwendet haben - Sie benötigen diese Version, wenn Sie beabsichtigen, die Binärtypen später zu importieren. Wenn Sie die Eigenschaften Ihres Eclipse-Projekts nicht geändert haben, wurde es wahrscheinlich mit der Standard-Java-Version für Eclipse kompiliert. Um die Standard-Java-Version zu sehen, gehen Sie in Eclipse folgendermaßen vor:

1. Klicken Sie im Menü **Window** auf **Preferences**.
2. Klicken Sie auf **Java** und anschließend auf **Installed JREs**.

Importieren von Java .class-Dateien

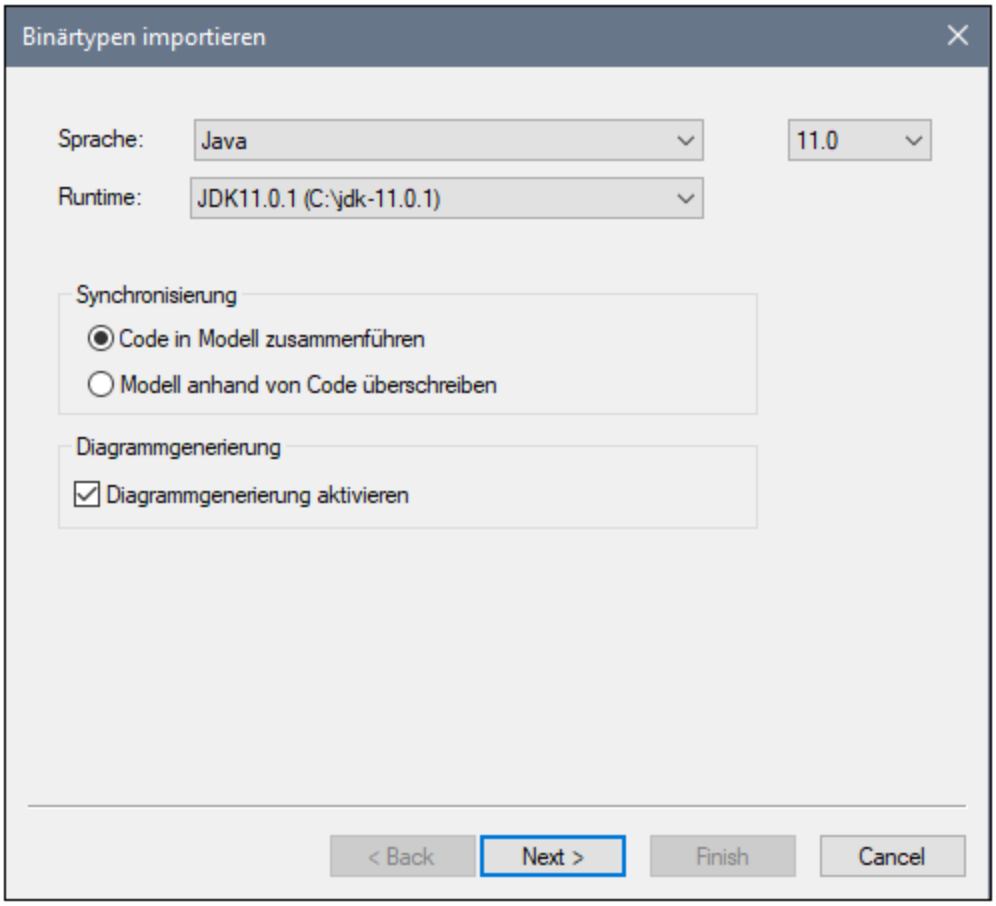
Wenn Sie bereits .class-Binärdateien wie die zuvor kompilierten zur Verfügung haben, können Sie diese nun in UModel importieren.

1. Erstellen Sie ein neues UModel-Projekt oder öffnen Sie ein vorhandenes. Wir importieren Binärtypen in diesem Beispiel in ein neues Projekt.
2. Wenn Ihr Projekt die Java JDK-Typen noch nicht enthält, gehen Sie folgendermaßen vor:

   a. Klicken Sie im Menü **Projekt** auf **Unterprojekt inkludieren**.
   b. Klicken Sie auf das Register **Java** und wählen Sie **Java JDK (types only)**.
   c. Wählen Sie **Durch Referenz** aus, wenn Sie gefragt werden.

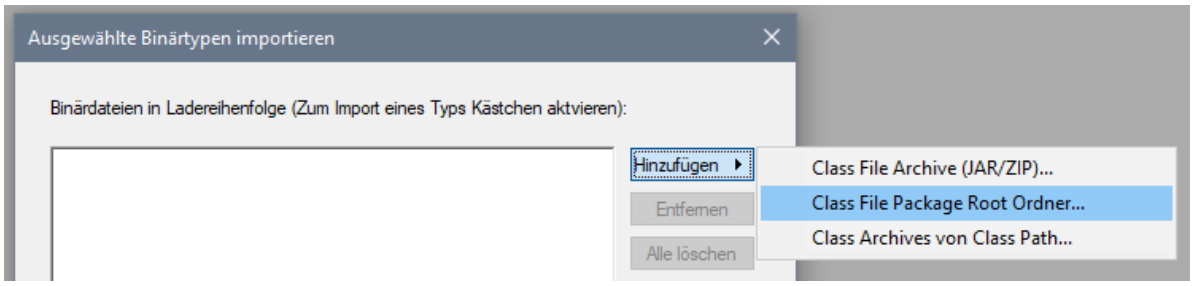
Anmerkung: Dies ist ein optionaler Schritt, mit dem Sie normalerweise vermeiden, dass das Paket "Unbekannte externe Elemente" nach abgeschlossenem Import im Projekt angezeigt wird.

3. Klicken Sie im Menü **Projekt** auf **Binärtypen importieren.**
4. Wählen Sie unter Sprache **Java** sowie die Java-Version, mit der der Java-Code kompiliert wurde, aus (z.B. 11.0).
5. Wählen Sie die Java Runtime aus, die von UModel zum Extrahieren von Informationen aus den Binärdateien verwendet werden soll (die so genannte "Reflection"). Die Runtime-Version muss der im vorherigen Schritt ausgewählten Java-Version entsprechen oder neuer sein.



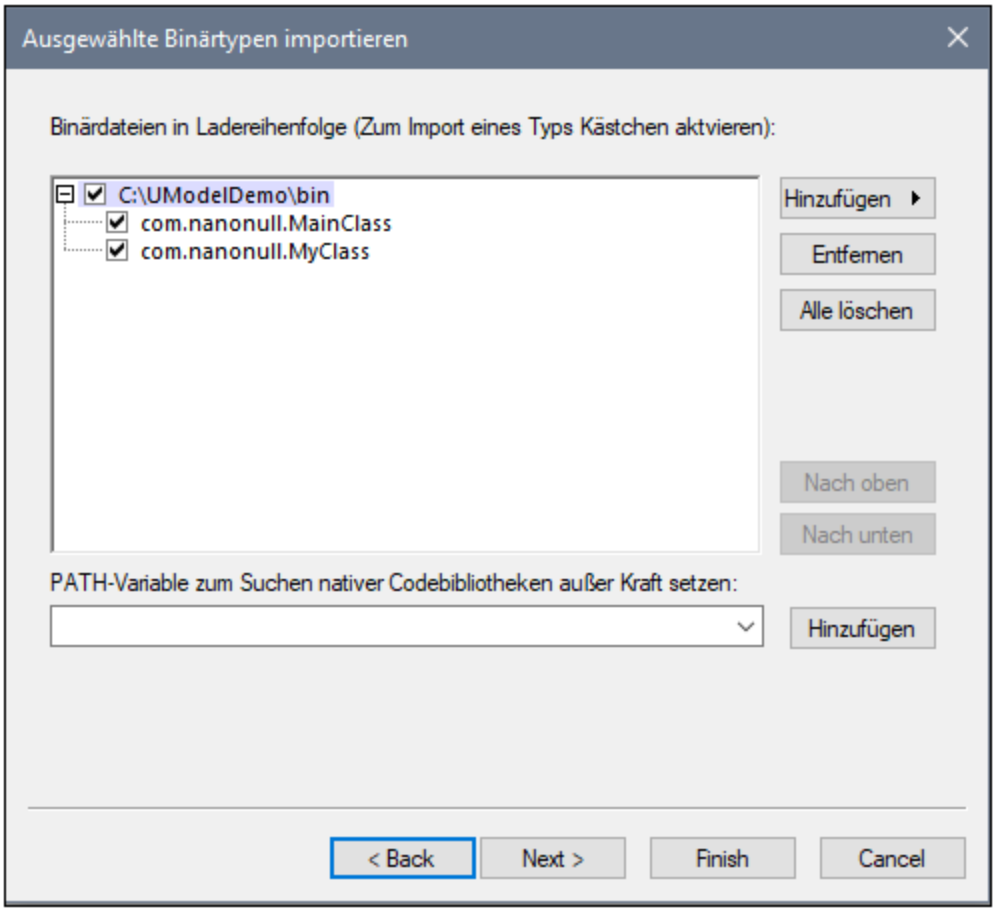
Anmerkung: Die **Runtime** Dropdown-Liste enthält nur automatisch ermittelte Java JDKs und JREs. Wenn Ihr JDK oder JRE nicht in der Liste aufscheint, wählen Sie den Eintrag **Java Runtime-Benutzerpfade bearbeiten** und navigieren Sie zu dem Verzeichnis, in dem die entsprechende Distribution auf Ihrem Rechner installiert wurde, siehe Hinzufügen von benutzerdefinierten Java Runtimes 207.

6. Wenn Sie Binärtypen in ein neues Projekt importieren, wählen Sie entweder **Code in Modell zusammenführen** oder **Modell anhand von Code überschreiben** aus. Wählen sie andernfalls **Code in Modell zusammenführen** aus.
7. Um anhand der importierten Binärtypen optional Klassen- und Paketdiagramme zu generieren, aktivieren Sie das Kontrollkästchen **Diagrammgenerierung aktivieren**. Wenn Sie diese Option auswählen, stehen in den darauf folgenden Schritten weitere Diagrammgenerierungsoptionen zur Verfügung, siehe auch Generieren von Klassendiagrammen 407 und Generieren von Paketdiagrammen 418.
8. Klicken Sie auf **Weiter**.

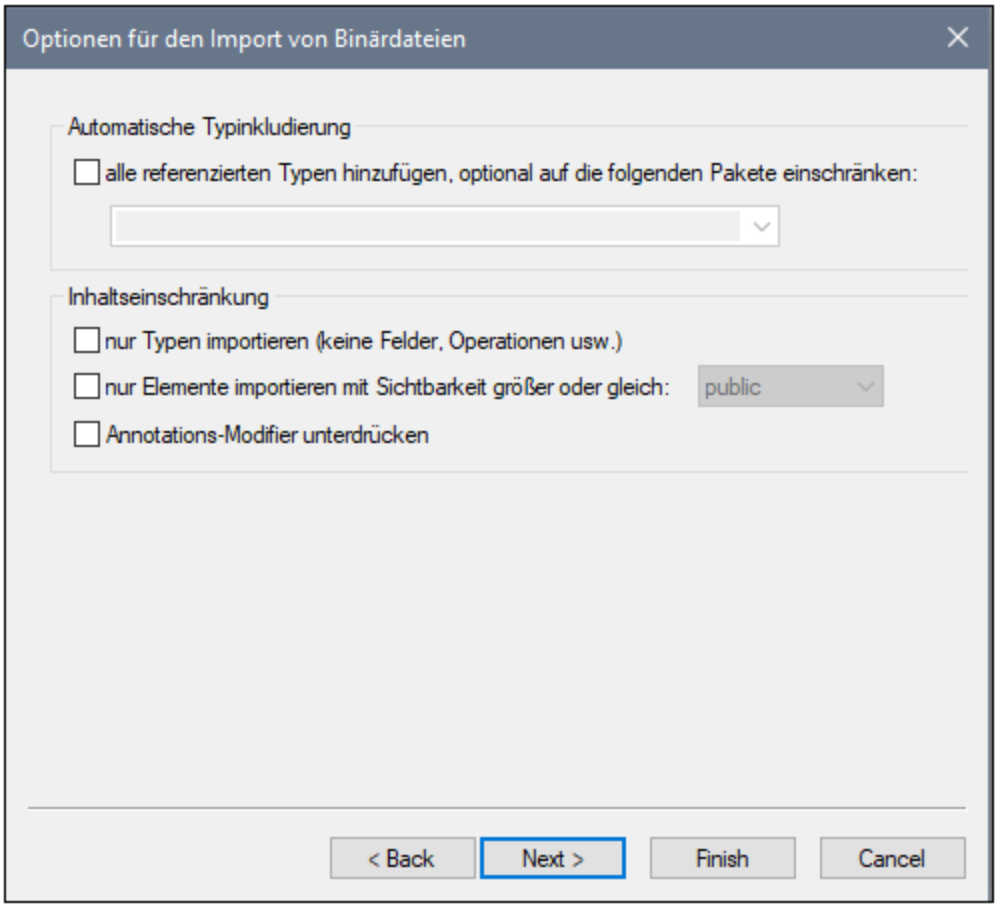


9. Wir importieren in diesem Beispiel Java .class-Dateien aus einer Paket-Root. Wählen Sie die Option **Hinzufügen | Class File Package Root Ordner** und navigieren Sie zum Verzeichnis **C:\UModelDemo\bin**. Falls dieses Verzeichnis fehlt, muss zuerst das Projekt kompiliert werden, wie im ersten Teil dieses Tutorials beschrieben.

 © 2016-2022 Altova GmbH



10. Wählen Sie die gewünschten Klassen aus und klicken Sie auf **Weiter**.



11. Wählen Sie die entsprechenden Importoptionen aus, siehe Optionen für den Import von Binärdateien 208.
12. Wenn Sie in einem früheren Schritt die Diagrammgenerierung aktiviert haben, klicken Sie auf Weiter und konfigurieren Sie die entsprechenden Optionen dafür. Klicken Sie andernfalls auf **Fertig stellen**.

UModel führt die Konvertierung durch und zeigt im Fenster **Meldungen** die Fortschritte an. Wenn die Konvertierung von Binärtypen nicht durchgeführt werden kann, finden Sie in der Fehlermeldung eventuell zusätzliche Informationen dazu. So kann es z.B. vorkommen, dass die Binärdatei, die Sie versuchen zu importieren, für eine Runtime-Version gedacht ist, die neuer als die von Ihnen im Dialogfeld **Binärtypen importieren** ausgewählte ist. Wählen Sie in diesem Fall eine neuere Runtime-Version aus und versuchen Sie es erneut.

 © 2016-2022 Altova GmbH

6.5 Synchronisieren von Modell und Quellcode

Sie können Modell und Code in beiden Richtungen und auf unterschiedlichen Ebenen (z.B. Projekt, Paket oder Klasse) synchronisieren.

> Wenn UModel (Enterprise oder Professional) als Eclipse- oder Visual Studio Plug-in ausgeführt wird, erfolgt die Synchronisierung zwischen Modell und Code automatisch. Eine manuelle Synchronisierung ist auf Projektebene möglich; die Option zum Aktualisieren einzelner Klassen oder Pakete steht nicht zur Verfügung.

Bei Rechtsklick auf ein Element in der Modellstruktur (z.B. eine Klasse) werden im Kontextmenü unter dem Eintrag **Code Engineering** die Befehle zur Synchronisierung oder zum Zusammenführen von Code angezeigt:

- **Merge Programmcode von UModel Projekt *****
- **Merge UModel Projekt von Programmcode *****

> *** ist je nach aktueller Auswahl ein Projekt, Paket, eine Komponente oder eine Klasse usw.

Je nachdem, welche Einstellungen Sie unter **Projekt | Synchronisierungseinstellungen** definiert haben, können die Namen dieser beiden Befehle auch folgendermaßen lauten:

- **Überschreibe Programmcode von UModel Projekt *****
- **Überschreibe UModel Projekt von Programmcode *****

Um das gesamte Projekt (nicht aber Klassen, Pakete oder andere lokale Elemente) zu aktualisieren, können Sie auch die beiden folgenden Befehle aus dem **Projekt**-Menü von UModel verwenden:

- **Merge (oder Überschreibe) Programmcode aus UModel Projekt**
- **Merge (oder Überschreibe) UModel Projekt aus Programmcode**

Aus Gründen der Einfachheit werden die oben aufgelisteten Befehle in diesem Kapitel allgemein als Codesynchronisierungsbefehle bezeichnet.

Wählen Sie eine der folgenden Methoden, um eine Synchronisierung auf Projekt- oder Root-Paketebene durchzuführen:

- Klicken Sie mit der rechten Maustaste auf das Root-Paket in der Modell-Struktur und wählen Sie den gewünschten Codesynchronisierungsbefehl aus.
- Klicken Sie im Menü **Projekt** auf den gewünschten Codesynchronisierungsbefehl.

So führen Sie eine Synchronisierung auf Paketebene durch:

1. Drücken Sie Umschalt + Klick oder Strg + Klick, um das/die gewünschte(n) Paket(e) auszuwählen
2. Klicken Sie mit der rechten Maustaste auf die Auswahl und wählen Sie den gewünschten Codesynchronisierungsbefehl aus.

So führen Sie eine Synchronisierung auf Klassenebene durch:

1. Drücken Sie Umschalt + Klick oder Strg + Klick, um die gewünschte(n) Klasse(n) auszuwählen
2. Klicken Sie mit der rechten Maustaste auf die Auswahl und wählen Sie den gewünschten Codesynchronisierungsbefehl aus.

Um beim Synchronisieren von Modell und Code unerwünschte Ergebnisse zu vermeiden, betrachten Sie bitte die folgenden Szenarien:

Auswahl des Befehls **Überschreibe UModel Projekt aus Programmcode** im Menü **Projekt**	• Dabei werden alle Verzeichnisse (Projektdateien) aller verschiedenen von Ihnen in Ihrem Projekt definierten Codesprachen überprüft. • Neue Dateien werden ermittelt und zum Projekt hinzugefügt. • In Ihrem Meldungsfenster erscheint der Eintrag "Quelldateien werden in 'C:\UMTest' gesammelt".
Rechtsklick auf eine Klasse oder Schnittstelle in der Modell-Struktur und Auswahl von **Code Engineering | Überschreibe UModel Klasse von Programmcode**	• Daraufhin wird nur die ausgewählte Klasse (Schnittstelle,...) Ihres Projekts aktualisiert. • Wenn der Quellcode neue oder seit der letzten Synchronisierung geänderte Klassen enthält, werden diese Änderungen nicht zum Modell hinzugefügt.
Rechtsklick auf eine Komponente in der Modell-Struktur (im Component View-Paket) und Auswahl von **Code Engineering | Überschreibe UModel Komponente von Programmcode**	• Daraufhin wird nur das entsprechende Verzeichnis (bzw. die entsprechende Projektdatei) aktualisiert. • Neue Dateien im Verzeichnis (in der Projektdatei) werden ermittelt und zum Projekt hinzugefügt. • In Ihrem Meldungsfenster erscheint der Eintrag "Quelldateien werden in 'C:\UMTest' gesammelt".

Anmerkung:
Unter Umständen wird beim Synchronisieren von Code ein Dialogfeld angezeigt, in dem Sie aufgefordert werden, Ihr UModel-Projekt vor dem Synchronisieren zu aktualisieren. Dies kommt nur bei UModel-Projekten vor, die mit einer Vorversion der letzten Release erstellt wurden. Klicken Sie bitte auf **JA** um Ihr Projekt zu aktualisieren und Ihre Projektdatei zu speichern. Anschließend wird diese Aufforderung nicht mehr angezeigt.

6.5.1 Tipps zur Synchronisierung

Umbenennen von Classifiern und Reverse Engineering

Der unten beschriebenen Vorgang beim Reverse Engineering oder der automatischen Synchronisierung bezieht sich sowohl auf die Standalone-Applikation als auch auf die Plug-In-Versionen (Visual Studio oder Eclipse).

Wenn Sie einen Classifier im Code-Fenster Ihrer Programmierapplikation umbenennen, wird er gelöscht und als neuer Classifier in die **Modell-Struktur** eingefügt.

Der neue Classifier wird nur in diejenigen **Modellierungsdiagramme** eingefügt, die beim Reverse Engineering automatisch erstellt werden, oder wenn ein Diagramm mit dem Befehl **In neuem Diagramm anzeigen |**

 © 2016-2022 Altova GmbH

Inhalt erstellt wird. Der neue Classifier wird an der Standardposition im Diagramm eingefügt. Diese ist wahrscheinlich nicht mit der vorherigen Position identisch.

Siehe auch Refactoring und Synchronisierung von Code 224.

Automatische Generierung von Komponentenrealisierungen

UModel kann in Rahmen der Codegenerierung automatisch Komponentenrealisierungen generieren. Komponentenrealisierungen werden nur generiert, wenn es absolut klar ist, welcher Komponente eine Klasse zugewiesen werden soll:

- Es ist nur eine Visual Studio-Projektdatei im .ump-Projekt vorhanden.
- Es gibt mehrere Visual Studio-Projekte, doch deren Klassen sind im Modell streng getrennt.

So aktivieren Sie die automatische Generierung von Komponentenrealisierungen:

1. Wählen Sie den Menübefehl **Extras | Optionen.**
2. Klicken Sie auf das Register **Code Engineering** und aktivieren Sie die Option **Fehlende Komponentenrealisierungen generieren.**

Automatische Komponentenrealisierungen werden für einen **Classifier** generiert, dem eine (und nur eine) Komponente zugewiesen werden kann.

- ohne Komponentenrealisierung oder
- eine im Namespace einer Codesprache enthaltene Komponente

Die Art, wie die Komponente gesucht wird, ist in den beiden Fällen unterschiedlich:

Komponente, die eine Code-Projektdatei repräsentiert (Eigenschaft "**projectfile**" ist aktiviert)

- wenn es EINE Komponente gibt, die Classifier im Paket, das sie enthält, hat/realisiert
- wenn es EINE Komponente gibt, die Classifier in einem Unterpaket, des Pakets, das sie enthält, hat/realisiert (von oben nach unten)
- wenn es EINE Komponente gibt, die Classifier in einem der übergeordneten Pakete hat/realisiert (von unten nach oben)
- wenn es EINE Komponente gibt, die Classifier in einem Unterpaket eines der übergeordneten Paket hat/realisiert (von oben nach unten)

Komponente, die ein Verzeichnis repräsentiert (Eigenschaft "**directory**" ist aktiviert)

- wenn es EINE Komponente gibt, die Classifier im Paket, das sie enthält, hat/realisiert
- wenn es EINE Komponente gibt, die Classifier in einem der übergeordneten Pakete hat/realisiert (von unten nach oben)

Anmerkungen:

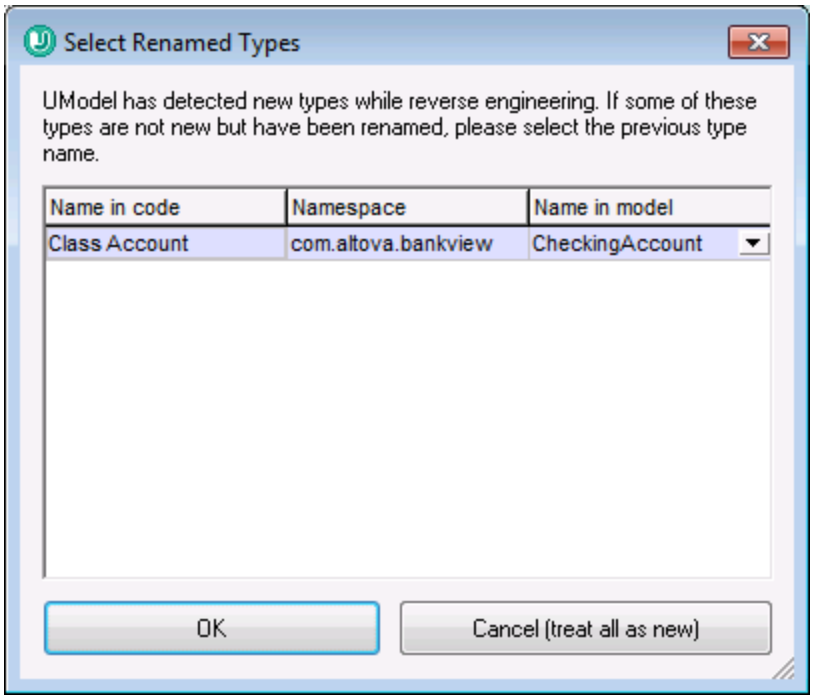
- Die Option "**Code Engineering | Fehlende Komponentenrealisierungen generierten**" muss aktiviert sein.
- Sobald bei einem der obigen Schritte EINE verwendbare Komponente gefunden wird, wird diese Komponente verwendet und die restlichen Schritte werden ignoriert!

Fehler/Warnungen:

- Wenn keine verwendbare Komponente gefunden wird, wird eine Warnmeldung generiert (Meldungsprotokoll)
- Wenn mehrere verwendbare Komponenten gefunden werden, wird eine Fehlermeldung generiert (Meldungsprotokoll)

6.5.2 Refactoring und Synchronisierung von Code

Beim Refactoring von Code müssen oft Klassennamen im Code geändert oder aktualisiert werden. Falls in UModel ab Version 2009 beim Reverse Engineering festgestellt wird, dass neue Typen hinzugekommen oder umbenannt worden sind, wird ein Dialogfeld geöffnet. Die neuen Typen werden in der Spalte "Name im Code" geöffnet, während der vom Programm als ursprünglicher Typname angenommene Name in der Spalte "Name im Modell" aufgelistet wird. UModel ermittelt den Originalnamen anhand von Namespace, Klasseninhalt, Basisklassen und anderen Daten.



Wenn eine Klasse umbenannt wurde, wählen Sie den früheren Klassennamen über die Auswahlliste in der Spalte "Name im Modell" aus, z.B. C1. Damit stellen Sie sicher, dass alle damit in Zusammenhang stehenden Daten beibehalten werden und das Code Engineering korrekt erfolgt.

Ändern von Klassennamen im Modell und Neugenerierung von Code

Wenn Sie ein Modell erstellt haben und anhand dieses Modells Code generiert haben, möchten Sie unter Umständen nochmals Änderungen am Modell vornehmen, bevor Sie die Synchronisierung vornehmen.

Beispiel: Sie haben beschlossen, Klassennamen zu ändern, bevor Sie Code zum zweiten Mal generieren. Da Sie jeder Klasse zuvor einen Dateinamen zugewiesen haben, würde der neue Name der Klasse und der Datei nun im Fenster "Eigenschaften" im Feld "Codedateiname" nicht mehr übereinstimmen.

 © 2016-2022 Altova GmbH

UModel fragt Sie, ob der Name der Codedatei an den neuen Namen der Klasse angepasst werden soll, bevor Sie die Synchronisierung starten. Beachten Sie, dass Sie auch die Möglichkeit haben, auch die Klassenkonstruktoren zu ändern.

Round-Trip Engineering und Beziehungen zwischen Modellelementen:

Beim Aktualisieren des Modells anhand von Code werden Assoziationen zwischen Modellelementen automatisch angezeigt, wenn die Option **Bearbeiten | Assoziationen automatisch erstellen** im Dialogfeld **Extras | Optionen** aktiviert wurde. Assoziationen werden für jene Elemente angezeigt, bei denen der Attributtyp definiert ist und bei denen sich das referenzierte "Typ"-Modellelement im selben Diagramm befindet.

Die Schnittstellenrealisierungen sowie die Generalisierungen werden beim Aktualisieren des Modells anhand von Code automatisch angezeigt.

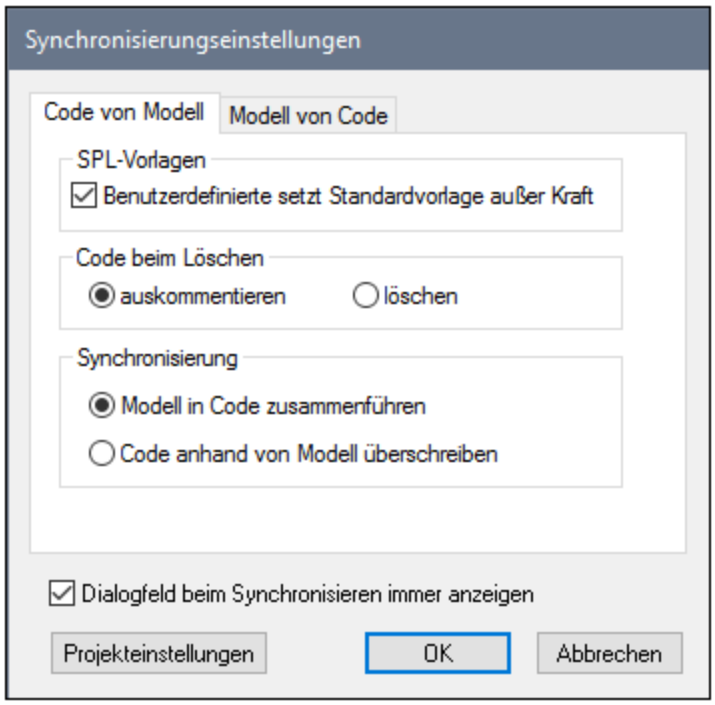
6.5.3 Codesynchronisierungseinstellungen

Die Codesynchronisierungseinstellungen sind in den folgenden Szenarien von Bedeutung:

- Wenn anhand des Modells Programmcode generiert wird (d.h. bei Auswahl des Menübefehls **Projekt | Merge Programmcode aus UModel-Projekt** oder **Projekt | Überschreibe Programmcode aus UModel-Projekt**)
- Wenn Quellcode in das Modell importiert wird (d.h. bei Auswahl des Befehls **Projekt | Merge UModel-Projekt aus Programmcode** oder **Projekt | Überschreibe UModel-Projekt aus Programmcode**)
- Wenn eine automatische Synchronisierung in eine der beiden Richtungen durchgeführt wird (Dies gilt für die UModel Enterprise und die Professional Edition, wenn UModel als Visual Studio- oder Eclipse Plug-in ausgeführt wird).

So ändern Sie die Codesynchronisierungseinstellungen:

- Klicken Sie im Menü **Projekt** auf **Synchronisierungseinstellungen**.



Dialogfeld "Synchronisierungseinstellungen"

Standardmäßig wird das Dialogfeld "Synchronisierungseinstellungen" automatisch jedes Mal angezeigt, wenn Sie einen der Befehle zur Codesynchronisierung auswählen. Damit das Dialogfeld nicht mehr angezeigt wird, deaktivieren Sie das Kontrollkästchen **Dialogfeld beim Synchronisieren immer anzeigen**.

Die verfügbaren Optionen sind auf zwei Registern gruppiert:

- **Code von Modell** (die Optionen auf diesem Register gelten für die Generierung von Programmcode anhand des Modells)
- **Modell von Code** (die Optionen auf diesem Register gelten für den Import von Programmcode in das Modell).

Option	Beschreibung
SPL-Vorlagen	Diese Option ist nur bei Generierung von Programmcode anwendbar. Aktivieren Sie das Kontrollkästchen **Benutzerdefinierte setzt Standardvorlage außer Kraft,** wenn Sie benutzerdefinierte Spy Programming Language (SPL)-Vorlagen erstellt haben, die die mit UModel bereitgestellten Standardvorlagen außer Kraft setzen sollen (siehe auch SPL-Vorlagen 192).
Code beim Löschen	Diese Option ist nur bei Generierung von Programmcode anwendbar. Wählen Sie aus, ob Programmcode bei der Synchronisierung gelöscht oder auskommentiert werden soll (vorausgesetzt die entsprechenden Objekte sind im Modell nicht mehr vorhanden).

 © 2016-2022 Altova GmbH

Option	Beschreibung
Synchronisierung	Diese Option ist sowohl auf die Generierung als auch auf den Import von Programmcode anwendbar. Sie können damit festlegen, ob Änderungen zusammengeführt oder überschrieben werden sollen. Angenommen, es wurde einmal Code anhand eines Modells generiert und es wurden sowohl am Modell als auch am Code Änderungen vorgenommen, z.B.: • wurde eine neue Klasse X in UModel hinzugefügt • wurde eine neue Klasse Y zum externen Code hinzugefügt. **Modell in Code zusammenführen** bedeutet, dass • die im externen Code neu hinzugefügte Klasse Y beibehalten wird • die über UModel neu hinzugefügte Klasse X zum Code hinzugefügt wird. **Code anhand von Modell überschreiben** bedeutet, dass • die neu hinzugefügte Klasse Y im externen Code gelöscht (oder, je nach aktueller Einstellung, auskommentiert) wird • die neu hinzugefügte Klasse X aus UModel zum Code hinzugefügt wird. **Code in Modell zusammenführen** bedeutet, dass • die neu hinzugefügte Klasse X in UModel beibehalten wird • die neu hinzugefügte Klasse Y aus dem externen Code zum Modell hinzugefügt wird **Modell anhand von Code überschreiben** bedeutet, dass • die neu hinzugefügte Klasse X in UModel gelöscht (oder, je nach aktueller Einstellung, auskommentiert) wird • die neu hinzugefügte Klasse Y aus dem externen Code zum Modell hinzugefügt wird.
Projekteinstellungen	Öffnet das Dialogfeld "Projekteinstellungen", wo Sie die Code Engineering-Einstellungen für die einzelnen Sprachen ändern können. Nähere Informationen zu allen Einstellungen finden Sie unter Optionen für den Code-Import 196 bzw. Codegenerierungsoptionen 176. Sie können das Dialogfeld "Projekteinstellungen" auch über den Menübefehl **Projekt | Projekteinstellungen** aufrufen. Beachten Sie, dass die Projekteinstellungen in diesem Dialogfeld global sind (Sie werden zusammen mit dem Projekt gespeichert und von jedem Rechner, auf dem das UModel-Projekt geöffnet wird, angewendet), während die Optionen, die Sie über das Menü **Extras | Optionen** definieren, lokal sind (Sie sind nur in der aktuellen Installation von UModel anwendbar).

6.6 UModel-Elemententsprechungen

In diesem Abschnitt sehen Sie die UModel-Elementenentsprechungen für Elemente (Konstrukte) in verschiedenen Programmiersprachen (C#, Java, VB.NET) sowie Datenbanken und XML-Schemas. Die Zuordnungen sind nach Sprache gruppiert und gelten für den Import von Code in das Modell oder die Generierung von Code anhand des Modells.

- C#-Entsprechungen 228
- VB.NET-Entsprechungen 248
- Java-Entsprechungen 263
- XML Schema-Entsprechungen 268

6.6.1 C#-Entsprechungen

In der Tabelle unten sehen Sie die 1:1-Entsprechungen zwischen:

- UModel-Elementen und C#-Codeelementen bei der Ausgabe von Modell in Code
- C#-Codeelementen und UModel-Modellelementen beim Import von Code in das Modell

C#-Projekt

C#		UModel	
Projekt	Projektdatei	Projektdatei	Komponente
	Verzeichnis	Verzeichnis	

C# Namespace

C#		UModel	
Namespace	Name	Name	Paket <<namespace>>

C#-Klasse

C#			UModel		
Klasse	Name		Name		Klasse
	modifiers	internal	Sichtbarkeit	Paket	
		protected internal		protected <<internal>>	
		public		public	
		protected		protected	
		private		private	
		sealed	leaf		

© 2016-2022 Altova GmbH

C#				UModel		
		abstract		abstract		
		static		<<static>>		
		unsafe		<<unsafe>>		
		partial		<<partial>>		
		new		<<new >>		
	Dateiname			Codedateiname		
	verknüpfte Projektdatei / Verzeichnis			Komponentenrealisierung		
	base types			Generalisierung, Schnittstellenrealisierung(en)		
	attribute sections			<<attributes>>		
	doc comments			Kommentar(->Dokumentation)		
	Feld	Name		Name		Eigenschaft
		modifiers	internal	Sichtbarkeit	Sichtbarkeit	
			protected internal		protected <<internal>>	
			public		public	
			protected		protected	
			private		private	
			static	static		
			readonly	readonly		
			volatile	<<volatile>>		
			unsafe	<<unsafe>>		
			new	<<new >>		
		type		Typ		
		type dimensions		Multiplizität		
		type pointer		Typ-Modifier		
		nullable		<<nullable>>		
		default value		default		
		attribute sections		<<attributes>>		
		doc comments		Kommentar(->Dokumentation)		
	Konstante	Name		Name		Eigenschaft

C#				UModel			
		modifiers	internal	Sichtbarkeit	Paket		
			protected internal		protected <<internal>>		
			public		public		
			protected		protected		
			private		private		
			new	<<new >>			
		type		Typ			
		type dimensions		Multiplizität			
		type pointer		Typ-Modifier			
		nullable		<<nullable>>			
		default value		default			
		attribute sections		<<attributes>>			
		doc comments		Kommentar(->Dokumentation)			
	Methode	Name		Name		Operation	
		modifiers	internal	Sichtbarkeit	Paket		
			protected internal		protected <<internal>>		
			public		public		
			protected		protected		
			private		private		
			static	static			
			abstract	abstract			
			sealed	leaf			
			override	<<override>>			
			partial	<<partial>>			
			virtual	<<virtual>>			
			new	<<new >>			
			unsafe	<<unsafe>>			
		attribute sections		<<attributes>>			
		doc comments		Kommentar(->Dokumentation)			
		implemented interfaces		implements			

© 2016-2022 Altova GmbH

C#					UModel				
		type			direction	return	Parameter		
		Parameter	Name		Name				
			modifiers	ref	direction	inout			
				out		out			
				params	varArgList				
			type		Typ				
			type dimensions		Multiplizität				
			type pointer		Typ-Modifier				
			this		<<this>>				
			nullable		<<nullable>>				
		Typparameter	Name		Name		Vorlagen-parameter		
			constraint		Einschränkender Classifier				
			predefined constraint	struct	<<ValueTypeConstraint>>				
				Klasse	<<ReferenceTypeConstraint>>				
				new ()	<<ConstructorConstraint>>				
			attribute sections		<<attributes>>				
	Konstruktor	Name			Name			Operation <<constructor>>	
		modifiers	internal		Sichtbarkeit	Paket			
			protected internal			protected <<internal>>			
			public			public			
			protected			protected			
			private			private			
			static		static				
			unsafe		<<unsafe>>				
		attribute sections			<<attributes>>				
		doc comments			Kommentar(->Dokumentation)				
		Parameter	Name		Name		Parameter		
			modifiers	ref	direction	inout			

C#					UModel				
				out		out			
				params	varArgList				
			type		Typ				
			type dimensions		Multiplizität				
			type pointer		Typ-Modifier				
			nullable		<<nullable>>				
	Destruktor	Name			Name			Operation <<destructor>>	
		modifiers	private		Sichtbarkeit	private			
			unsafe		<<unsafe>>				
		attribute sections			<<attributes>>				
		doc comments			Kommentar(->Dokumentation)				
	Eigenschaft	Name			Name			Operation <<property>>	
		modifiers	internal		Sichtbarkeit	Paket			
			protected internal			protected <<internal>>			
			public			public			
			protected			protected			
			private			private			
			static		static				
			abstract		abstract				
			sealed		leaf				
			override		<<override>>				
			virtual		<<virtual>>				
			new		<<new >>				
			unsafe		<<unsafe>>				
		attribute sections			<<attributes>>				
		doc comments			Kommentar(->Dokumentation)				
		type			direction	return	Parameter		
		type dimensions			Multiplizität				
		nullable			<<nullable>>				

© 2016-2022 Altova GmbH

C#					UModel				
		Get Accessor	modifiers	internal	Sichtbarkeit	internal	<<GetAccessor>>		
				protected internal		protected internal			
				protected		protected			
				private		private			
		Set Accessor	modifiers	internal	Sichtbarkeit	internal	<<SetAccessor>>		
				protected internal		protected internal			
				protected		protected			
				private		private			
	Operator	Name			Name			Operation <<operator>>	
		modifiers	public		Sichtbarkeit	public			
			static		static				
			unsafe		<<unsafe>>				
		attribute sections			<<attributes>>				
		doc comments			Kommentar(->Dokumentation)				
		type			direction	return	Parameter		
		Parameter	Name		Name				
			modifier	params	varArgList				
			type		Typ				
			type dimensions		Multiplizität				
			type pointer		Typ-Modifier				
			nullable		<<nullable>>				
	Indexer	Name (="this")			Name (="this")			Operation <<indexer>>	
		modifiers	internal		Sichtbarkeit	package			
			protected internal			protected <<internal>>			
			public			public			
			protected			protected			
			private			private			
			static		static				

C#					UModel				
			abstract		abstract				
			sealed		leaf				
			override		<<override>>				
			virtual		<<virtual>>				
			new		<<new >>				
			unsafe		<<unsafe>>				
		attribute sections			<<attributes>>				
		doc comments			Kommentar(->Dokumentation)				
		type			direction	return	Parameter		
		Parameter	Name		Name				
			modifier	params	varArgList				
			type		type				
			type dimensions		Multiplizität				
			type pointer		Typ-Modifier				
			nullable		<<nullable>>				
		Get Accessor	modifiers	internal	Sichtbarkeit	internal	<<GetAccessor>>		
				protected internal		protected internal			
				protected		protected			
				private		private			
		Set Accessor	modifiers	internal	Sichtbarkeit	internal	<<SetAccessor>>		
				protected internal		protected internal			
				protected		protected			
				private		private			
	Event	Name			Name			Operation <<event>>	
		modifiers	internal		Sichtbarkeit	package			
			protected internal			protected <<internal>>			
			public			public			
			protected			protected			
			private			private			

© 2016-2022 Altova GmbH

C#				UModel				
			static	static				
			abstract	abstract				
			sealed	leaf				
			override	<<override>>				
			virtual	<<virtual>>				
			new	<<new >>				
			unsafe	<<unsafe>>				
		attribute sections		<<attributes>>				
		doc comments		Kommentar(->Dokumentation)				
		type		direction	return	Parameter		
		type dimensions		Multiplizität				
		nullable		<<nullable>>				
		Add Accessor		<<AddRemoveAccessor>>				
		Remove Accessor						
	Typparameter	Name		Name			Vorlagen-parameter	
		constraint		Einschränkender Classifier				
		predefined constraint	struct	<<ValueTypeConstraint>>				
			Klasse	<<ReferenceTypeConstraint>>				
			new ()	<<ConstructorConstraint>>				
		attribute sections		<<attributes>>				

C# Struct

C#			UModel		
Struct	Name		Name		Klasse <<struct>>
	modifiers	internal	Sichtbarkeit	package	
		protected internal		protected <<internal>>	
		public		public	
		protected		protected	
		private		private	

C#				UModel			
		unsafe		<<unsafe>>			
		partial		<<partial>>			
		new		<<new >>			
	Dateiname			Codedateiname			
	verknüpfte Projektdatei / Verzeichnis			Komponentenrealisierung			
	base types			InterfaceRealization(s)			
	attribute sections			<<attributes>>			
	doc comments			Kommentar(->Dokumentation)			
	Feld	Name		Name		Property	
		modifiers	internal	Sichtbarkeit	package		
			protected internal		protected <<internal>>		
			public		public		
			protected		protected		
			private		private		
			static	static			
			readonly	readonly			
			volatile	<<volatile>>			
			unsafe	<<unsafe>>			
			new	<<new >>			
		type		type			
		type dimensions		Multiplizität			
		type pointer		Typ-Modifier			
		nullable		<<nullable>>			
		default value		default			
		attribute sections		<<attributes>>			
		doc comments		Kommentar(->Dokumentation)			
	Konstante	Name		Name		Eigenschaft <<const>>	
		modifiers	internal	Sichtbarkeit	package		
			protected internal		protected <<internal>>		
			public		public		

© 2016-2022 Altova GmbH

C#				UModel			
			protected		protected		
			private		private		
			new	<<new >>			
		type		Typ			
		type dimensions		Multiplizität			
		type pointer		Typ-Modifier			
		nullable		<<nullable>>			
		default value		default			
		attribute sections		<<attributes>>			
		doc comments		Kommentar(->Dokumentation)			
	Fixedsize Buffer	Name		Name		Eigenschaft <<fixed>>	
		modifiers	internal	Sichtbarkeit	package		
			protected internal		protected <<internal>>		
			public		public		
			protected		protected		
			private		private		
			unsafe	<<unsafe>>			
			new	<<new >>			
		type		Typ			
		type pointer		Typ-Modifier			
		nullable		<<nullable>>			
		buffer size		default			
		attribute sections		<<attributes>>			
		doc comments		Kommentar(->Dokumentation)			
	Methode	Name		Name		Operation	
		modifiers	internal	Sichtbarkeit	package		
			protected internal		protected <<internal>>		
			public		public		
			protected		protected		
			private		private		

C#					UModel				
			static		static				
			abstract		abstract				
			sealed		leaf				
			override		<<override>>				
			partial		<<partial>>				
			virtual		<<virtual>>				
			new		<<new >>				
			unsafe		<<unsafe>>				
		attribute sections			<<attributes>>				
		doc comments			Kommentar(->Dokumentation)				
		implemented interfaces			implements				
		type			direction	return	Parameter		
		Parameter	Name		Name				
			modifiers	ref	direction	inout			
				out		out			
				params	varArgList				
			type		type				
			type dimensions		Multiplizität				
			type pointer		Typ-Modifier				
			this		<<this>>				
			nullable		<<nullable>>				
		Typparameter	Name		Name		Vorlagen-parameter		
			constraint		Einschränkender Classifier				
			predefined constraint	struct	<<ValueTypeConstraint>>				
				Klasse	<<ReferenceTypeConstraint>>				
				new ()	<<ConstructorConstraint>>				
			attribute sections		<<attributes>>				

© 2016-2022 Altova GmbH

C#					UModel			
	Konstruktor	Name			Name			Operation <<constructor>>
		modifiers	internal		Sichtbarkeit	package		
			protected internal			protected <<internal>>		
			public			public		
			protected			protected		
			private			private		
			static		static			
			unsafe		<<unsafe>>			
		attribute sections			<<attributes>>			
		doc comments			Kommentar(->Dokumentation)			
		Parameter	Name		Name		Parameter	
			modifiers	ref	direction	inout		
				out		out		
				params	varArgList			
			type		Typ			
			type dimensions		Multiplizität			
			type pointer		Typ-Modifier			
			nullable		<<nullable>>			
	Destruktor	Name			Name			Operation <<destructor>>
		modifiers	private		Sichtbarkeit	private		
			unsafe		<<unsafe>>			
		attribute sections			<<attributes>>			
		doc comments			Kommentar(->Dokumentation)			
	Eigenschaft	Name			Name			Operation <<property>>
		modifiers	internal		Sichtbarkeit	package		
			protected internal			protected <<internal>>		
			public			public		
			protected			protected		
			private			private		

C#					UModel				
			static		static				
			abstract		abstract				
			sealed		leaf				
			override		<<override>>				
			virtual		<<virtual>>				
			new		<<new >>				
			unsafe		<<unsafe>>				
		attribute sections			<<attributes>>				
		doc comments			Kommentar(->Dokumentation)				
		type			direction	return	Parameter		
		type dimensions			Multiplizität				
		nullable			<<nullable>>				
		Get Accessor	modifiers	internal	Sichtbarkeit	internal	<<GetAccessor>>		
				protected internal		protected internal			
				protected		protected			
				private		private			
		Set Accessor	modifiers	internal	Sichtbarkeit	internal	<<SetAccessor>>		
				protected internal		protected internal			
				protected		protected			
				private		private			
	Operator	Name			Name			Operation <<operator>>	
		modifiers	public		Sichtbarkeit	public			
			static		static				
			unsafe		<<unsafe>>				
		attribute sections			<<attributes>>				
		doc comments			Kommentar(->Dokumentation)				
		type			direction	return	Parameter		
		Parameter	Name		Name				

© 2016-2022 Altova GmbH

C#					UModel				
			modifier	params	varArgList				
			type		Typ				
			type dimensions		Multiplizität				
			type pointer		Typ-Modifier				
			nullable		<<nullable>>				
	Indexer	Name (="this")			Name (="this")			Operation <<indexer>>	
		modifiers	internal		Sichtbarkeit	package			
			protected internal			protected <<internal>>			
			public			public			
			protected			protected			
			private			private			
			static		static				
			abstract		abstract				
			sealed		leaf				
			override		<<override>>				
			virtual		<<virtual>>				
			new		<<new >>				
			unsafe		<<unsafe>>				
		attribute sections			<<attributes>>				
		doc comments			Kommentar(->Dokumentation)				
		type			direction	return	Parameter		
		Parameter	Name		Name				
			modifier	params	varArgList				
			type		Typ				
			type dimensions		Multiplizität				
			type pointer		Typ-Modifier				
			nullable		<<nullable>>				
		Get Accessor	modifiers	internal	Sichtbarkeit	internal	<<GetAccessor>>		
				protected internal		protected internal			

C#					UModel				
				protected		protected			
				private		private			
		Set Accessor	modifiers	internal	Sichtbarkeit	internal	<<SetAccessor>>		
				protected internal		protected internal			
				protected		protected			
				private		private			
	Event	Name			Name			Operation <<event>>	
		modifiers	internal		Sichtbarkeit	package			
			protected internal			protected <<internal>>			
			public			public			
			protected			protected			
			private			private			
			static		static				
			abstract		abstract				
			sealed		leaf				
			override		<<override>>				
			virtual		<<virtual>>				
			new		<<new >>				
			unsafe		<<unsafe>>				
		attribute sections			<<attributes>>				
		doc comments			Kommentar(->Dokumentation)				
		type			direction	return	Parameter		
		type dimensions			Multiplizität				
		nullable			<<nullable>>				
		Add Accessor			<<AddRemoveAccessor>>				
		Remove Accessor							
	Typparameter	Name			Name			Vorlagen-parameter	
		constraint			Einschränkender Classifier				
		predefined	struct		<<ValueTypeConstraint>>				

© 2016-2022 Altova GmbH

C#				UModel		
		constraint	Klasse	<<ReferenceTypeConstraint>>		
			new ( )	<<ConstructorConstraint>>		
		attribute sections		<<attributes>>		

C#-Schnittstelle

C#				UModel			
Schnittstelle	Name			Name			Schnittstelle
	modifiers	internal		Sichtbarkeit	package		
		protected internal			protected <<internal>>		
		public			public		
		protected			protected		
		private			private		
		unsafe		<<unsafe>>			
		partial		<<partial>>			
		new		<<new >>			
	Dateiname			Codedateiname			
	verknüpfte Projektdatei / Verzeichnis			Komponentenrealisierung			
	base types			Generalisierung(en)			
	attribute sections			<<attributes>>			
	doc comments			Kommentar(->Dokumentation)			
	Methode	Name		Name			Operation
		modifiers	public	Sichtbarkeit	public		
			new	<<new >>			
			unsafe	<<unsafe>>			
		attribute sections		<<attributes>>			
		doc comments		Kommentar(->Dokumentation)			
		type		direction	return	Parameter	
		Parameter	Name	Name			

C#					UModel				
			modifiers	ref	direction	inout			
				out		out			
				params	varArgList				
			type		Typ				
			type dimensions		Multiplizität				
			type pointer		Typ-Modifier				
			this		<<this>>				
			nullable		<<nullable>>				
		Typparameter	Name		Name		Vorlagen-parameter		
			constraint		Einschränkender Classifier				
			predefined constraint	struct	<<ValueTypeConstraint>>				
				Klasse	<<ReferenceTypeConstraint>>				
				new ()	<<ConstructorConstraint>>				
			attribute sections		<<attributes>>				
	Property	Name			Name			Operation <<property>>	
		modifiers	public		Sichtbarkeit	public			
			new		<<new >>				
			unsafe		<<unsafe>>				
		attribute sections			<<attributes>>				
		doc comments			Kommentar(->Dokumentation)				
		type			direction	return	Parameter		
		type dimensions			Multiplizität				
		nullable			<<nullable>>				
		Get Accessor	modifiers	internal	Sichtbarkeit	internal	<<GetAccessor>>		
				protected internal		protected internal			
				protected		protected			
				private		private			

C#					UModel				
		Set Accessor	modifiers	internal	Sichtbarkeit	internal	<<SetAccessor>>		
				protected internal		protected internal			
				protected		protected			
				private		private			
	Indexer	Name (="this")			Name (="this")			Operation <<indexer>>	
		modifiers	public		Sichtbarkeit	public			
			new		<<new >>				
			unsafe		<<unsafe>>				
		attribute sections			<<attributes>>				
		doc comments			Kommentar(->Dokumentation)				
		type			direction	return	Parameter		
		Parameter	Name		Name				
			modifier	params	varArgList				
			type		type				
			type dimensions		Multiplizität				
			type pointer		Typ-Modifier				
			nullable		<<nullable>>				
		Get Accessor	modifiers	internal	Sichtbarkeit	internal	<<GetAccessor>>		
				protected internal		protected internal			
				protected		protected			
				private		private			
		Set Accessor	modifiers	internal	Sichtbarkeit	internal	<<SetAccessor>>		
				protected internal		protected internal			
				protected		protected			
				private		private			
	Event	Name			Name			Operation <<event>>	
		modifiers	public		Sichtbarkeit	public			

C#				UModel				
			new	<<new >>				
			unsafe	<<unsafe>>				
		attribute sections		<<attributes>>				
		doc comments		Kommentar(->Dokumentation)				
		type		direction	return	Parameter		
		type dimensions		Multiplizität				
		nullable		<<nullable>>				
		Add Accessor		<<AddRemoveAccessor>>				
		Remove Accessor						
	Typparameter	Name		Name			Vorlagen-parameter	
		constraint		Einschränkender Classifier				
		predefined constraint	struct	<<ValueTypeConstraint>>				
			Klasse	<<ReferenceTypeConstraint>>				
			new ()	<<ConstructorConstraint>>				
		attribute sections		<<attributes>>				

C#-Delegat

C#			UModel		
Delegat	Name		Name		Klasse <<delegate>>
	modifiers	internal	Sichtbarkeit	package	
		protected internal		protected <<internal>>	
		public		public	
		protected		protected	
		private		private	
		unsafe	<<unsafe>>		
		new	<<new >>		
	Dateiname		Codedateiname		
	verknüpfte Projektdatei / Verzeichnis		Komponentenrealisierung		
	attribute sections		<<attributes>>		
	doc comments		Kommentar(->Dokumentation)		

© 2016-2022 Altova GmbH

C#					UModel				
	type				direction	return	Parameter	Operation	
	Parameter	Name			Name				
		modifiers	ref		direction	inout			
			out			out			
			params		varArgList				
		type			Typ				
		type dimensions			Multiplizität				
		type pointer			Typ-Modifier				
		nullable			<<nullable>>				
	Typparameter	Name			Name			Vorlagen-parameter	
		constraint			Einschränkender Classifier				
		predefined constraint	struct		<<ValueTypeConstraint>>				
			Klasse		<<ReferenceTypeConstraint>>				
			new ()		<<ConstructorConstraint>>				
		attribute sections			<<attributes>>				

C#-Enum

C#			UModel		
Enum	Name		Name		Enumeration
	modifiers	internal	Sichtbarkeit	package	
		protected internal		protected <<internal>>	
		public		public	
		protected		protected	
		private		private	
		new	<<new >>		

C#			UModel		
	Dateiname		Codedateiname		
	verknüpfte Projektdatei / Verzeichnis		Komponentenrealisierung		
	base type		Typ	<<BaseType>>	
	attribute sections		<<attributes>>		
	doc comments		Kommentar(->Dokumentation)		
	Enum Constant	Name	Name	Enumeration Literal	
		default value	default		
		attribute sections	<<attributes>>		
		doc comments	Kommentar(->Dokumentation)		

Parametrisierter C#-Typ

C#	UModel
Parametrisierter Typ	Anonymes gebundenes Element

6.6.2 VB.NET-Entsprechungen

In der Tabelle unten sehen Sie die 1:1-Entsprechungen zwischen:

- UModel-Elementen und VB.NET-Codeelementen bei der Ausgabe von Modell in Code
- VB.NET-Codeelementen und UModel-Modellelementen beim Import von Code in das Modell

VB.NET			UModel		
Projekt	Projektdatei		Projektdatei		Komponente
	Verzeichnis		Verzeichnis		
Namespace	Name		Name		Paket <<namespace>>
Klasse	Name		Name		Klasse
	modifiers	Friend	Sichtbarkeit	Paket	
		Protected Friend		protected <<Friend>>	
		Public		public	

VB.NET				UModel			
		Protected			protected		
		Private			private		
		NotInheritable		leaf			
		MustInherit		abstract			
		Partial		<<Partial>>			
		Shadows		<<Shadows>>			
	Dateiname			Codedateiname			
	verknüpfte Projektdatei/Verzeichnis			Komponentenrealisierung			
	base types			Generalisierung, InterfaceRealization(s)			
	attribute sections			<<Attributes>>			
	doc comments			Kommentar(->Dokumentation)			
	Feld	Name		Name		Eigenschaft	
		modifiers	Friend	Sichtbarkeit	Paket		
			Protected Friend		protected <<Friend>>		
			Public		public		
			Protected		protected		
			Private		private		
			Shared	static			
			ReadOnly	readonly			
			Shadows	<<Shadows>>			
		type		Typ			
		type dimensions		Multiplizität			
		nullable		<<Nullable>>			
		default value		default			
		attribute sections		<<Attributes>>			
		doc comments		Kommentar(->Dokumentation)			
	Konstante	Name		Name		Eigenschaft <<Const>>	
		modifiers	Friend	Sichtbarkeit	Paket		
			Protected Friend		protected <<Friend>>		
			Public		public		

VB.NET					UModel				
			Protected			protected			
			Private			private			
			Shadows		<<Shadows>>				
		type			Typ				
		type dimensions			Multiplizität				
		nullable			<<Nullable>>				
		default value			default				
		attribute sections			<<Attributes>>				
		doc comments			Kommentar(->Dokumentation)				
	Methode	Name			Name			Operation	
		modifiers	Friend		Sichtbarkeit	Paket			
			Protected Friend			protected <<Friend>>			
			Public			public			
			Protected			protected			
			Private			private			
			Shared		static				
			MustOverride		abstract				
			NotOverridable		leaf				
			Overrides		<<Overrides>>				
			Overridable		<<Overridable>>				
			Partial		<<Partial>>				
			Shadows		<<Shadows>>				
			Overloads		<<Overloads>>				
		attribute sections			<<Attributes>>				
		doc comments			Kommentar(->Dokumentation)				
		implemented interfaces			implements				
		type (function)			direction	return	Parameter		
		Parameter	Name		Name				
			modifiers	ByRef	direction	inout			
				ByVal		in			

© 2016-2022 Altova GmbH

VB.NET					UModel			
				ParamArray	varArgList			
				Optional	default			
			type		Typ			
			type dimensions		Multiplizität			
			nullable		<<Nullable>>			
		Typparameter	Name		Name	Vorlagen-parameter		
			constraint		Einschränkender Classifier			
			predefined constraint	Structure	<<ValueTypeConstraint>>			
				Klasse	<<ReferenceTypeConstraint>>			
				New	<<ConstructorConstraint>>			
			attribute sections		<<Attributes>>			
	Konstruktor	Name			Name		Operation <<Constructor>>	
		modifiers	Friend		Sichtbarkeit	Paket		
			Protected Friend			protected <<Friend>>		
			Public			public		
			Protected			protected		
			Private			private		
			Shared		static			
		attribute sections			<<Attributes>>			
		doc comments			Kommentar(->Dokumentation)			
		Parameter	Name		Name		Parameter	
			modifiers	ByRef	direction	inout		
				ByVal		in		
				ParamArray	varArgList			
				Optional	default			
			type		type			
			type dimensions		Multiplizität			

VB.NET					UModel				
			nullable		<<Nullable>>				
	Eigenschaft	Name			Name			Operation <<Property>>	
		modifiers	Friend		Sichtbarkeit	Paket			
			Protected Friend			protected <<Friend>>			
			Public			public			
			Protected			protected			
			Private			private			
			Default		<<Property>> (Default <= IsDefault)				
			Shared		static				
			MustOverride		abstract				
			NotOverridable		leaf				
			Overrides		<<Overrides>>				
			Overridable		<<Overridable>>				
			Shadows		<<Shadows>>				
			Overloads		<<Overloads>>				
			ReadOnly		<<GetAccessor>> (without <<SetAccessor>>)				
			WriteOnly		<<SetAccessor>> (without <<GetAccessor>>)				
		attribute sections			<<Attributes>>				
		doc comments			Kommentar(->Dokumentation)				
		type			direction	return	Parameter		
		type dimensions			Multiplizität				
		nullable			<<Nullable>>				
		Get Accessor	modifiers	Friend	Sichtbarkeit	Friend	<<GetAccessor>>		
				Protected Friend		Protected Friend			
				Protected		Protected			
				Private		Private			
		Set Accessor	modifiers	Friend	Sichtbarkeit	Friend	<<SetAccessor>>		
				Protected Friend		Protected Friend			

© 2016-2022 Altova GmbH

VB.NET					UModel				
				Protected		Protected			
				Private		Private			
	Operator	Name			Name			Operation <<Operator>>	
		modifiers	Public		Sichtbarkeit	Public			
			Shared		static				
			Narrowing		Name <= Narrowing				
			Widening		Name <= Widening				
		attribute sections			<<Attributes>>				
		doc comments			Kommentar(->Dokumentation)				
		type			direction	return	Parameter		
		Parameter	Name		Name				
			modifier	ByVal	direction	in			
			type		Typ				
			type dimensions		Multiplizität				
			nullable		<<Nullable>>				
	Event	Name			Name			Operation <<Event>>	
		modifiers	Friend		Sichtbarkeit	Paket			
			Protected Friend			protected <<Friend>>			
			Public			public			
			Protected			protected			
			Private			private			
			Shared		static				
			MustOverride		abstract				
			NotOverridable		leaf				
			Overrides		<<Overrides>>				
			Overridable		<<Overridable>>				
			Shadows		<<Shadows>>				
			Overloads		<<Overloads>>				
		kind	without specifying a delegate type		<<Event>> (Type <= Simple)				

VB.NET				UModel				
			with specifying a delegate type	<<Event>> (Type <= Regular)				
			with custom accessors	<<Event>> (Type <= Custom)				
		attribute sections		<<Attributes>>				
		doc comments		Kommentar(->Dokumentation)				
		type		direction	return	Parameter		
		type dimensions		Multiplizität				
		nullable		<<Nullable>>				
	Typparameter	Name		Name			Vorlagen-parameter	
		constraint		Einschränkender Classifier				
		predefined constraint	Structure	<<ValueTypeConstraint>>				
			Klasse	<<ReferenceTypeConstraint>>				
			New	<<ConstructorConstraint>>				
		attribute sections		<<Attributes>>				
Struktur	Name			Name				Klasse <<Structure>>
	modifiers	Friend		Sichtbarkeit	Paket			
		Protected Friend			protected <<Friend>>			
		Public			public			
		Protected			protected			
		Private			private			
		Partial		<<Partial>>				
		Shadows		<<Shadows>>				
	Dateiname			Codedateiname				
	verknüpfte Projektdatei/Verzeichnis			Komponentenrealisierung				
	base types			InterfaceRealization(s)				
	attribute sections			<<Attributes>>				
	doc comments			Kommentar(->Dokumentation)				
	Feld	Name		Name			Eigenschaft	
		modifiers	Friend	Sichtbarkeit	Paket			
			Public		public			

© 2016-2022 Altova GmbH

VB.NET				UModel			
			Private		private		
			Shared	static			
			ReadOnly	readonly			
			Shadows	<<Shadows>>			
		type		type			
		type dimensions		Multiplizität			
		nullable		<<Nullable>>			
		default value		default			
		attribute sections		<<Attributes>>			
		doc comments		Kommentar(->Dokumentation)			
	Konstante	Name		Name		Eigenschaft <<Const>>	
		modifiers	Friend	Sichtbarkeit	Paket		
			Public		public		
			Private		private		
			Shadows	<<Shadows>>			
		type		type			
		type dimensions		Multiplizität			
		nullable		<<Nullable>>			
		default value		default			
		attribute sections		<<Attributes>>			
		doc comments		Kommentar(->Dokumentation)			
	Methode	Name		Name		Operation	
		modifiers	Friend	Sichtbarkeit	Paket		
			Public		public		
			Private		private		
			Shared	static			
			MustOverride	abstract			
			NotOverridable	leaf			
			Overrides	<<Overrides>>			
			Overridable	<<Overridable>>			

VB.NET					UModel				
			Partial		<<Partial>>				
			Shadows		<<Shadows>>				
			Overloads		<<Overloads>>				
		attribute sections			<<Attributes>>				
		doc comments			Kommentar(->Dokumentation)				
		implemented interfaces			implements				
		type (function)			direction	return	Parameter		
		Parameter	Name		Name				
			modifiers	ByRef	direction	inout			
				ByVal		in			
				ParamArray	varArgList				
				Optional	default				
			type		Typ				
			type dimensions		Multiplizität				
			nullable		<<Nullable>>				
		Typparameter	Name		Name		Vorlagen-parameter		
			constraint		Einschränkender Classifier				
			predefined constraint	Structure	<<ValueTypeConstraint>>				
				Klasse	<<ReferenceTypeConstraint>>				
				New	<<ConstructorConstraint>>				
			attribute sections		<<Attributes>>				
	Konstruktor	Name			Name			Operation <<Constructor>>	
		modifiers	Friend		Sichtbarkeit	Paket			
			Public			public			
			Private			private			
			Shared		static				
		attribute sections			<<Attributes>>				

© 2016-2022 Altova GmbH

VB.NET					UModel				
		doc comments			Kommentar(->Dokumentation)				
		Parameter	Name		Name		Parameter		
			modifiers	ByRef	direction	inout			
				ByVal		in			
				ParamArray	varArgList				
				Optional	default				
			type		type				
			type dimensions		Multiplizität				
			nullable		<<Nullable>>				
	Eigenschaft	Name			Name			Operation <<Property>>	
		modifiers	Friend		Sichtbarkeit	Paket			
			Public			public			
			Private			private			
			Shared		static				
			Default		<<Property>> (Default <= IsDefault)				
			MustOverride		abstract				
			NotOverridable		leaf				
			Overrides		<<Overrides>>				
			Overridable		<<Overridable>>				
			Shadows		<<Shadows>>				
			Overloads		<<Overloads>>				
			ReadOnly		<<GetAccessor>> (without <<SetAccessor>>)				
			WriteOnly		<<SetAccessor>> (without <<GetAccessor>>)				
		attribute sections			<<Attributes>>				
		doc comments			Kommentar(->Dokumentation)				
		type			direction	return	Parameter		
		type dimensions			Multiplizität				
		nullable			<<Nullable>>				

VB.NET					UModel				
		Get Accessor	modifiers	Friend	Sichtbarkeit	Friend	<<GetAccessor>>		
				Private		Private			
		Set Accessor	modifiers	Friend	Sichtbarkeit	Friend	<<SetAccessor>>		
				Private		Private			
	Operator	Name			Name			Operation <<Operator>>	
		modifiers	Public		Sichtbarkeit	Public			
			Shared		static				
			Narrowing		Name <= Narrowing				
			Widening		Name <= Widening				
		attribute sections			<<Attributes>>				
		doc comments			Kommentar(->Dokumentation)				
		type			direction	return	Parameter		
		Parameter	Name		Name				
			modifier	ByVal	direction	in			
			type		type				
			type dimensions		Multiplizität				
			nullable		<<Nullable>>				
	Event	Name			Name			Operation <<Event>>	
		modifiers	Friend		Sichtbarkeit	Paket			
			Public			public			
			Private			private			
			Shared		static				
			MustOverride		abstract				
			NotOverridable		leaf				
			Overrides		<<Overrides>>				
			Overridable		<<Overridable>>				
			Shadows		<<Shadows>>				
			Overloads		<<Overloads>>				
		kind	without specifying a delegate type		<<Event>> (Type <= Simple)				

© 2016-2022 Altova GmbH

VB.NET				UModel				
			w ith specifying a delegate type	<<Event>> (Type <= Regular)				
			w ith custom accessors	<<Event>> (Type <= Custom)				
		attribute sections		<<Attributes>>				
		doc comments		Kommentar(->Dokumentation)				
		type		direction	return	Parameter		
		type dimensions		Multiplizität				
		nullable		<<Nullable>>				
	Typparam eter	Name		Name			Vorlagen-parameter	
		constraint		Einschränkender Classifier				
		predefine d constraint	Structure	<<ValueTypeConstraint>>				
			Klasse	<<ReferenceTypeConstraint>>				
			New	<<ConstructorConstraint>>				
		attribute sections		<<Attributes>>				
Schnittstel le	Name			Name				Schnittstel le
	modifiers	Friend		Sichtbarke it	Paket			
		Protected Friend			protected <<Friend>>			
		Public			public			
		Protected			protected			
		Private			private			
		Shadow s		<<Shadow s>>				
	Dateiname			Codedateiname				
	verknüpfte Projektdatei/Verzeichnis			Komponentenrealisierung				
	base types			Generalisierung(en)				
	attribute sections			<<Attributes>>				
	doc comments			Kommentar(->Dokumentation)				
	Methode	Name		Name			Operation	
		modifiers	Public	Sichtbarke it	public			
			Shadow s	<<Shadow s>>				
		attribute sections		<<Attributes>>				

VB.NET					UModel				
		doc comments			Kommentar(->Dokumentation)				
		type (function)			direction	return	Parameter		
		Parameter	Name		Name				
			modifiers	ByRef	direction	inout			
				ByVal		in			
				ParamArray	varArgList				
				Optional	default				
			type		Typ				
			type dimensions		Multiplizität				
			nullable		<<Nullable>>				
		Typparameter	Name		Name		Vorlagen-parameter		
			constraint		Einschränkender Classifier				
			predefined constraint	Structure	<<ValueTypeConstraint>>				
				Klasse	<<ReferenceTypeConstraint>>				
				New	<<ConstructorConstraint>>				
			attribute sections		<<Attributes>>				
	Eigenschaft	Name			Name			Operation <<Property>>	
		modifiers	Public		Sichtbarkeit	public			
			Default		<<Property>> (Default <= IsDefault)				
			Shadows		<<Shadows>>				
			ReadOnly		<<GetAccessor>> (without <<SetAccessor>>)				
			WriteOnly		<<SetAccessor>> (without <<GetAccessor>>)				
		attribute sections			<<Attributes>>				
		doc comments			Kommentar(->Dokumentation)				
		type			direction	return	Parameter		
		type dimensions			Multiplizität				

© 2016-2022 Altova GmbH

VB.NET				UModel			
		nullable		<<Nullable>>			
	Event	Name		Name		Operation <<Event>>	
		modifiers	Public	Sichtbarkeit	public		
			Shadows	<<Shadows>>			
		kind	without specifying a delegate type	<<Event>> (Type <= Simple)			
			with specifying a delegate type	<<Event>> (Type <= Regular)			
		attribute sections		<<Attributes>>			
		doc comments		Kommentar(->Dokumentation)			
		type		direction	return	Parameter	
		type dimensions		Multiplizität			
		nullable		<<Nullable>>			
	Typparameter	Name		Name		Vorlagen-parameter	
		constraint		Einschränkender Classifier			
		predefined constraint	Structure	<<ValueTypeConstraint>>			
			Klasse	<<ReferenceTypeConstraint>>			
			New	<<ConstructorConstraint>>			
		attribute sections		<<Attributes>>			
Delegat	Name			Name			Klasse <<Delegate>>
	modifiers	Friend		Sichtbarkeit	Paket		
		Protected Friend			protected <<Friend>>		
		Public			public		
		Protected			protected		
		Private			private		
		Shadows		<<Shadows>>			
	Dateiname			Codedateiname			
	verknüpfte Projektdatei/Verzeichnis			Komponentenrealisierung			
	attribute sections			<<Attributes>>			
	doc comments			Kommentar(->Dokumentation)			

VB.NET				UModel				
	type			direction	return	Parameter	Operation	
	Parameter	Name		Name				
		modifiers	ByRef	direction	inout			
			ByVal		in			
		type		type				
		type dimensions		Multiplizität				
		nullable		<<Nullable>>				
	Typparameter	Name		Name			Vorlagen-parameter	
		constraint		Einschränkender Classifier				
		predefined constraint	struct	<<ValueTypeConstraint>>				
			Klasse	<<ReferenceTypeConstraint>>				
			new ()	<<ConstructorConstraint>>				
		attribute sections		<<Attributes>>				
Enum	Name			Name				Enumeration
	modifiers	Friend		Sichtbarkeit	Paket			
		Protected Friend			protected <<Friend>>			
		Public			public			
		Protected			protected			
		Private			private			
		Shadows		<<Shadows>>				
	Dateiname			Codedateiname				
	verknüpfte Projektdatei/Verzeichnis			Komponentenrealisierung				
	base type			Typ			<<BaseType>>	
	attribute sections			<<Attributes>>				
	doc comments			Kommentar(->Dokumentation)				
	Enum Constant	Name		Name			Enumeration Literal	
		default value		default				
		attribute sections		<<Attributes>>				
		doc comments		Kommentar(->Dokumentation)				

© 2016-2022 Altova GmbH

VB.NET	UModel
Parametrisierter Typ	Anonymes gebundenes Element

6.6.3 Java-Entsprechungen

In der Tabelle unten sehen Sie die 1:1-Entsprechungen zwischen:

- UModel-Elementen und Java-Codeelementen bei der Ausgabe von Modell in Code
- Java-Codeelementen und UModel-Modellelementen beim Import von Code in das Modell

Java				UModel			
Projekt	Projektdatei			Projektdatei			Komponente
	Verzeichnis			Verzeichnis			
Paket	Name			Name			Paket <<namespace>>
Klasse	Name			Name			Klasse
	modifiers	Paket		visibility	Paket		
		public			public		
		protected			protected		
		private			private		
		abstract		abstract			
		strictfp		<<strictfp>>			
		final		<<final>>			
	Dateiname			Codedateiname			
	verknüpfte Projektdatei/Verzeichnis			Komponentenrealisierung			
	extends clause			Generalization			
	implements clause			Schnittstellenrealisierung(en)			
	java docs			Kommentar(->Dokumentation)			
	Feld	Name		Name		Eigenschaft	
		modifiers	Paket	Sichtbarkeit	Paket		
			public		public		
			protected		protected		
			private		private		

Java					UModel				
			static		static				
			transient		<<transient>>				
			volatile		<<volatile>>				
			final		<<final>>				
		type			Typ				
		type dimensions			Multiplizität				
		default value			default				
		java docs			Kommentar(->Dokumentation)				
	Methode	Name			Name			Operation	
		modifiers	Paket		Sichtbarkeit	Paket			
			public			public			
			protected			protected			
			private			private			
			static		static				
			abstract		abstract				
			final		<<final>>				
			native		<<native>>				
			strictfp		<<strictfp>>				
			synchronized		<<synchronized>>				
		throws clause			Ausnahmeereignisse				
		java docs			Kommentar(->Dokumentation)				
		type			direction	return	Parameter		
		Parameter	Name		Name				
			modifier	final	<<final>>				
			...		varArgList				
			type		Typ				
			type dimensions		Multiplizität				
		Typparameter	Name		Name		Vorlagen-parameter		
			bound		Einschränkender Classifier				

© 2016-2022 Altova GmbH

Java					UModel				
	Construktor	Name			Name			Operation <<constructor>>	
		modifiers	public		Sichtbarkeit	public			
			protected			protected			
			private			private			
		throws clause			Ausnahmeereignisse				
		java docs			Kommentar(->Dokumentation)				
		Parameter	Name		Name		Parameter		
			modifier	final	<<final>>				
			...		varArgList				
			type		Typ				
			type dimensions		Multiplizität				
		Typparameter	Name		Name		Vorlagen-parameter		
			bound		Einschränkender Classifier				
	Typparameter	Name			Name			Vorlagen-parameter	
		bound			Einschränkender Classifier				
Schnittstelle	Name				Name				Schnittstelle
	modifiers	Paket			Sichtbarkeit	Paket			
		public				public			
		protected				protected			
		private				private			
		abstract			abstract				
		strictfp			<<strictfp>>				
	Dateiname				Codedateiname				
	verknüpfte Projektdatei/Verzeichnis				Komponentenrealisierung				
	extends clause				Generalization(s)				
	java docs				Kommentar(->Dokumentation)				
	Feld	Name			Name			Eigenschaft	
		modifiers	public		Sichtbarkeit	public			
			static		static				

Java					UModel				
			final		<<final>>				
		type			type				
		type dimensions			Multiplizität				
		default value			default				
		java docs			Kommentar(->Dokumentation)				
	Methode	Name			Name			Operation	
		modifiers	public		Sichtbarkeit	public			
			abstract		abstract				
		throws clause			Ausnahmeereignisse				
		java docs			Kommentar(->Dokumentation)				
		type			direction	return	Parameter		
		Parameter	Name		Name				
			modifier	final	<<final>>				
			...		varArgList				
			type		Typ				
			type dimensions		Multiplizität				
		Typparameter	Name		Name		Vorlagen-parameter		
			bound		Einschränkender Classifier				
	Typparameter	Name			Name			Vorlagen-parameter	
		bound			Einschränkender Classifier				
Enum	Name				Name				Enumeration
	modifiers	Paket			Sichtbarkeit	Paket			
		public				public			
		protected				protected			
		private				private			
	Dateiname				Codedateiname				
	verknüpfte Projektdatei/Verzeichnis				Komponentenrealisierung				
	java docs				Kommentar(->Dokumentation)				

© 2016-2022 Altova GmbH

Java				UModel				
	Enum Constant	Name		Name			Enumerations-Literal	
	Feld	Name		Name			Eigenschaft	
		modifiers	Paket	Sichtbarkeit	Paket			
			public		public			
			protected		protected			
			private		private			
			static	static				
			transient	<<transient>>				
			volatile	<<volatile>>				
			final	<<final>>				
		type		Typ				
		type dimensions		Multiplizität				
		default value		default				
		java docs		Kommentar(->Dokumentation)				
	Methode	Name		Name			Operation	
		modifiers	Paket	Sichtbarkeit	Paket			
			public		public			
			protected		protected			
			private		private			
			static	static				
			abstract	abstract				
			final	<<final>>				
			native	<<native>>				
			strictfp	<<strictfp>>				
			synchronized	<<synchronized>>				
		throws clause		Ausnahmeereignisse				
		java docs		Kommentar(->Dokumentation)				
		type		direction	return	Parameter		
		Parameter	Name	Name				

Java					UModel			
			modifier	final	<<final>>			
			...		varArgList			
			type		Typ			
			type dimensions		Multiplizität			
		Typparameter	Name		Name	Vorlagen-parameter		
			bound		Einschränkender Classifier			
	Constructor	Name			Name		Operation <<constructor>>	
		modifiers	public		Sichtbarkeit	public		
			protected			protected		
			private			private		
		throws clause			Ausnahmeereignisse			
		java docs			Kommentar(->Dokumentation)			
		Parameter	Name		Name	Parameter		
			modifier	final	<<final>>			
			...		varArgList			
			type		Typ			
			type dimensions		Multiplizität			
		Typparameter	Name		Name	Vorlagen-parameter		
			bound		Einschränkender Classifier			
Parametrisierter Typ					Anonymes gebundenes Element			
Annotation					<<annotations> modifiers			

6.6.4 XML Schema-Entsprechungen

In der Tabelle unten sehen Sie die 1:1-Entsprechungen zwischen:

- UModel-Elementen und XML-Schema-Elementen bei der Ausgabe von Modell in Code
- XML-Schema-Elementen und UModel-Modellelementen beim Import von Code in das Modell

Legende:

 © 2016-2022 Altova GmbH

XSD/UML Element

Stereotypeigenschaft (=Eigenschaftswert)

XSD				UModel			
Dateipfad				Projektdatei			Komponente
Schema	Ziel-Namespace			Name			Paket <<namespace>>
	attributeFormDefault			attributeFormDefault			Klasse <<schema>>
	blockDefault			blockDefault			
	elementFormDefault			elementFormDefault			
	finalDefault			finalDefault			
	version			version			
	xml:lang			xml:lang			
	xmlns			xmlns			
	Annotation	source		source			
		appinfo				Kommentar <<appinfo>>	
		documentation	xml:lang	xml:lang		Kommentar <<documentation>>	
	attributeGroup	Name		Name			Klasse <<attributeGroup>>
		annotation	appinfo			Kommentar <<appinfo>>	
			documentation			Kommentar <<documentation>>	
		attribute	Name	Name	Eigenschaft <<attribute>>		
			form	form			
			use	use			

XSD				UModel			
			ref	type			
			type				
			default	default			
			fixed		fixed		
		attributeGroup	ref	type		Eigenschaft <<attributeGroup>>	
		anyAttribute	namespace	namespace		Eigenschaft <<anyAttribute>>	
			processContents	processContents			
	Attribut	Name		Name			Klasse <<attribute>>
		form		form			
		use		use			
		type		type		Eigenschaft	
		default		default			
		fixed			fixed		
		annotation	appinfo			Kommentar <<appinfo>>	
			documentation			Kommentar <<documentation>>	
		simpleType		Name (= Name der Klasse + "_anonymousType[n]")		DataType <<simpleType>>	
	Element	Name		Name			Klasse <<element>>
		abstract		abstract			
		block		block			
		final		final			
		form		form			
		nillable		nillable			
		type		type		Eigenschaft	

XSD					UModel				
		default			default				
		fixed				fixed			
		substitutionGroup			general		Generalisierung <<substitution>>		
		Annotation	appinfo				Kommentar <<appinfo>>		
			documentation				Kommentar <<documentation>>		
		simpleType			Name (= Name der Klasse + "_anonymousType[n]")		DataType <<simpleType>>		
		complexType			Name (= name of Class + "_anonymousType[n]")		Klasse <<complexType>>		
	group	Name			Name			Klasse <<group>>	
		annotation	appinfo				Kommentar <<appinfo>>		
			documentation				Kommentar <<documentation>>		
		all			Name (= "_all")		Eigenschaft		
					Name (= "mg"_ + "all")		Klasse <<all>>		
			annotation	appinfo		Kommentar <<appinfo>>			
				documentation		Kommentar <<documentation>>			
			element	Name	Name	Eigenschaft			

XSD					UModel				
				ref	type	<<element>>			
				type					
		choice			Name (= "_choice")		Eigenschaft		
					Name (= "mg"_ + "choice")		Klasse <<choice>>		
			annotation	appinfo		Kommentar <<appinfo>>			
				documentation		Kommentar <<documentation>>			
			element	Name	Name	Eigenschaft <<element>>			
				ref	type				
				type					
			group			Eigenschaft <<group>>			
			any	namespace	namespace	Eigenschaft <<any>>			
				processContents	processContents				
			choice			Eigenschaft			
						Klasse <<choice>>			
			sequence			Eigenschaft			
						Klasse <<sequence>>			
		sequence			Name (= "_sequence")		Eigenschaft		
					Name (= "mg"_ + "sequence")		Klasse <<sequen		

© 2016-2022 Altova GmbH

XSD					UModel				
			annotation	appinfo		Kommentar <<appinfo>>			
				documentation		Kommentar <<documentation>>			
			element	Name	Name	Eigenschaft <<element>>			
				ref	type				
				type					
			group			Eigenschaft <<group>>			
			any	namespace	namespace	Eigenschaft <<any>>			
				processContents	processContents				
			choice			Eigenschaft			
						Klasse <<choice>>			
			sequence			Eigenschaft			
						Klasse <<sequence>>			
	notation	Name			Name			DataType <<notation>>	
		system			system				
		public			public				
		annotation	appinfo				Kommentar <<appinfo>>		
			documentation				Kommentar <<documentation>>		

XSD					UModel			
	complexType	Name			Name		Klasse <<complexType>>	
		abstract			abstract			
		block			block			
		final			final			
		mixed			mixed			
		annotation	source		source			
			appinfo			Kommentar <<appinfo>>		
			documentation	xml:lang	xml:lang	Kommentar <<documentation>>		
		group			Name (= "_ref[n]")	Eigenschaft <<group>>		
			maxOccurs		Multiplizität			
			minOccurs					
			ref		type			
		all			Name (= "mg"_ + "all")	Klasse <<all>>		
					Name (= "_all")	Eigenschaft		
			maxOccurs		Multiplizität			
			minOccurs					
		choice			Name (= "mg"_ + "choice[n]")	Klasse <<choice>>		
					Name (= "_choice[n]")	Eigenschaft		
			maxOccurs		Multiplizität			
			minOccurs					
		sequence			Name (= "mg"_ + "sequence[n]")	Klasse <<sequence>>		
					Name (= "_sequence[n]")	Eigenschaft		

© 2016-2022 Altova GmbH

XSD					UModel				
			maxOccurs		Multiplizität				
			minOccurs						
		attribute	Name		Name		Eigenschaft <<attribute>>		
			ref		type				
			type						
		attributeGroup	ref		type		Eigenschaft <<attributeGroup>>		
		anyAttribute	namespace		namespace		Eigenschaft <<anyAttribute>>		
			processContents		processContents				
		complexContent	restriction	base	general		Generalisierung <<restriction>>		
			extension				Generalisierung <<extension>>		
	simpleType	Name			Name			DataType <<simpleType>> Enumeration <<simpleType>>	
		final			final				
		annotation	source		source				
			appinfo				Kommentar <<appinfo>>		
			documentation	xml:lang	xml:lang		Kommentar <<documentation>>		
		list	itemType		Name (= "_itemType")	Eigenschaft <<itemType>>	<<list>>		
			simpleType		DataType <<simpleType>>				
		union	memberTypes		Name (= "memberType[n]")	Eigenschaft <<memberType>>	<<union>>		

XSD					UModel			
			simpleType		DataType <<simpleType>>			
		minExclusive	Wert		Wert	<<minExclusive>>		
			fixed		fixed			
		minInclusive	Wert		Wert	<<minInclusive>>		
			fixed		fixed			
		maxExclusive	Wert		Wert	<<maxExclusive>>		
			fixed		fixed			
		maxInclusive	Wert		Wert	<<maxInclusive>>		
			fixed		fixed			
		totalDigits	Wert		Wert	<<totalDigits>>		
			fixed		fixed			
		fractionDigits	Wert		Wert	<<fractionDigits>>		
			fixed		fixed			
		length	Wert		Wert	<<length>>		
			fixed		fixed			
		minLength	Wert		Wert	<<minLength>>		
			fixed		fixed			
		maxLength	Wert		Wert	<<maxLength>>		
			fixed		fixed			
		whitespace	Wert		Wert	<<whitespace>>		
			fixed		fixed			
		pattern	Wert		Wert	<<whitespace>>		
		enumeration	Wert		Name	Enumerations-literal		
		simpleType				DataType <<simpleType>>		
		restriction	base		general	Generalisierung <<restriction>>		

 © 2016-2022 Altova GmbH

XSD					UModel				
	complexType simpleContent	Name			Name			DataType <<complexType>> <<simpleContent>>	
		annotation	source		source				
			appinfo				Kommentar <<appinfo>>		
			documentation	xml:lang	xml:lang		Kommentar <<documentation>>		
		minExclusive	Wert		Wert		<<minExclusive>>		
			fixed		fixed				
		minInclusive	Wert		Wert		<<minInclusive>>		
			fixed		fixed				
		maxExclusive	Wert		Wert		<<maxExclusive>>		
			fixed		fixed				
		maxInclusive	Wert		Wert		<<maxInclusive>>		
			fixed		fixed				
		totalDigits	Wert		Wert		<<totalDigits>>		
			fixed		fixed				
		fractionDigits	Wert		Wert		<<fractionDigits>>		
			fixed		fixed				
		length	Wert		Wert		<<length>>		
			fixed		fixed				
		minLength	Wert		Wert		<<minLength>>		
			fixed		fixed				
		maxLength	Wert		Wert		<<maxLength>>		
			fixed		fixed				
		whitespace	Wert		Wert		<<whitespace>>		
			fixed		fixed				
		pattern	Wert		Wert		<<whitespace>>		

XSD					UModel				
		attribute	Name		Name		Eigenschaft <<attribute>>		
			ref		type				
			type						
		attributeGroup	ref		type		Eigenschaft <<attributeGroup>>		
		anyAttribute	namespace		namespace		Eigenschaft <<anyAttribute>>		
			processContents		processContents				
		simpleType					DataType <<simpleType>>		
		restriction	base		general		Generalisierung <<restriction>>		
		extension	base		general		Generalisierung <<extension>>		
	import	schemaLocation			schemaLocation			ElementImport <<import>>	
		namespace			namespace				
	include	schemaLocation			schemaLocation			ElementImport <<include>>	
	redefine	schemaLocation			schemaLocation			ElementImport <<redefine>>	
		simpleType			<<redefine>>			DataType <<simpleType>>	
		complexType						Klasse <<complexType>>	
		attributeGroup						Klasse <<attributeGroup>>	

© 2016-2022 Altova GmbH

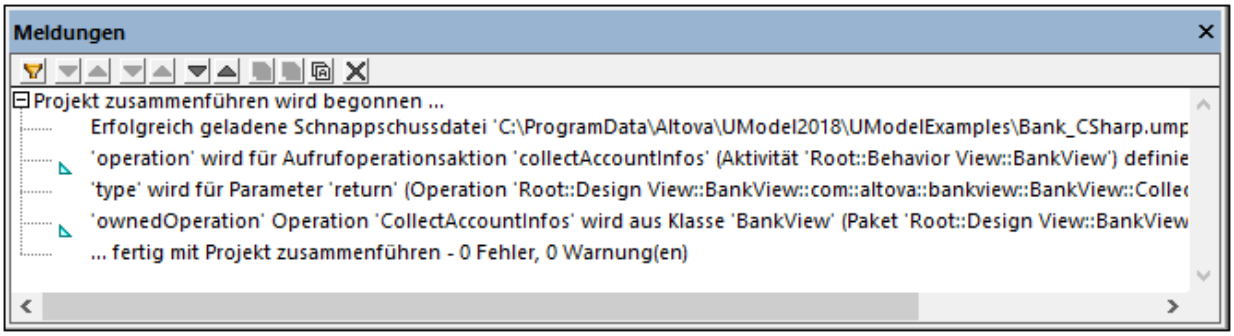
XSD				UModel		
		group			Klasse <<group>>	

6.7 Zusammenführen von UModel-Projekten

UModel unterstützt 2-Weg- und 3-Weg-Projektzusammenführungen. In beiden Fällen werden unterschiedliche UModel-Projektdateien zu einem gemeinsamen UModel *.ump Modell zusammengeführt. Dies ist hilfreich, wenn mehrere Personen gleichzeitig am selben Projekt arbeiten oder wenn Sie Ihre Arbeit einfach in einem einzigen Modell zusammenführen möchten.

So führen Sie zwei UML-Projekte zusammen:

1. Öffnen Sie die UML-Datei, in die das zweite Modell überführt werden soll.
2. Wählen Sie die Menüoption **Projekt | Projekt zusammenführen....**
3. Wählen Sie das zweite UML-Projekt aus, also das Projekt, das in die erste Datei überführt werden soll. Im Meldungsfenster sehen Sie Meldungen über den Ablauf der Zusammenführung und die relevanten Einzelheiten werden protokolliert.



Anmerkung: Wenn Sie auf einen der Einträge im Meldungsfenster klicken, wird das entsprechende Modellierungselement in der Modell-Struktur angezeigt.

Ergebnisse der Zusammenführung:

- Neue Modellierungselemente, d.h. Elemente, die in der Quelle nicht vorhanden sind, werden zum Projekt hinzugefügt.
- Unterschiede in denselben Modellierungselementen; die Elemente aus dem **zweiten** Modell haben Vorrang. So kann es z.B. nur einen Standardwert für ein Attribut geben. Es wird der Standardwert aus der zweiten Datei verwendet.
- Diagrammunterschiede: UModel überprüft zuerst, ob es Unterschiede zwischen den Diagrammen der beiden Modelle gibt.
  Wenn ja, so wird das neue/unterschiedliche Diagramm zum Modell hinzugefügt (und eine fortlaufende Nummerierung wird an den Namen des Diagramms angehängt, z.B: activity1 usw.) und das ursprüngliche Diagramm wird beibehalten.
  Falls keine Unterschiede vorhanden sind, werden identische Diagramme ignoriert und es werden keine Änderungen vorgenommen.
  Sie können anschließend entscheiden, welches der Diagramme Sie beibehalten oder löschen möchten. Natürlich können Sie auch beide Diagramme beibehalten, wenn Sie möchten.
- Die gesamte Zusammenführung kann Schritt für Schritt rückgängig gemacht werden. Klicken Sie dazu in der Symbolleiste auf das Symbol "**Rückgängig**" oder drücken Sie **Strg + Z**.

 © 2016-2022 Altova GmbH

- Wenn Sie im Meldungsfenster auf einen Eintrag klicken, wird das entsprechende Element in der Modell-Struktur angezeigt.
- Der Dateiname der zusammengeführten Datei, also der ersten Datei, die Sie geöffnet haben, wird beibehalten!

6.7.1 3-Weg-Projektzusammenführung

UModel unterstützt nun die Zusammenführung mehrerer UModel-Projekte, die gleichzeitig von mehreren Entwicklern bearbeitet wurden, in einer 3-Weg-Projektzusammenführung.

Die 3-Weg-Projektzusammenführung funktioniert bei UModel-Projekten der **obersten Ebene**, d.h. Projekten die auch Unterprojekte enthalten können. Nicht unterstützt wird die Zusammenführung einzelner **Dateien**, wenn diese Dateien nicht aufgelöste Referenzen auf andere Dateien enthalten.

Bei der Zusammenführung von **Hauptprojekten** werden auch alle editierbaren Unterprojekte automatisch zusammengeführt. Die Unterprojekte müssen nicht separat zusammengeführt werden. Ein Beispiel dazu finden Sie unter Beispiel: Manuelle 3-Weg-Projektzusammenführung 283.

- Die gesamte Projektzusammenführung kann durch Klicken auf die "Rückgängig"-Schaltfläche in der Symbolleiste oder Drücken von **Strg+Z** Schritt für Schritt rückgängig gemacht werden.
- Wenn Sie im Fenster "Meldungen" auf einen Eintrag klicken, wird dieses Element in der Modell-Struktur angezeigt.
- Der Dateiname der zusammengeführten Datei, also der ersten Datei, die Sie geöffnet haben, wird beibehalten.

Ergebnisse der Zusammenführung

Mit "Quelle" wird im Folgenden die Anfangsprojektdatei bezeichnet, also die Datei, die Sie geöffnet haben, bevor Sie mit der Zusammenführung begonnen haben.

- Neue Modellierungselemente in der zweiten Datei, also Elemente, die in der Quelle nicht vorhanden sind, werden zum zusammengeführten Modell hinzugefügt.
- Neue Modellierungselemente in der Quelldatei, also Elemente, die in der zweiten Datei nicht vorhanden sind, bleiben im zusammengeführten Modell erhalten.
- Gelöschte Modellierungselemente aus der zweiten Datei, also Elemente, die in der Quelle noch immer vorhanden sind, werden aus dem zusammengeführten Modell entfernt.
- Gelöschte Modellierungselemente aus der Quelldatei, also Elemente, die in der zweiten Datei noch vorhanden sind, bleiben im zusammengeführten Modell gelöscht.

Unterschiede im selben Modellierungselement:

- Wenn eine Eigenschaft (z.B. die Sichtbarkeit einer Klasse) in der Quelldatei oder der zweiten Datei geändert wurde, so wird im zusammengeführten Modell der aktualisierte Wert verwendet
- Wenn eine Eigenschaft (z.B. die Sichtbarkeit einer Klasse) sowohl in der Quelle als auch in der zweiten Datei geändert wurde, so wird der Wert aus der zweiten Datei verwendet (und im Meldungsfenster unterhalb wird eine Warnmeldung angezeigt)

Verschobene Elemente:

- Wenn ein Element in der Quelle oder der zweiten Datei verschoben wurde, so wird das Element im zusammengeführten Modell ebenfalls verschoben

- Wenn ein Element sowohl in der Quelle als auch in der zweiten Datei (in unterschiedliche Parent-Elemente) verschoben wird, so erscheint eine Meldung, in der Sie aufgefordert werden, das Parent-Element im zusammengeführten Modell manuell auszuwählen.

Diagrammunterschiede:

UModel überprüft zuerst, ob zwischen den Diagrammen der beiden Modelle Unterschiede bestehen. Falls dies der Fall ist, so wird das neue/geänderte Diagramm zum zusammengeführten Modell (mit einer fortlaufenden Nummer, wie activity1 usw.) hinzugefügt und das Original wird beibehalten. Wenn es keine Unterschiede gibt, so werden identische Diagramme ignoriert, d.h. es gibt keine Änderungen. Sie können später entscheiden, welches der Diagramme beibehalten oder gelöscht werden soll. Natürlich können Sie auch beide Diagramme beibehalten.

Versionskontrollunterstützung für 3-Weg-Zusammenführungen

Beim Ein-/Auschecken von Projektdateien generiert UModel automatisch "gemeinsame Vorgängerdateien" (oder Schnappschussdateien), anhand derer anschließend die 3-Weg-Projektzusammenführung durchgeführt wird. Dies ermöglicht viel genauere Zusammenführungsresultate als die normale 2-Weg-Zusammenführung.

Es hängt vom jeweiligen Versionskontrollsystem, das Sie verwenden, ab, ob UModel die automatische 3-Weg-Zusammenführung über die Schnappschussdatei unterstützt. Eine manuelle 3-Weg-Zusammenführung ist jedoch immer möglich.

- Versionskontrollsysteme, die die Dateizusammenführung durchführen, ohne dass der Benutzer darauf Einfluss nehmen kann, werden eine automatische 3-Weg-Zusammenführung wahrscheinlich nicht unterstützen.
- Versionskontrollsysteme, in denen Sie bei einem geänderten Projekt zwischen Ersetzen und Zusammenführen wählen können, unterstützen eine 3-Weg-Zusammenführung im Allgemeinen. Nachdem das Versionskontroll-Plugin die Datei ersetzt hat, wird bei Auswahl des Befehls "Ersetzen" eine UModel-Dateiwarnung aktiviert, die Ihnen dann gestattet, eine 3-Weg-Zusammenführung durchzuführen. Das Ein- und Auschecken muss über UModel erfolgen.
- Haupt- und Unterprojekte können unter Versionskontrolle gestellt werden. Wenn Daten in einem Unterprojekt geändert wurden, werden Sie automatisch gefragt, ob das/die Unterprojekte ausgecheckt werden sollen.
- Bei jedem Ein-/Auscheckvorgang wird eine gemeinsame Vorgänger- oder Schnappschussdatei erstellt, die dann für die 3-Weg-Projektzusammenführung verwendet wird.

Anmerkung: Schnappschussdateien werden nur mit den Standalone-Versionen von UModel automatisch erstellt und verwendet, d.h. diese Funktionen stehen in der Eclipse- und der Visual Studio-Plugin-Version nicht zur Verfügung.

Beispiel

Benutzer A bearbeitet eine UModel-Projektdatei und ändert den Namen einer Klasse im BankView-Hauptdiagramm. Benutzer B öffnet dieselbe Projektdatei und ändert die Sichtbarkeit derselben Klasse.

Anhand der Schnappschussdateien, die für die einzelnen Benutzer erstellt wurden, wird ein Bearbeitungsverlauf erstellt, der eine Zusammenführung der einzelnen Änderungen im Projekt ermöglicht. Sowohl die Änderungen am Namen als auch an der Sichtbarkeit werden bei der 3-Weg-Zusammenführung in der Projektdatei zusammengeführt.

 © 2016-2022 Altova GmbH

6.7.2 Beispiel: Manuelle 3-Weg-Projektzusammenführung

In diesem Beispiel wird eine einfache 3-Weg-Projektzusammenführung gezeigt. Angenommen zwei Personen, Tom und Alice, haben jeweils eine eigene Kopie eines UModel-Projekts erstellt und Änderungen daran vorgenommen. Es gibt nun drei Versionen desselben Projekts: das Originalprojekt, die Kopie von Tom und die Kopie von Alice. Bei einer 3-Weg-Projektzusammenführung bildet das Originalprojekt die "gemeinsame Vorgängerdatei".

Als gemeinsame Vorgängerdatei verwenden wir für dieses Beispiel das Projekt **Bank_CSharp.ump** aus dem Ordner **C:\Benutzer\<Benutzername>\Dokumente\Altova\UModel2023\UModelExamples**. Die Kopien von Tom und Alice müssen manuell erstellt werden. Erstellen wir daher zuerst in Unterordnern des Ordners ... \UModelExamples zwei Kopien des Projekts **Bank_Csharp.ump.** Nennen wir die Unterordner **Alice** und **Tom**; und lassen wir den Projektnamen unverändert.

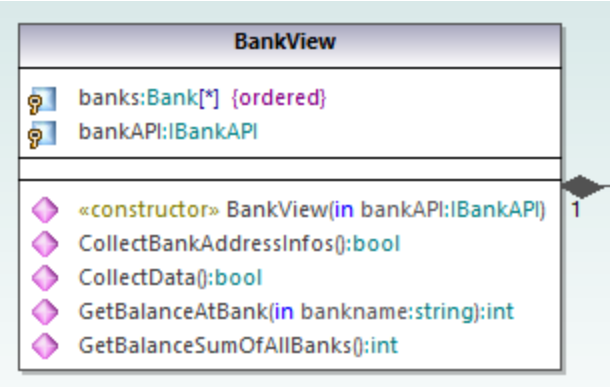
> Verwenden Sie zum Erstellen der Kopien von Tom und Alice den Befehl **Projekt | Speichern unter.** Wenn Sie gefragt werden, ob die relativen Pfade angepasst werden sollen, klicken Sie auf **Ja**. Dadurch vermeiden Sie Syntaxfehler in den Projektkopien.

Wir wollen in diesem Beispiel zeigen, wie Alice Änderungen nicht nur aus dem Originalprojekt **Bank_CSharp.ump,** sondern auch aus Toms Projekt in einem neuen Modell (in einer so genannten 3-Weg-Zusammenführung) zusammenführt.

Schritt 1: Vorbereiten von Toms Projekt

Tom öffnet die Projektdatei **Bank_CSharp.ump** im Ordner **Tom,** öffnet das Diagramm "BankView Main" und nimmt Änderungen an der Klasse `BankView` vor.

1. Die Operation `CollectAccountInfos():bool` wird aus der `BankView`-Klasse gelöscht.
2. Die Sichtbarkeit (**visibility)** der Operation `CollectBankAddressInfos():bool` wird von "protected" in "public" geändert.

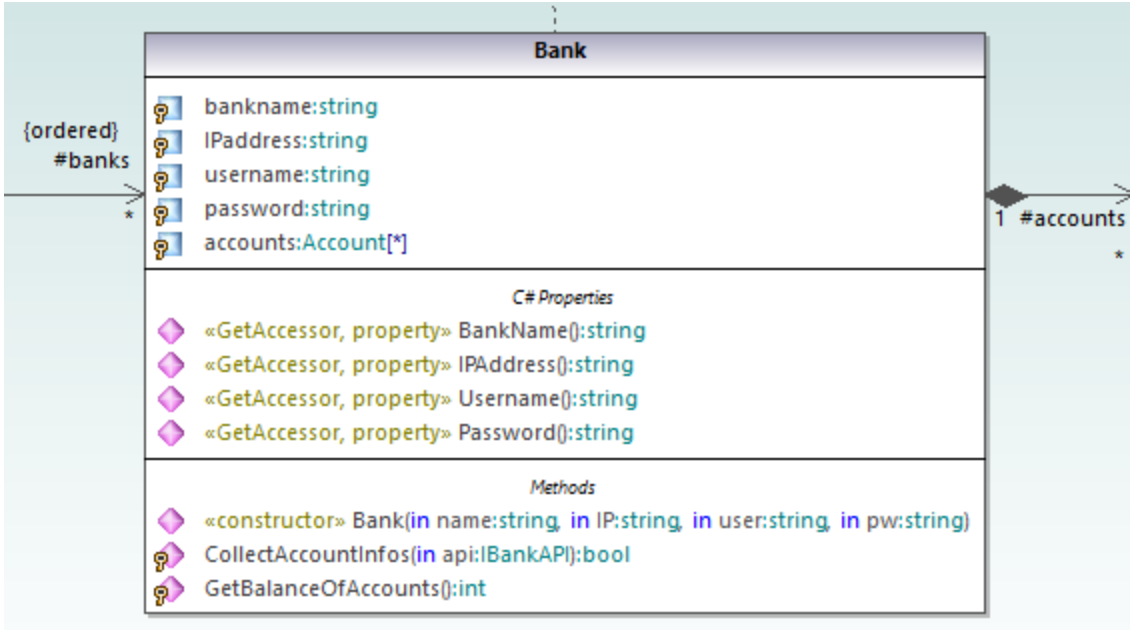


3. Anschließend wird das Projekt gespeichert.

Schritt 2: Vorbereiten des Projekts von Alice

Alice öffnet die Projektdatei **Bank_CSharp.ump** im Ordner **Alice** öffnet das Diagramm "BankView Main" und nimmt Änderungen an der Klasse `Bank` vor.

1. Die Operationen `CollectAccountInfos` und `GetBalanceOfAccounts` werden beide von "public" in "protected" geändert.



2. Anschließend wird das Projekt gespeichert.

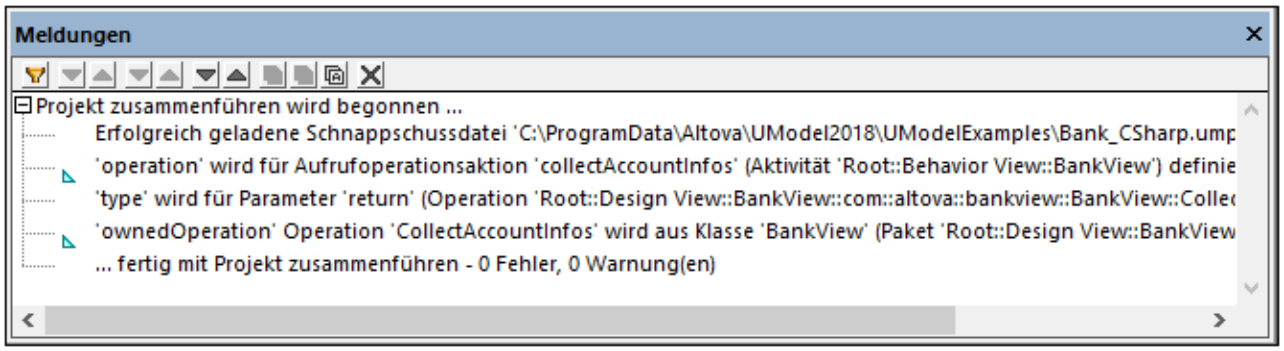
Schritt 3: Durchführung einer 3-Weg-Zusammenführung

Alice beginnt nun eine 3-Weg-Projektzusammenführung:

1. Öffnen Sie das Projekt von Alice aus dem Ordner **Alice.**
2. Wählen Sie im Menü **Projekt** den Befehl **Projekt zusammenführen (3-Weg)** und wählen Sie die von Tom geänderte Projektdatei aus dem Ordner **Tom** aus.
3. Sie werden nun aufgefordert, die gemeinsame Vorgängerdatei zu öffnen. Wählen Sie die Originalprojektdatei **Bank_CSharp.ump** aus dem Ordner **...\UModelExamples** aus.

Die 3-Weg-Zusammenführung wird gestartet und sie kehren zur Projektdatei, von der aus Sie die 3-Weg-Zusammenführung gestartet haben, also zur Projektdatei im Ordner Alice, zurück. Im Fenster "Meldungen" wird die Zusammenführung im Detail angezeigt.

 © 2016-2022 Altova GmbH



Das Resultat der 3-Weg-Zusammenführung ist das folgende:

- Die von Tom am Projekt vorgenommenen Änderungen werden in der Projektdatei von Alice repliziert.
- Die von Alice am Projekt vorgenommenen Änderungen werden in der Projektdatei beibehalten.

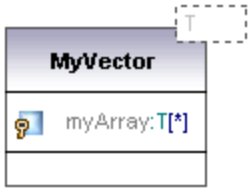
Anmerkung: Für zukünftige 3-Weg-Zusammenführungen zwischen den Projektdateien in den Ordnern **Tom** und **Alice** sollte nun in Zukunft die Projektdatei im Ordner Alice als gemeinsame Vorgängerdatei verwendet werden.

6.8 UML-Vorlagen

UModel unterstützt die Verwendung von UML-Vorlagen (Templates) und das Mappen dieser Vorlagen auf oder von Java-, C#- und Visual Basic Generics.

- Vorlagen (Templates) sind "potentielle" Modellelemente mit nicht gebundenen formalen Parametern.
- Diese parametrisierten Modellelemente beschreiben eine Gruppe von Modellelementen eines bestimmten Typs: Classifier oder Operationen.
- Vorlagen können nicht direkt als Typen verwendet werden. Die Parameter müssen gebunden sein.
- Instantiieren bedeutet, die Vorlagenparameter an aktuelle Werte zu binden.
- Aktuelle Werte für Parameter sind Ausdrücke.
- Durch die Bindung zwischen einer Vorlage und einem Modellelement wird ein neues Modellelement (ein gebundenes Element) auf Basis der Vorlage erzeugt.
- Bei Vorhandensein mehrerer einschränkender Classifier in C# können die Vorlagenparameter direkt auf dem Register "Eigenschaften" bearbeitet werden, wenn der Vorlagenparameter ausgewählt ist.

Anzeige der **Vorlagensignatur** in UModel:



- Klassenvorlage mit dem Namen **MyVector** mit dem formalen Vorlagenparameter "**T**", der in einem gestrichelten Rechteck angezeigt wird.
- Formale Parameter ohne Typinfo (T) sind implizit Classifier: Class, Datatype, Enumeration, PrimitiveType, Interface. Alle anderen Parametertypen müssen explizit angezeigt werden z.B. Integer.
- Eigenschaft **myArray** mit einer unbeschränkten Anzahl an Elementen vom Typ T.

  Wenn Sie mit der rechten Maustaste auf die Vorlage klicken und den Eintrag **Anzeigen | Gebundene Elemente** auswählen, werden die gebundenen Elemente angezeigt.

Anzeige der **Vorlagenverwendung**:

- eine gebundene benannte Vorlage **intvector**
- Vorlage vom Typ **MyVector**, wobei
- der Parameter **T** durch **int** ersetzt wird
- **"Ersetzt durch"** wird angezeigt durch **- >**.

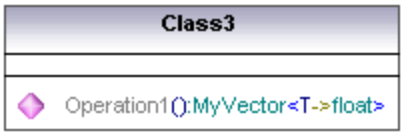
Verwendung von Vorlagen in Eigenschaften/Operationen:

Eine anonyme Vorlagenverwendung:

- Eigenschaft MyFloatVector vom Typ **MyVector<T->float>**

© 2016-2022 Altova GmbH

Vorlagen können auch beim Definieren von Eigenschaften oder Operationen definiert werden. Die Autokomplettierungsfunktion hilft Ihnen, die Syntaxvorgaben einzuhalten.



- Operation1 gibt einen Vektor von floats zurück.

6.8.1 Vorlagensignaturen

Eine Vorlagensignatur ist ein String, der die formalen Vorlagenparameter definiert. Eine Vorlage (Template) ist ein parametrisiertes Element, das zum Generieren neuer Modellelemente verwendet wird, indem die formalen Parameter durch tatsächliche Parameter (Werte) ersetzt werden bzw. daran gebunden werden.

Formaler Vorlagenparameter

T
Vorlage mit einem einzigen formalen Parameter, ohne Typenkonzept
(speichert Elemente vom Typ T)

Multiple formale Vorlagenparameter

KeyType:DateType, ValueType

Parameterersetzung

T>aBaseClass

Die Parametersubstitution muss vom Typ "aBaseClass" oder davon abgeleitet sein.

Standardwerte für Vorlagenparameter

T=aDefaultValue

Ersetzen von Classifiern

T>{contract}aBaseClass

allowsSubstitutable is true
Der Parameter muss ein Classifier sein, der an die Stelle des Classifiers gesetzt werden kann, der durch den Classifier-Namen definiert ist.

Einschränken von Vorlagenparametern

T:Interface>anInterface

Wenn Sie Parameter auf etwas anderes als eine Klasse einschränken wollen (Schnittstelle, Datentyp), wird die Einschränkung nach dem ":"-Zeichen angezeigt. So wird T z.B. an eine Schnittstelle gebunden (T:Interface) die vom Typ "anInterface" (>anInterface) sein muss.

Verwendung von Platzhalterzeichen in Vorlagensignaturen

T>vector<T->?<aBaseClass>

Der Vorlagenparameter T muss vom Typ "vector" sein, der Objekte enthält, die ein übergeordneter Typ von aBaseClass sind.

Erweitern von Vorlagenparametern

T>Comparable<T->T>

6.8.2 Vorlagenverwendung

Bei der Vorlagenverwendung werden die formalen Parameter durch tatsächliche Werte ersetzt, d.h. die Vorlage wird instantiiert. UModel generiert in diesem Fall automatisch anonym gebundene Klassen. Vorlagenverwendungen können wie unten gezeigt im Klassennamen-Feld definiert werden.

Ersetzen/Binden von formalen Parametern

Vektor <T->int>

Erstellen von Vorlagenverwendungen über den Klassennamen

a_float_vector:vector<T->float>

Binden von mehreren Vorlagen gleichzeitig

Class5:vector<T->int, map<KeyType->int, ValueType<T->int>

Verwendung von Platzhalterzeichen ? als Parameter (Java 5.0)

vector<T->?>

Einschränken von Platzhalterzeichen - upper bounds (UModel Erweiterung)

vector<T->?>aBaseClass>

Einschränken von Platzhalterzeichen - lower bounds (UModel Erweiterung)

vector<T->?<aDerivedClass>

6.8.3 Vorlagenverwendung in Operationen und Eigenschaften

Operation, die eine gebundene Vorlage zurückgibt

Class1
Operation1():vector<T->int>

Der Parameter T ist an "int" gebunden. Operation1 gibt einen Vektor von ints zurück.

Klasse, die eine Vorlagenoperation enthält

Class1
Operation1<T>(in T):T

Verwenden von Platzhalterzeichen

Class1
Property1:vector<T->?>

 © 2016-2022 Altova GmbH

Diese Klasse enthält einen generischen Vektor eines nicht spezifizierten Typs (? ist das Platzhalterzeichen).

Typisierte Eigenschaften können folgendermaßen als Assoziationen angezeigt werden:

- Rechtsklicken Sie auf eine Eigenschaft und wählen Sie **Anzeigen | PropertyX als Assoziation** oder
- Ziehen Sie eine Eigenschaft auf den Diagrammhintergrund.

7 Generieren von UML-Dokumentation

Altova Website: UML-Projektdokumentation

Mit dem Befehl **Projekt | Dokumentation generieren** wird detaillierte Dokumentation zu Ihrem UML-Projekt in den Formaten HTML, Microsoft Word, RTF oder PDF generiert. Mit diesem Befehl generierte Dokumentation kann beliebig geändert und verwendet werden; Sie benötigen dazu keine Genehmigung von Altova.

Anmerkungen

- Um Dokumentation im PDF-Format zu generieren oder die generierte Dokumentation anzupassen, muss Altova StyleVision (https://www.altova.com/de/stylevision) auf Ihrem Rechner installiert und lizenziert sein.
- Um Dokumentation im Format MS Word zu generieren, muss MS Word (Version 2000 oder höher) installiert sein.

Die Dokumentation wird für die Modellierungselemente generiert, die Sie im Dialogfeld "Dokumentation generieren" ausgewählt haben. Sie können entweder das vordefinierte Design oder ein benutzerdefiniertes StyleVision Power Stylesheet (SPS) für das Design verwenden. Bei Verwendung eines StyleVision Power Stylesheet können Sie die Ausgabe der generierten Dokumentation anpassen, siehe Anpassen der Ausgabe mit StyleVision 299.

Sie können auch einzelne Teile eines Projekts dokumentieren. Rechtsklicken Sie dazu in der Modellstruktur auf ein Element (bzw. mit Strg + Klick auf mehrere Elemente) und wählen Sie den Befehl **Dokumentation generieren**. Beim Element kann es sich um einen Ordner, eine Klasse, eine Schnittstelle, usw. handeln. Die Dokumentationsoptionen sind in beiden Fällen dieselben.

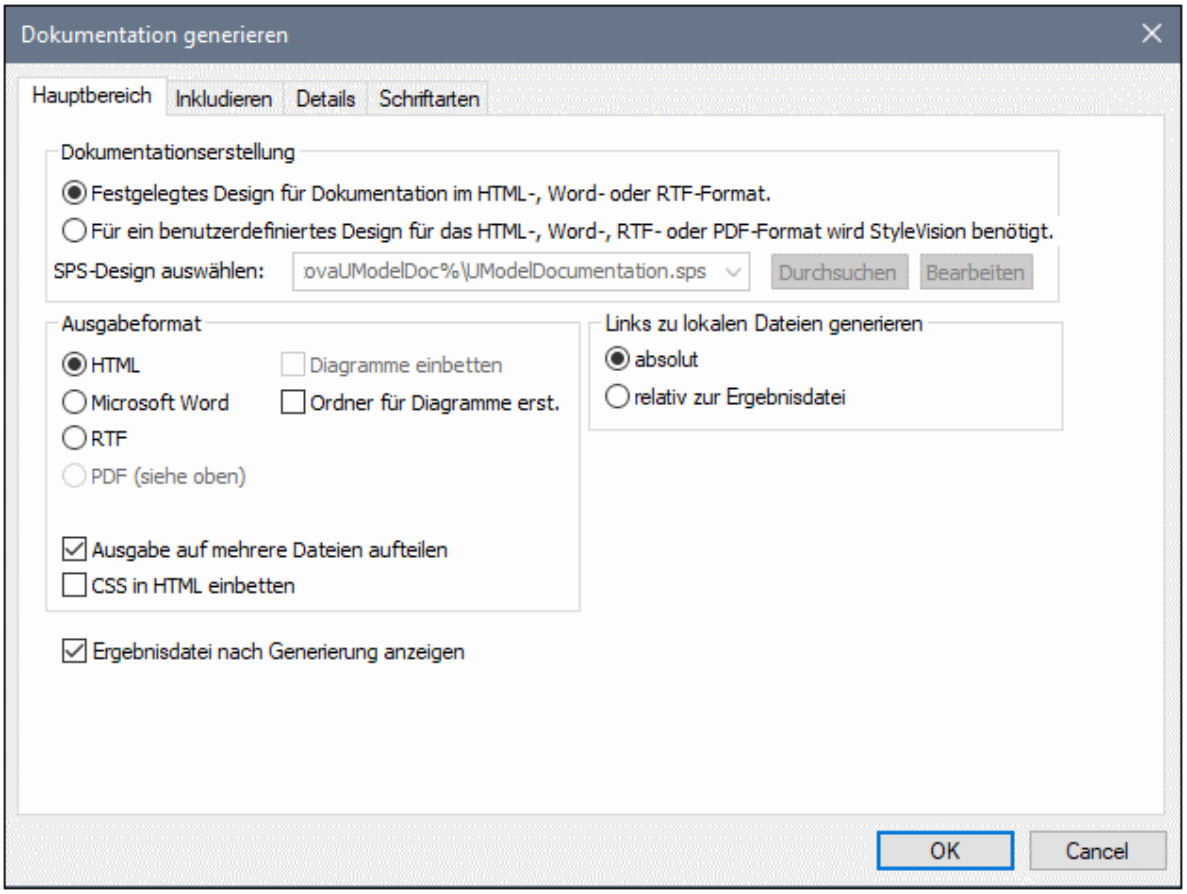
Miteinander in Beziehung stehende Elemente werden in der generierten Ausgabe durch Hyperlinks miteinander verbunden, sodass Sie von Komponente zu Komponente navigieren können. In der Dokumentation sind auch alle manuell erstellten Hyperlinks enthalten.

Wenn Ihr Projekt UModel-Profile (wie C#, Java, VB.NET, usw.) enthält, werden diese in die generierte Dokumentation inkludiert, wenn die Option **Inkludierte Unterprojekte** auf dem Register **Inkludieren** ausgewählt ist, siehe Optionen zur Generierung von Dokumentation 294.

So generieren Sie Dokumentation:

1. Öffnen Sie ein Projekt (z.B. **C:\Benutzer\<Benutzername>\Dokumente\Altova\UModel2023\UModelExamples\Bank_Java.ump**).
2. Klicken Sie im Menü **Projekt** auf **Dokumentation generieren.**

 © 2016-2022 Altova GmbH



3. Wählen Sie ein Ausgabeformat aus (HTML, Word, RTF, PDF).
4. Passen Sie optional die Generierungsoptionen an, siehe Optionen zur Generierung von Dokumentation 294.
5. Klicken Sie auf **OK** und wählen Sie einen Zielordner für die Ausgabe.

In den folgenden Abbildungen sehen Sie einen Ausschnitt aus einer UModel-Dokumentation mit vordefiniertem Design, die anhand der Projektdatei **Bank_Java.ump** generiert wurde.

Bank_Java.ump

Projektpfad **C:\Documents\Altova\UModel2019\UModelExamples\Bank_Java.ump**

Diagrammindex:

Aktivitätsdiagramm	collectData Draft	
Klassendiagramm	BankView Main	Hierarchy of Account
Komponentendiagramm	BankView realization	Overview
Kompositionsstrukturdiagramm	Account Transfer	
Deployment-Diagramm	Deployment	
Objektdiagramm	Sample Accounts	
Profildiagramm	Apply Java Profile	
Sequenzdiagramm	Collect Account Information	Connect to BankAPI
Zustandsdiagramm	BankAPI Draft	Query BankServer Draft
UseCase-Diagramm	Overview Account Balance	

Elementindex:

Akteur	Bank	Standard User	
Klasse	Account CreditCardAccount	Bank SavingsAccount	BankView
Komponente	Bank API client	BankView	BankView GUI
Schnittstelle	IBankAPI		

Wie oben gezeigt, enthält die generierte Dokumentation im oberen Bereich der HTML-Datei einen Diagramm- und Elementindex (mit Links).

In der Abbildung unten sehen Sie einen Ausschnitt aus der generierten Dokumentation für die Klasse `Account`. Beachten Sie, dass die einzelnen Mitglieder in Klassendiagrammen auch mit Hyperlinks zu ihren Definitionen versehen sind. Wenn Sie z.B. auf eine Eigenschaft oder Operation klicken, gelangen Sie zu ihrer Definition. Auch die Hierarchieklassen sowie unterstrichener Text sind mit Hyperlinks versehen.

© 2016-2022 Altova GmbH

Klasse **Account**

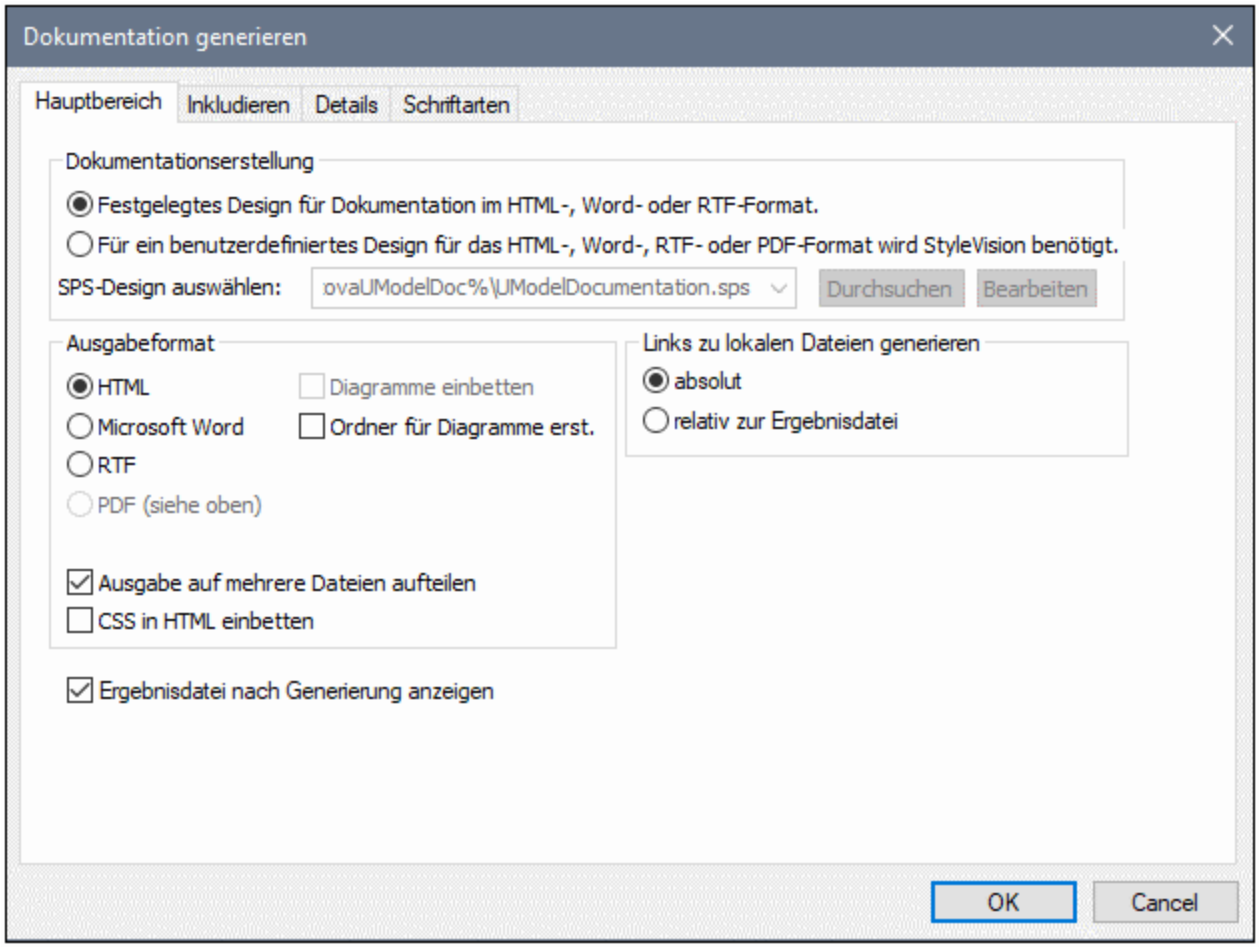
Diagramm	*Account* balance:float=0 id:String «constructor» Account() getBalance():float getId():String collectAccountInfo(in bankAPI:IBankAPI):boolean
Hierarchie	*Account* CheckingAccount SavingsAccount CreditCardAccount
Owner	**bankview**
Eigenschaften	Qualifizierter Name Design View::BankView::com::altova::bankview::Account Sichtbarkeit public leaf false abstrakt true isFinalSpecialization false aktiv false Codedateiname Account.java Codedateipfad C:\UML_Bank_Sample\JavaCode\com\altova\bankview\Account.java «annotations» false «static» false «strictfp» false

7.1 Optionen zur Generierung von Dokumentation

Vor der Generierung von Dokumentation anhand von UModel-Projekten können Sie verschiedene Option, wie unten beschrieben, definieren. Die Optionen sind nach dem Register, auf dem sie im Dialogfeld "Dokumentation generieren" angezeigt werden, angeordnet.

Register "Hauptbereich"

Das Register **Hauptbereich** enthält die allgemeinen Dokumentationsgenerierungsoptionen.



Dokumentationserstellung:

- Wählen Sie die Option **Festgelegtes Design ...verwenden,** um das vordefinierte UModel-Dokumentationdesign zu verwenden.
- Wählen Sie die Option **Für ein benutzerdefiniertes Design..,**. um Dokumentation zu generieren, die mit Hilfe eines in StyleVision erstellten StyleVision Power Stylesheet (.sps-Datei) formatiert wird. Anmerkung: Für diese Option muss Altova StyleVision installiert sein, siehe Anpassen der Ausgabe mit StyleVision 299.
- Klicken Sie auf **Durchsuchen,** um zu einer vordefinierten Stylesheet-Datei zu navigieren.

 © 2016-2022 Altova GmbH

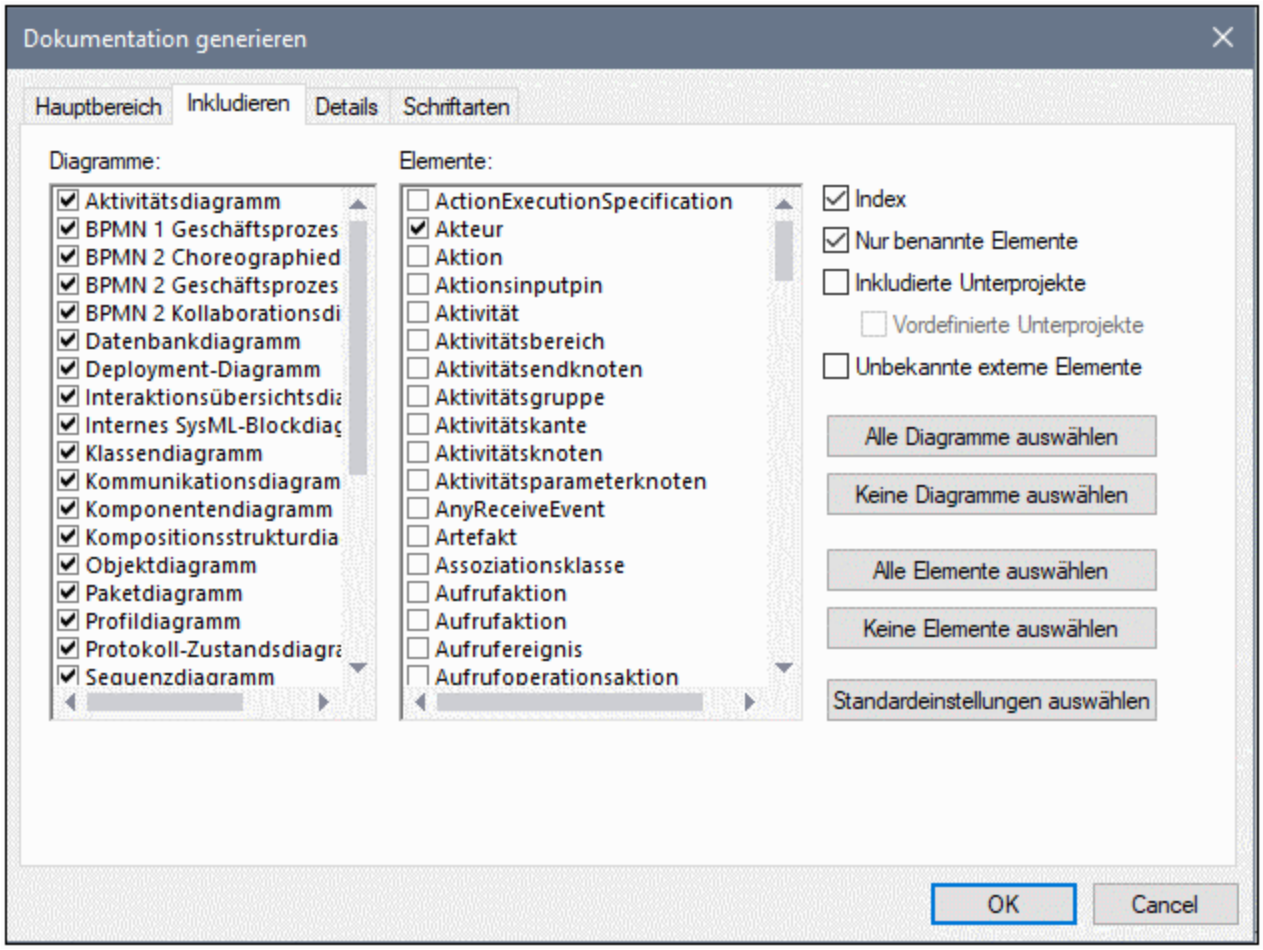
- Klicken Sie auf **Bearbeiten**, um StyleVision zu starten und die ausgewählte Stylesheet-Datei in einem StyleVision-Fenster zu öffnen.

Ausgabeformat:

- Als Ausgabeformat kann eines der folgenden definiert werden: HTML, Microsoft Word, RTF oder PDF. Microsoft Word-Dokumente werden mit der Dateierweiterung .doc angelegt, wenn sie anhand eines festgelegten Designs erstellt werden, und mit der Dateierweiterung .docx, wenn sie anhand eines StyleVision Power Stylesheet generiert werden. Für das Ausgabeformat PDF muss Altova StyleVision installiert sein.
- Mit der Option **Ausgabe auf mehrere Dateien aufteilen** wird für jedes Modellierungselement (Klasse, Schnittstelle, Diagramm, usw.) eine Ausgabedatei erstellt. Deaktivieren Sie dieses Kontrollkästchen, um eine einzige globale Datei mit allen Modellierungselementen zu generieren.
- Mit der Option **CSS in HTML einbetten** können Sie den generierten CSS-Code in die HTML-Dokumentation einbetten. Deaktivieren Sie dieses Kontrollkästchen, damit die CSS-Datei extern bleibt.
- Die Option **Diagramme einbetten** ist für die Ausgabeoptionen Microsoft Word und RTF aktiviert. Wenn dieses Kontrollkästchen aktiviert ist, werden Diagramm in die generierte Datei eingebettet. Diagramme werden als PNG-Dateien erstellt und über Objektlinks in der Ergebnisdatei angezeigt.
- Mit der Option **Ordner für Diagramme erstellen** wird unterhalb des ausgewählten Ausgabeordners ein Unterordner erstellt, der alle Diagramme enthält.
- Die Option **Ergebnisdatei nach Generierung anzeigen** ist für alle Ausgabeformate aktiviert. Wenn dieses Kontrollkästchen aktiviert ist, wird die generierte Hauptdatei (bei HTML-Dateien) im Standard-Browser, (bei Word-Dateien) in Microsoft Word oder (für .pdf- oder .rtf-Dateien) in der Standardapplikation für angezeigt.
- Über die Option **Links zu lokalen Dateien generieren** können Sie festlegen, ob die generierten Links absolute oder relative Links zur Ausgabedatei sein sollen.

Register "Inkludieren"

Auf dem Register **Inkludieren** können Sie auswählen, welche Diagramme und Modellierungselemente in der Dokumentation aufscheinen sollen.



Um zu verhindern, dass Unterprojekte oder Profile dokumentiert werden, deaktivieren Sie das Kontrollkästchen **Inkludierte Unterprojekte**. Bedenken Sie, dass Elemente oder Diagramme in Unterprojekten nicht in die generierte Dokumentation inkludiert werden, wenn dieses Kontrollkästchen deaktiviert ist. Aktivieren Sie das Kontrollkästchen **Vordefinierte Unterprojekte,** um vordefinierte UModel-Profile wie C#- oder Java-Profile zu inkludieren. Beachten Sie jedoch, dass das Generieren von Dokumentation anhand von vordefinierten Projekten sehr lange dauert. **Unbekannte externe Elemente** bezieht sich auf Elemente, deren Art nicht identifiziert werden konnte. Dies kommt normalerweise dann vor, wenn Sie Quellcode in UModel importieren, ohne zuerst die vordefinierten Unterprojekte für diese Sprache oder Sprachversion zu inkludieren, siehe Inkludieren von Unterprojekten 164.

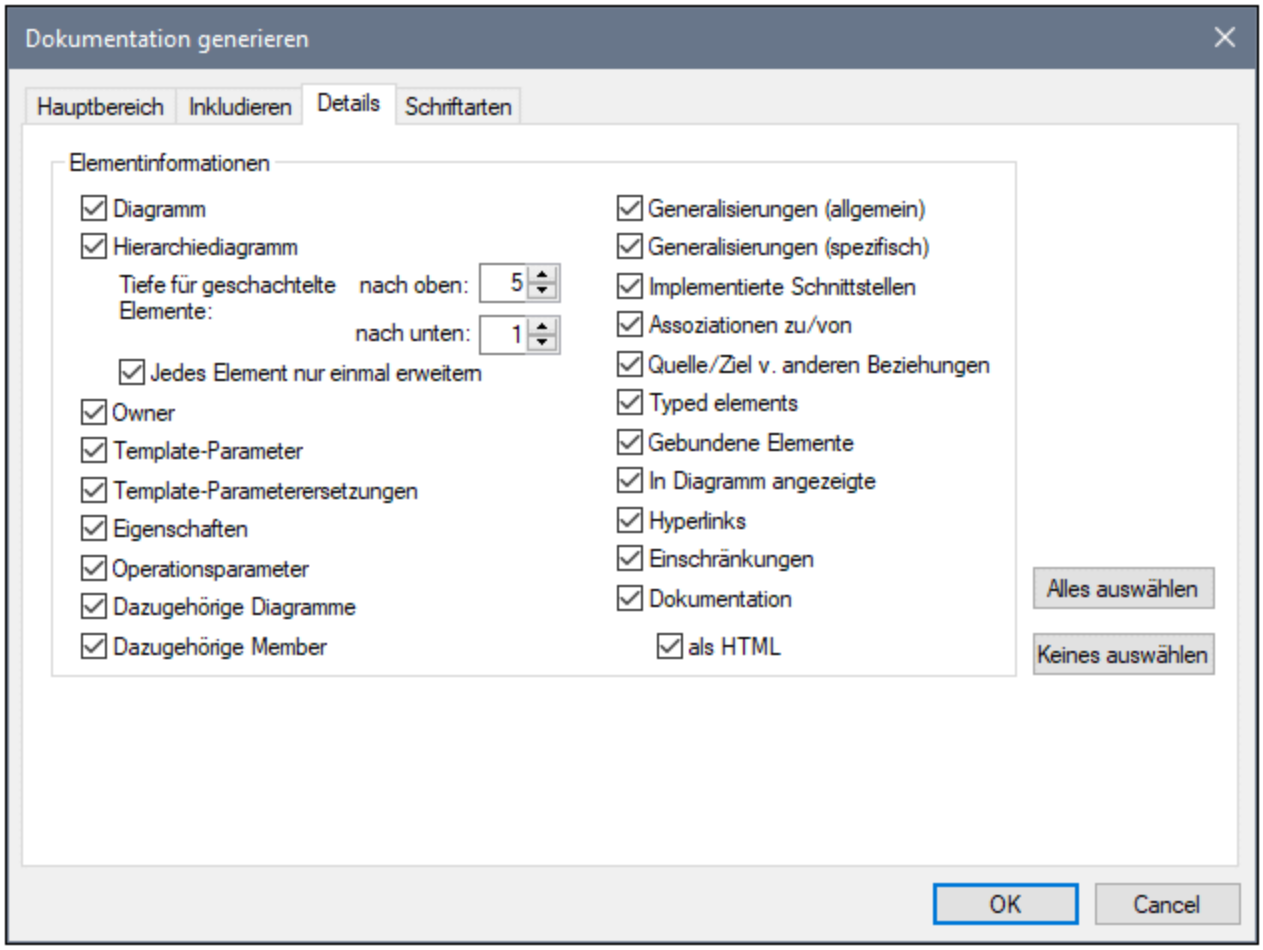
Register "Details"

Auf dem Register **Details** können Sie die Elementdetails auswählen, die in der Dokumentation aufscheinen sollen.

- Wenn Sie den Text von **XML-Tags** in Ihre Dokumentation importieren möchten, deaktivieren Sie unter der Option **Dokumentation** die Option **als HTML**.
- In den Feldern **Nach oben** / **Nach unten** können Sie die Verschachtelungstiefe festlegen, die im Hierarchiediagramm oberhalb / unterhalb der aktuellen Klasse angezeigt werden soll.

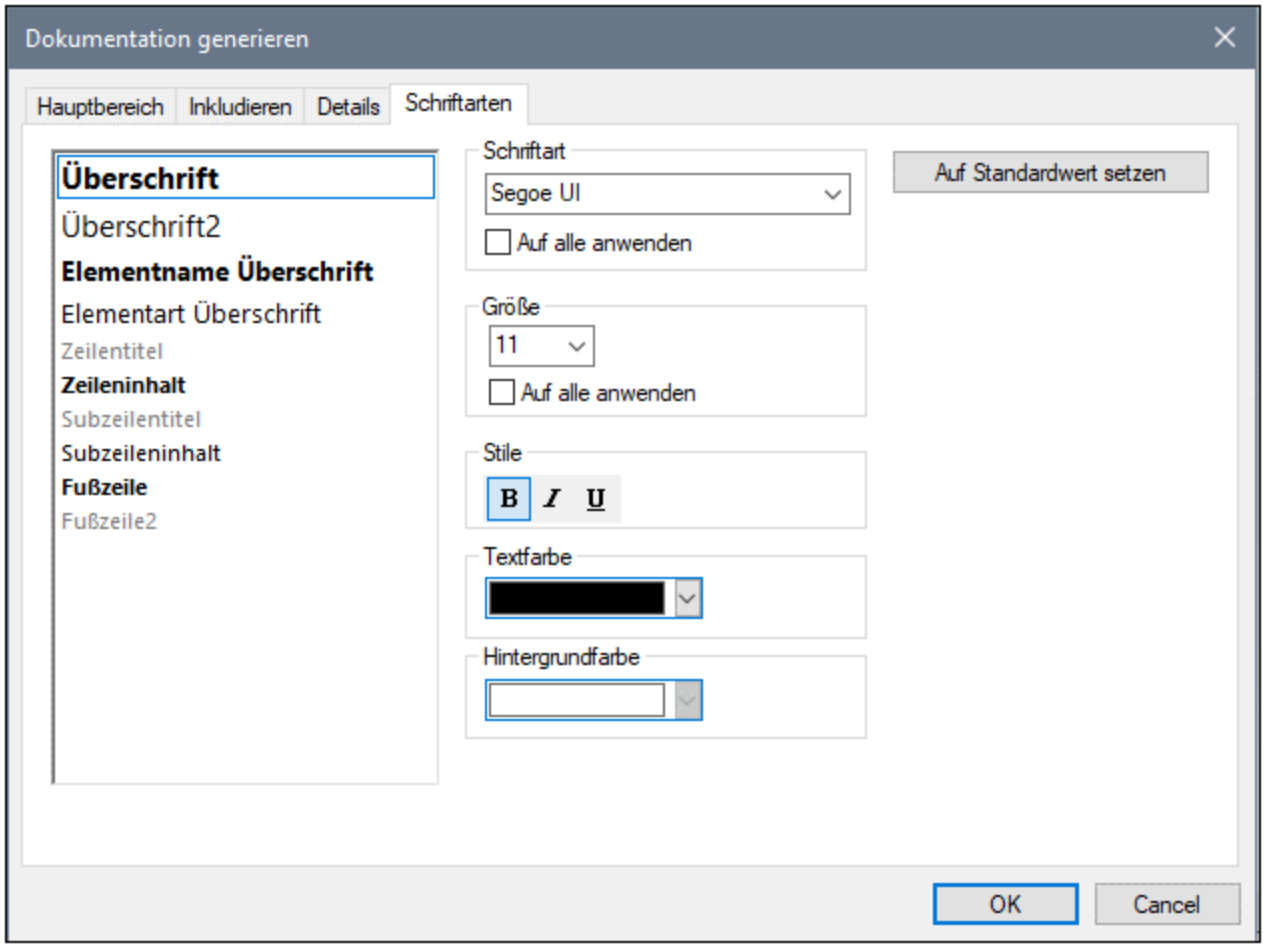
 © 2016-2022 Altova GmbH

- Mit der Option **Jedes Element nur einmal erweitern** wird im selben Bild oder Diagramm nur immer jeweils ein Classifier erweitert.



Register "Schriftarten"

Auf dem Register "Schriftarten" können Sie die Schriftarteinstellungen für die verschiedenen Überschriften und Textinhalte anpassen.



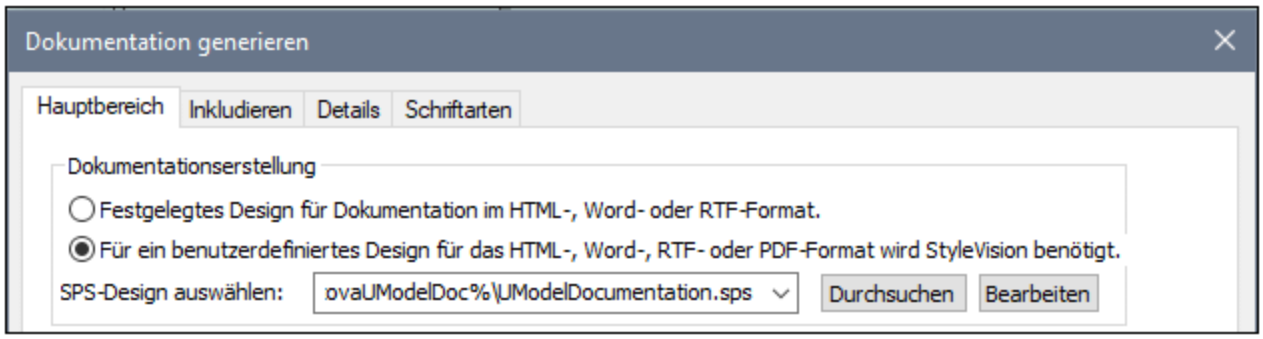
© 2016-2022 Altova GmbH

7.2 Anpassen der Ausgabe mit StyleVision

Sie können das Design von mit UModel generierter Dokumentation mit Hilfe von StyleVision Power Stylesheet (.sps)-Dateien anpassen. Solche Dateien werden in Altova StyleVision (https://www.altova.com/de/stylevision) generiert. Der Vorteil bei der Generierung der Dokumentation anhand einer .sps-Datei ist, dass Sie völlig freie Hand bei der Gestaltung der Dokumentation haben. Außerdem kann die Dokumentation bei Verwendung einer .sps-Datei auch im PDF-Format generiert werden.

> Damit Sie mit Hilfe von .sps-Dateien Dokumentation generieren können, muss Altova StyleVision auf Ihrem Rechner installiert und lizenziert sein.

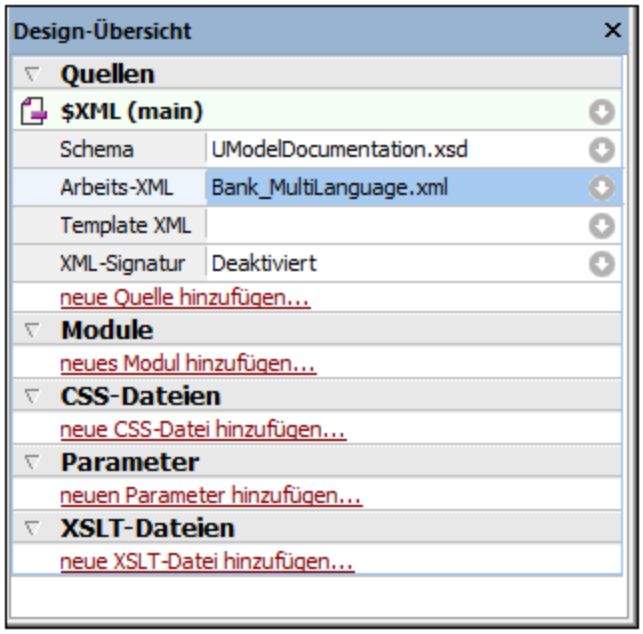
Im Lieferumfang von UModel ist eine vordefinierte .sps-Datei inkludiert, die unter folgendem Pfad zur Verfügung steht: **C:\Benutzer\<Benutzername\Dokumente\UModel2023\Documentation\UModel\UModelDocumentation.sps**. Um die generierte Dokumentation mit Hilfe einer benutzerdefinierten .sps-Datei zu formatieren, wählen Sie bei der Generierung von Dokumentation diese Option aus, z.B.:



Als Ausgangsbasis für Ihr benutzerdefiniertes Stylesheet erstellen Sie eine Kopie der Standard-.sps-Datei **UModelDocumentation.sps** und bearbeiten Sie diese in StyleVision. Sie können z.,B. das vorhandene Format ändern oder Links und Bilder zum Design hinzufügen.

Jedem StyleVision Power Stylesheet liegt ein XML-Schema zugrunde. Im Fall von Stylesheets zur Definition des Designs von UModel-generierter Dokumentation befindet sich das Schema unter dem folgenden Pfad: **C:\Benutzer\<Benutzername\Dokumente\UModel2023\Documentation\UModel\UModelDocumentation.xsd**. Beachten Sie, dass die Datei **ModelDocumentation.xsd** die Datei **Documentation.xsd** im übergeordneten Ordner referenziert.

Wenn Sie benutzerdefinierte .sps-Dateien für die UModel-Dokumentation in StyleVision erstellen, muss als Schema die Datei **UModelDocumentation.xsd** verwendet werden. In der Abbildung unten sehen Sie das Fenster "Design-Übersicht" von StyleVision, nachdem Sie die Datei **UModelDocumentation.sps** geöffnet haben. Beachten Sie, dass darin die Schema-Datei **UModelDocumentation.xsd** sowie eine XML-Arbeitsdatei für die Design-Vorschau verwendet werden. Die XML-Arbeitsdatei steht relativ zur Schema-Datei im Unterordner **SampleData** zur Verfügung.



Eine Anleitung, wie Sie .sps-Dateien bearbeiten, finden Sie in der Dokumentation zu StyleVision (https://www.altova.com/de/documentation).

© 2016-2022 Altova GmbH

8 UML-Diagramme

Altova Website: UML-Diagramme

UML-Diagramme werden in zwei große Gruppen eingeteilt: Strukturdiagramme, in denen eine statische Ansicht des Modells zu sehen ist, und Verhaltensdiagramme, in denen die dynamische Ansicht dargestellt wird. UModel unterstützt alle vierzehn Diagramme der UML 2.5-Spezifikation sowie weitere zusätzliche Diagramme.

Zu den Verhaltensdiagrammen[302] gehören Aktivitäts-, Zustands-, Protokoll-Zustands- und Use Case-Diagramme sowie Interaktionsdiagramme Kommunikations-, Interaktionsübersichtsdiagramme, Sequenzdiagramme und Zeitverlaufsdiagramme.

Zu den Strukturdiagrammen[394] gehören: Klassen-, Kompositionsstruktur-, Komponenten-, Deployment-, Objekt- und Paketdiagramme.

Zusätzliche Diagramme[435]: XML-Schema-Diagramme.

Anmerkung:
Mit **Strg+Eingabetaste** können Sie mehrzeilige Beschriftungen für die meisten Modellierungsdiagramme erstellt werden, z.B. Lebenslinienbeschriftungen in Sequenzdiagrammen, Zeitverlaufsdiagrammen; Guard-Bedingungen, Zustandsnamen, Aktivitätsnamen usw.

8.1 Verhaltensdiagramme

In diesen Diagrammen werden Verhaltensaspekte eines Systems oder Geschäftsvorgangs beschrieben. Sie enthalten eine Untergruppe von Diagrammen zur Darstellung der Wechselbeziehungen zwischen Objekten.

Verhaltensdiagramme

Aktivitätsdiagramm
Zustandsdiagramm
Protokoll-Zustandsdiagramm
Use Case-Diagramm 347

Eine Untergruppe der Verhaltensdiagramme bilden jene zur Darstellung von Wechselwirkungen zwischen Objekten:

Kommunikationsdiagramm
Interaktionsübersichtsdiagramm
Sequenzdiagramm
Zeitverlaufsdiagramm 385

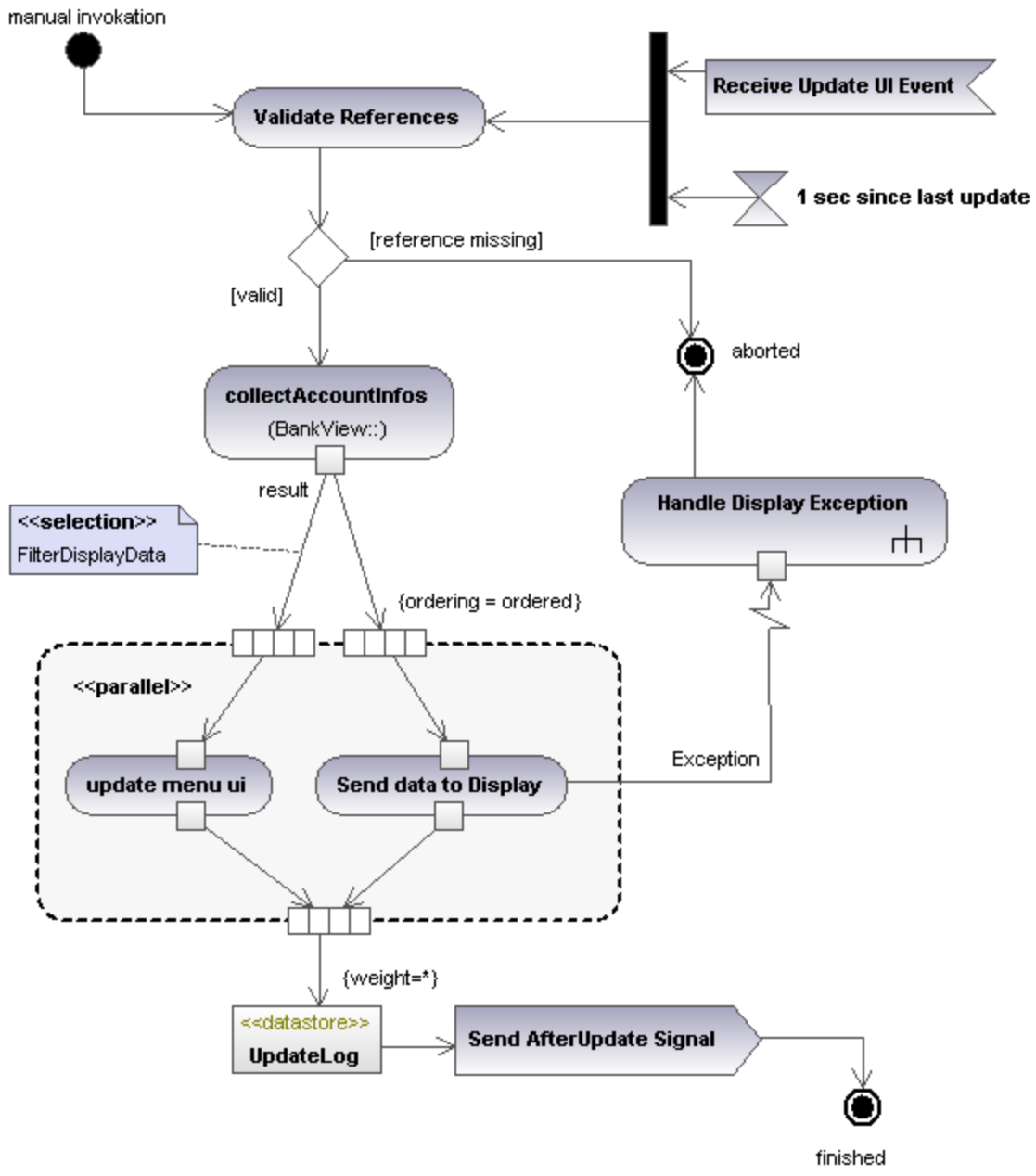
8.1.1 Aktivitätsdiagramm

Altova Website: UML-Aktivitätsdiagramme

Aktivitätsdiagramme eignen sich, um reale Arbeitsabläufe von Geschäftsprozessen zu modellieren und um anzuzeigen, welche Aktionen dabei erforderlich sind und welche Abhängigkeiten diese haben. Im Aktivitätsdiagramm wird die Reihenfolge von Aktivitäten beschrieben. Es wird sowohl bedingte als auch parallele Verarbeitung unterstützt. Das Aktivitätsdiagramm ist eine Variante des Zustandsdiagramms, wobei die Zustände hier Aktivitäten sind.

Das unten gezeigte Aktivitätsdiagramm steht im UModel-Ordner **...\UModelExamples** im Beispiel **Bank_MultiLanguage**.ump zur Verfügung.

 © 2016-2022 Altova GmbH



8.1.1.1 Einfügen von Aktivitätsdiagrammelementen

So fügen Sie Elemente zum Diagramm hinzu:

1. Klicken Sie in der Symbolleiste "Aktivitätsdiagramm" auf die Symbolleisten-Schaltfläche des Elements.

2. Klicken Sie in das Aktivitätsdiagramm, um das Element einzufügen.

Um mehrere Elemente des ausgewählten Typs einzufügen, halten Sie die **Strg-Taste** gedrückt und klicken Sie in das Diagrammfenster.

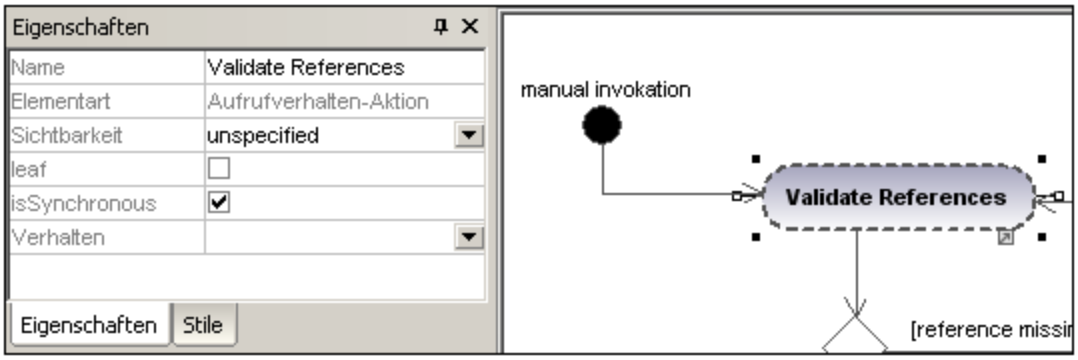
Ziehen bestehender Elemente in das Aktivitätsdiagramm

Die meisten Elemente, die in anderen Aktivitätsdiagrammen vorkommen, können in ein bestehendes Aktivitätsdiagramm eingefügt werden.

1. Suchen Sie das gewünschte Element im Fenster "Modell-Struktur" 80 (Sie können dazu das Suchfunktionstextfeld verwenden oder **Strg + F** drücken).
2. Ziehen Sie das/die Element(e) in das Aktivitätsdiagramm.

Einfügen einer Aktion (Aufrufverhalten)

1. Klicken Sie auf die Symbolleisten-Schaltfläche **Aktion (Aufrufverhalten)** und anschließend in das Aktivitätsdiagramm, um es einzufügen.
2. Geben Sie den Namen der Aktion ein, z.B. "Validate References" (=Referenzen validieren) und drücken Sie zur Bestätigung die **Eingabetaste**.

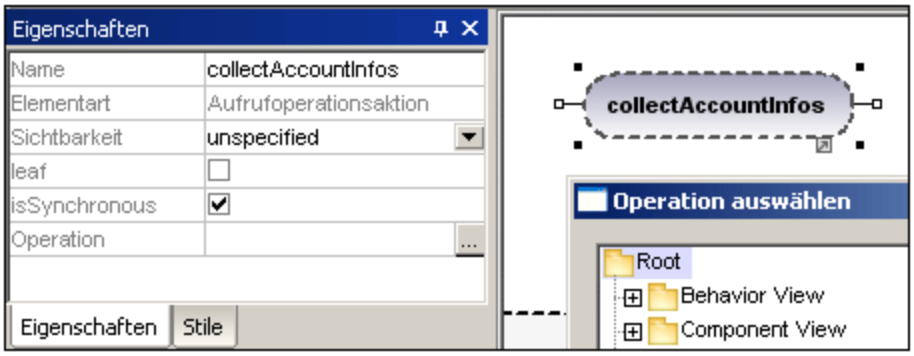


Anmerkung: Durch Drücken von **Strg+Eingabetaste** können Sie einen mehrzeiligen Namen erstellen.

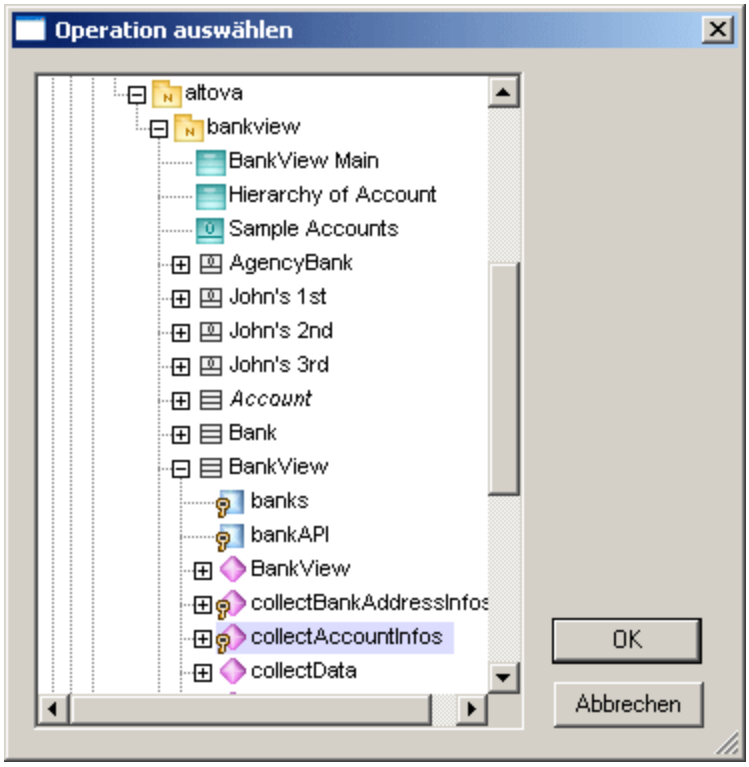
Einfügen einer Aktion (Aufrufoperation) und Auswählen einer bestimmten Operation

1. Klicken Sie in der Symbolleiste auf das Symbol **Aktion (Aufrufoperation)** und anschließend in das Aktivitätsdiagramm um es einzufügen.
2. Geben Sie den Namen der Aktion ein, z.B. collectAccountInfo und drücken Sie zur Bestätigung die **Eingabetaste**.
3. Klicken Sie auf dem Register **Eigenschaften** auf die **Durchsuchen**-Schaltfläche rechts vom Feld "Operation". Daraufhin wird das Dialogfeld "Operation auswählen" geöffnet. Hier können Sie die gewünschte Operation auswählen.

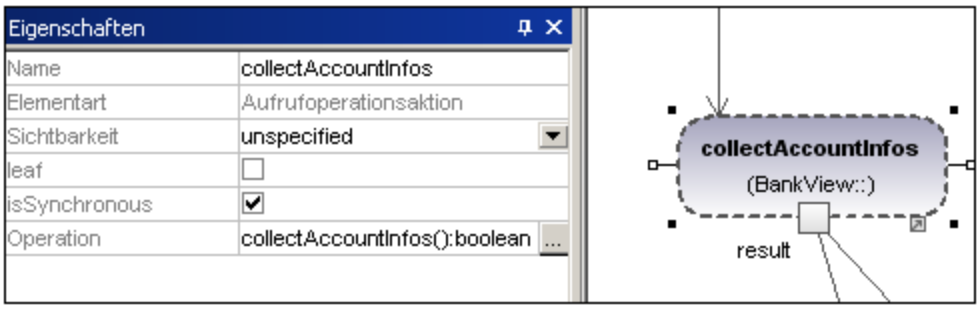
 © 2016-2022 Altova GmbH



4. Navigieren Sie zur gewünschten Operation und bestätigen Sie die Auswahl mit **OK**.



In diesem Beispiel befindet sich die Operation "collectAccountInfos" in der Klasse `BankView`.



8.1.1.2 Erstellen von Verzweigungen und Merges

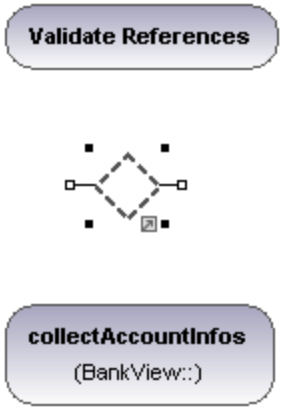
Erstellen einer Verzweigung (alternativer Fluss)
Eine Verzweigung hat einen einzigen eingehenden Fluss und mehrere mit "Guards" versehene ausgehende Flüsse. Es kann nur einer der ausgehenden Flüsse ausgewählt werden, daher sollten die Guards einander gegenseitig ausschließen.

In diesem Beispiel müssen die (BankView) Referenzen validiert werden.

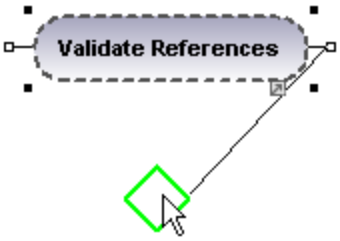
- branch1 (Verzweigung1) hat den Guard "reference missing" (=Referenz fehlt). Diese Transition endet im Abbruch der Aktivität.
- branch2 hat den Guard "valid" (= gültig). Diese Transition führt zur Aktivität collectAccountInfos (Kontoinformationen abrufen).

Erstellen einer Verzweigung (alternate flow)

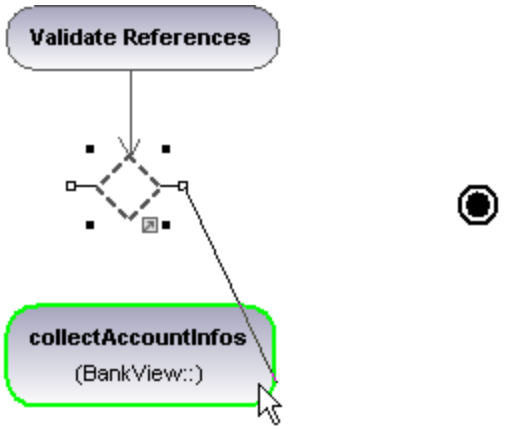
1. Klicken Sie in der Symbolleiste auf das Symbol **Verzweigungsknoten** und fügen Sie das Element im Aktivitätsdiagramm ein.



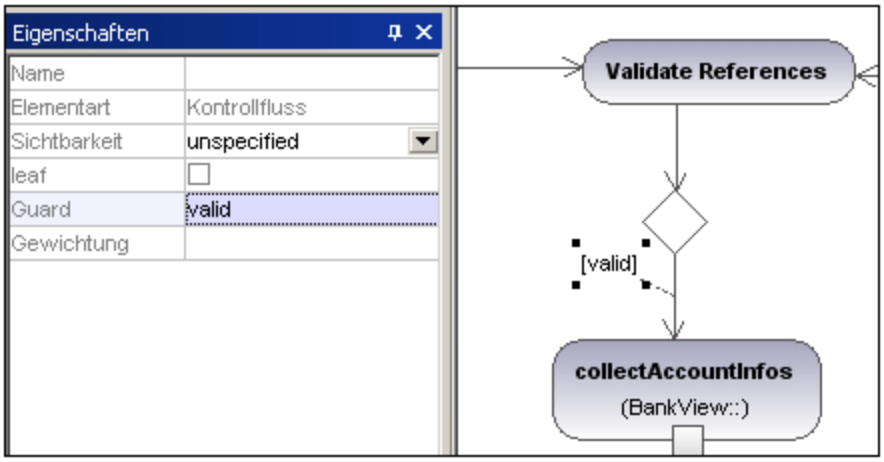
2. Klicken Sie auf das Symbol "Aktivitätsendknoten" , welches für die Aktivität "Abbruch" steht und fügen Sie es in das Aktivitätsdiagramm ein.
3. Klicken Sie auf die Aktivität "Validate References", um sie auszuwählen, anschließend auf den rechten Ziehpunkt **Kontrollfluss** und ziehen Sie den Konnektor, der angezeigt wird, auf das Verzweigungsknoten-Element.



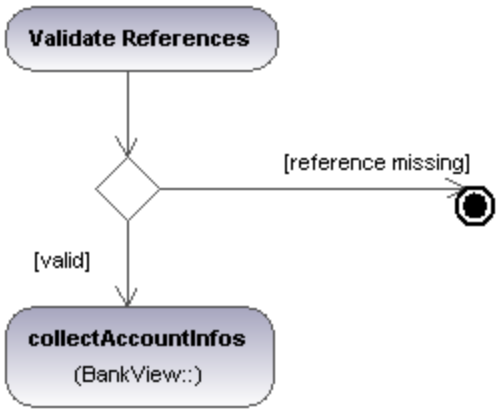
Sobald das Element markiert angezeigt wird, können Sie die Maustaste loslassen.
4. Klicken Sie auf das Verzweigungsknoten-Element, klicken Sie auf den rechten Konnektor **Kontrollfluss** und ziehen Sie ihn auf die Aktion "collectAccountInfos". Nähere Informationen dazu finden Sie unter "Einfügen einer Aktion (Aufrufoperation) 304".



5. Geben Sie auf dem Register "Eigenschaften" die Guard-Bedingung "valid" ein.



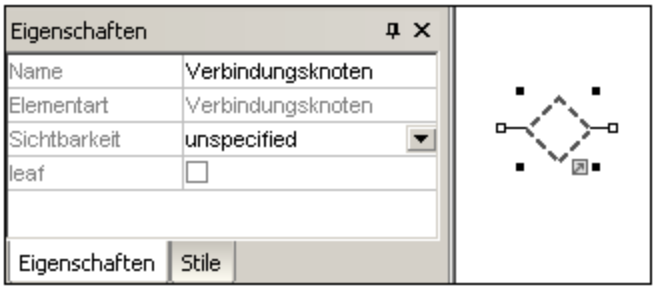
6. Klicken Sie auf das **Verzweigungsknotenelement** und ziehen Sie den rechten Ziehpunkt **Kontrollfluss** auf das Element "Aktivitätsendknoten".
Die Guard-Bedingung zu dieser Transition wird automatisch als "else" definiert. Doppelklicken Sie im Diagramm auf die Guard-Bedingung und ändern Sie diese z.B. in "reference missing".



Bitte beachten Sie, dass UModel die Anzahl der Kontroll- / Objektflüsse in einem Diagramm nicht validiert oder überprüft.

Erstellen eines Merge

1. Klicken Sie in der Symbolleiste auf das Verbindungsknotensymbol und anschließend in das Aktivitätsdiagramm, um es einzufügen.

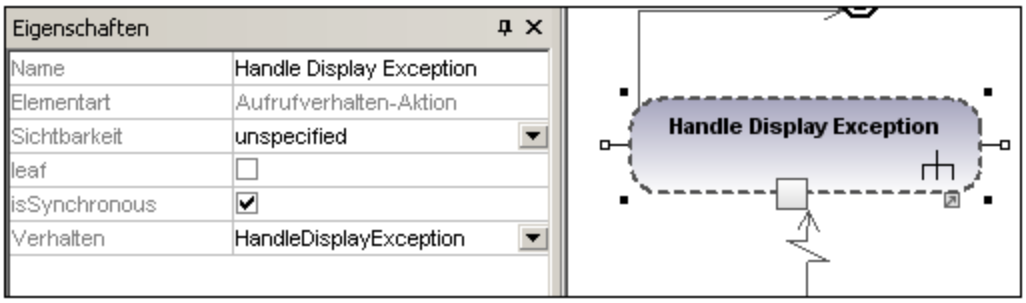


2. Klicken Sie auf die Kontrollfluss (Objektfluss)-Ziehpunkte der zu vereinigenden Aktionen und ziehen Sie die Pfeile auf das Verbindungsknotensymbol.

8.1.1.3 Aktivitätsdiagramm-Elemente

Aktion (Aufrufverhalten)

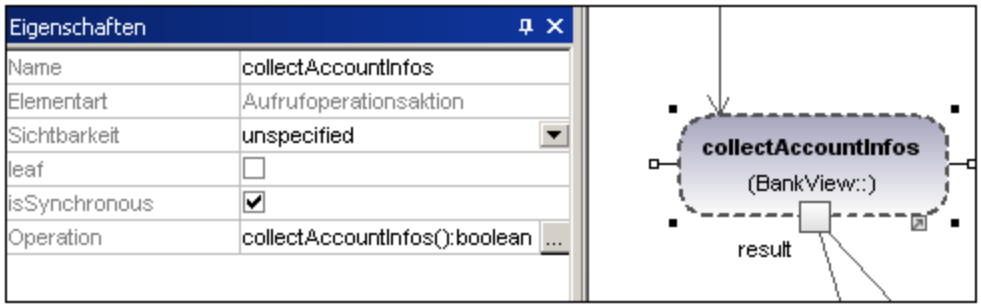
Fügt das Aktionselement **Aufrufverhalten-Aktion** ein, welches ein bestimmtes Verhalten direkt aufruft. Wenn Sie über die Auswahlliste **Verhalten** ein vorhandenes Verhalten auswählen, z.B. HandleDisplayException, wird innerhalb des Elements das Symbol eines Rechens angezeigt.



Aktion (Aufrufoperation)

Fügt eine **Aufrufoperationsaktion** ein, welche ein bestimmtes Verhalten als Methode direkt aufruft. Nähere Informationen finden Sie unter "Einfügen einer Aktion (Aufrufoperation) 304".

 © 2016-2022 Altova GmbH



Aktion (OpaqueAction)

Eine Art von Aktion, mit der Implementierungsinformationen definiert werden. Kann als Platzhalter verwendet werden, bis fest steht, welche spezifische Aktion verwendet werden soll.

Aktion (ValueSpecificationAction)

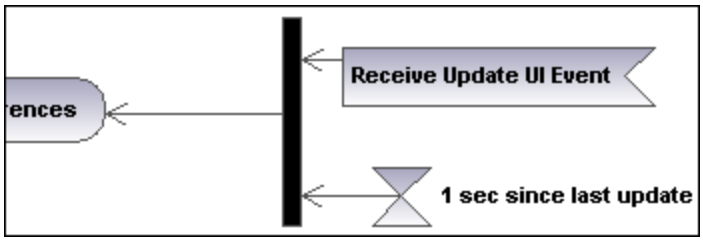
Eine Art von Aktion zum Auswerten (/Generieren)

Ereignisannahmeaktion

Fügt die Ereignisannahmeaktion ein, welche auf das Eintreten eines Ereignisses wartet, das bestimmte Bedingungen erfüllt.

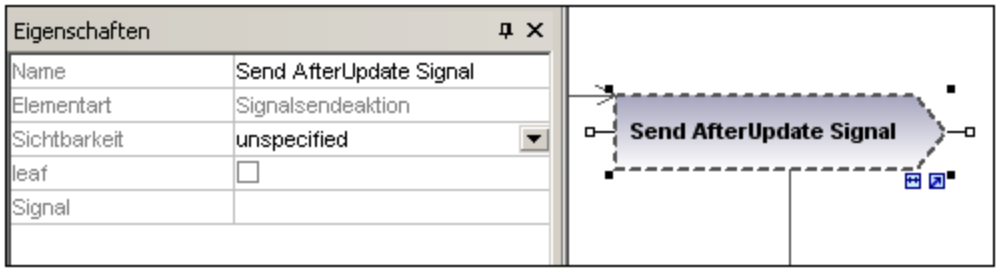
Ereignisannahmeaktion (Zeitereignis)

Fügt eine **Ereignisannahmeaktion** ein, die durch ein Zeitereignis ausgelöst wird, welches ein Zeitintervall - z.B. 1 sec since last update - mit einem Ausdruck definiert.



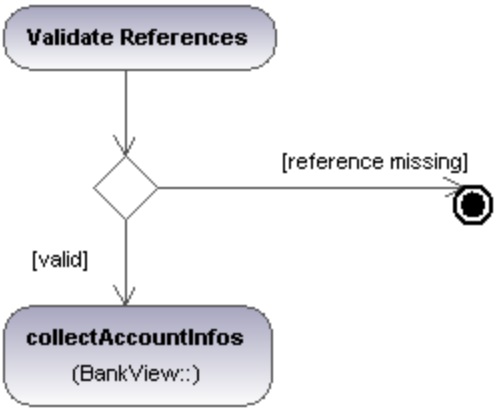
Signalsendeaktion

Fügt die **Signalsendeaktion** ein. Dabei wird ein Signal von den Inputs erstellt. Dieses Signal wird an das Zielobjekt übertragen, wo es die Ausführung einer Aktivität auslösen kann.



Verzweigungsknoten

Fügt einen Verzweigungsknoten ein, welcher eine einzige eingehende Transition hat und mehrere ausgehende mit Guards versehene Transitionen. Nähere Informationen dazu finden Sie unter "Erstellen einer Verzweigung 306".



Verbindungsknoten

Fügt einen Verbindungsknoten ein, der mehrere durch einen Verzweigungsknoten definierte alternative Transitionen zusammenführt. Die gleichzeitig ablaufenden Prozesse werden dabei nicht synchronisiert, sondern es wird einer der Prozesse ausgewählt.

Startknoten

Der Anfang der Aktivität. Eine Aktivität kann mehrere Startknoten haben.

Aktivitätsendknoten

Das Ende der Aktivität. Eine Aktivität kann mehrere Endknoten haben. Alle Flüsse in der Aktivität werden beendet, wenn der "erste" Endknoten erreicht wird.

Endknoten für Kontrollflüsse

Fügt den Endknoten für Kontrollflüsse ein, d.h. ein Fluss wird dadurch beendet. Diese Beendung hat keinen Einfluss auf andere Flüsse in der Aktivität.

 © 2016-2022 Altova GmbH

Parallelisierungsknoten

Fügt einen vertikalen Parallelisierungsknoten ein.
Dient zum Aufteilen von Flüssen in mehrere gleichzeitige Flüsse.

Parallelisierungsknoten (horizontal)

Fügt einen horizontalen Parallelisierungsknoten ein.
Dient zum Aufteilen von Flüssen in mehrere gleichzeitige Flüsse.

Synchronisationsknoten

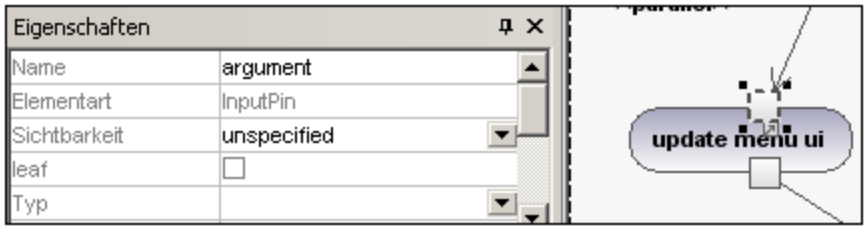
Fügt einen vertikalen Parallelisierungsknoten ein.
Ein Synchronisationsknoten synchronisiert mehrere durch einen Parallelisierungsknoten definierte Flüsse.

Synchronisationsknoten (horizontal)

Fügt einen horizontalen Parallelisierungsknoten ein.
Ein Synchronisationsknoten synchronisiert mehrere durch einen Parallelisierungsknoten definierte Flüsse.

InputPin

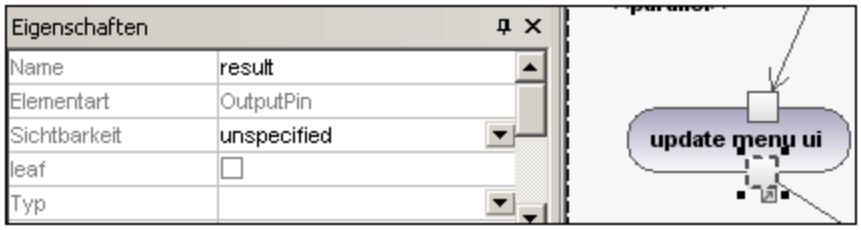
Fügt einen Input Pin in ein Aufrufverhalten oder eine Aufrufoperation ein. Input Pins liefern Eingabewerte, die von einer Aktion verwendet werden. Ein Input Pin erhält automatisch den Standardnamen "argument".



Das Input Pin-Symbol kann nur auf jene Aktivitätselemente platziert werden, bei denen der Mauszeiger sich in das Symbol einer Hand verwandelt. Wenn Sie das Symbol mit der Maus ziehen, wird es am Rand des Elements neu positioniert.

OutputPin

Fügt eine Output Pin-Aktion ein. Output Pins enthalten Werte, die von einer Aktion erzeugt werden. Dem Output Pin wird automatisch ein Name zugewiesen, der der UML-Eigenschaft dieser Aktion - z.B. "result" - entspricht.



Das Output Pin-Symbol kann nur auf jene Aktivitätselemente platziert werden, bei denen der Mauszeiger sich in eine Hand verwandelt. Wenn Sie das Symbol mit der Maus ziehen, wird es am Rand des Elements neu positioniert.

Ausnahmepin

Ein Output Pin kann durch Klicken auf den Pin und Auswahl der Option "isExceptionPin" im Fenster "Eigenschaften" in einen Ausnahmepin umgewandelt werden.

Wertpin

Fügt einen Wertpin ein. Dabei handelt es sich um einen Input Pin, der für eine Aktion einen Wert bereitstellt, der nicht aus einem eingehenden Objektfluss stammt. Er wird als ein Input Pin-Symbol dargestellt und hat dieselben Eigenschaften wie ein Input Pin.

Objektknoten

Fügt einen Objektknoten ein. Ein Objektknoten ist ein abstrakter Aktivitätsknoten, der den Objektfluss in einer Aktivität definiert. Objektknoten können nur Werte zur Laufzeit enthalten, die dem Typ des Objektknotens entsprechen.

Pufferknoten

Fügt einen Pufferknoten ein, der als Puffer für mehrere eingehende und ausgehende Flüsse von anderen Objektknoten dient.

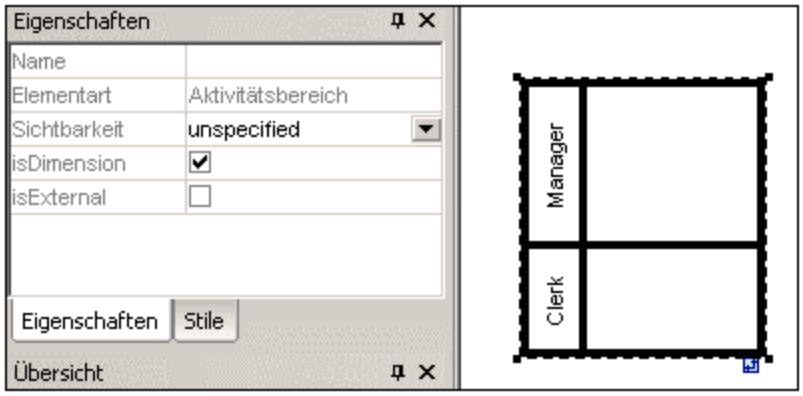
Datenspeicherknoten

Fügt einen Datenspeicherknoten ein. Dabei handelt es sich um einen speziellen Pufferknoten, der zum Speichern dauerhafter (d.h. nicht temporärer) Daten dient.

Aktivitätsbereich (horizontal)

Fügt einen horizontalen Aktivitätsbereich ein. Dabei handelt es sich um eine Art von Aktivitätsgruppe, die zum Kennzeichnen von Aktionen dient, die einige Eigenschaften gemeinsam haben. Dies entspricht oft Organisationseinheiten in einem Geschäftsmodell.

© 2016-2022 Altova GmbH



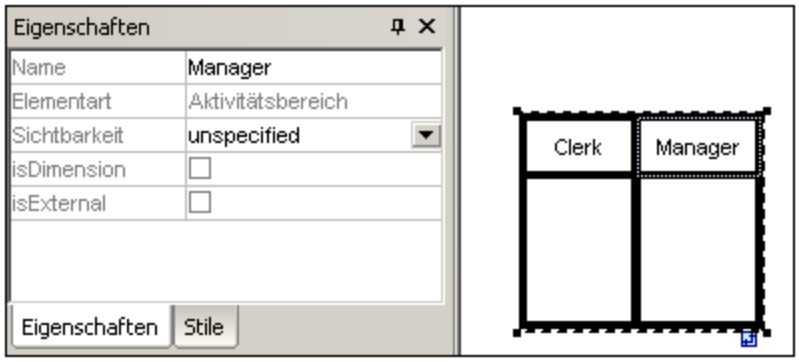
Wenn Sie auf eine Beschriftung doppelklicken, lässt sich diese direkt editieren; bei Drücken der Eingabetaste wird der Text korrekt ausgerichtet.

Bitte beachten Sie, dass Aktivitätsbereiche in UML 2.0 neu eingeführt wurden und die "swimlane" Funktion früherer UML-Versionen ersetzen.

- Elemente, die innerhalb einer ActivityPartition platziert werden, werden Teil davon, sobald die Umrandung markiert erscheint.
  - Objekte innerhalb einer ActivityPartition können mit Hilfe von Strg+Klick oder durch Aufziehen eines Rechtecks in der Umrandung einzeln ausgewählt werden.
  - Klicken Sie auf die Umrandung oder den titel der ActivityPartition und ziehen Sie sie/ihn, um diese an eine andere Stelle zu verschieben.

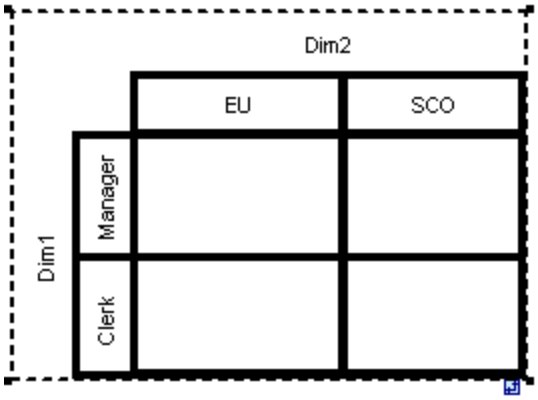
Aktivitätsbereich (vertikal)

Fügt einen vertikalen Aktivitätsbereich ein. Dabei handelt es sich um eine Art von Aktivitätsgruppe, die zum Kennzeichnen von Aktionen dient, die einige Eigenschaften gemeinsam haben. Dies entspricht oft Organisationseinheiten in einem Geschäftsmodell.



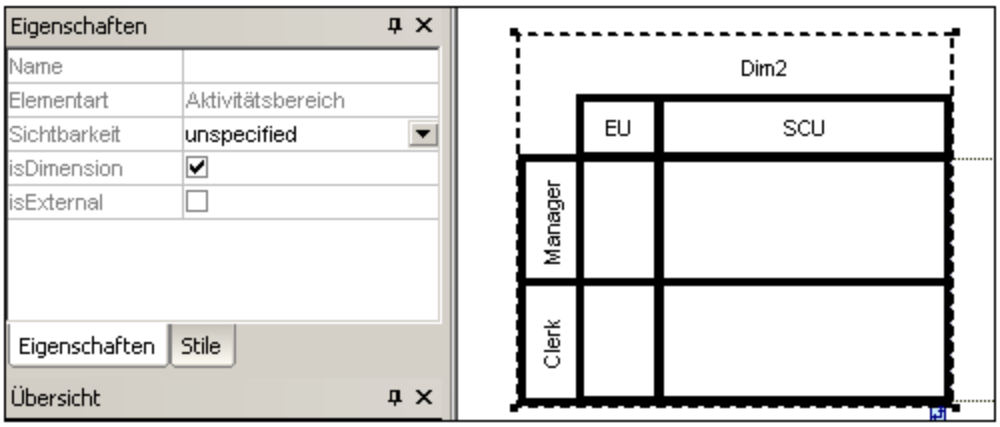
Aktivitätsbereich (2 Dimensionen)

Fügt einen zweidimensionalen Aktivitätsbereich ein. Dabei handelt es sich um eine Art von Aktivitätsgruppe, die zum Kennzeichnen von Aktionen dient, die einige Eigenschaften gemeinsam haben. Beide Achsen haben editierbare Beschriftungen.



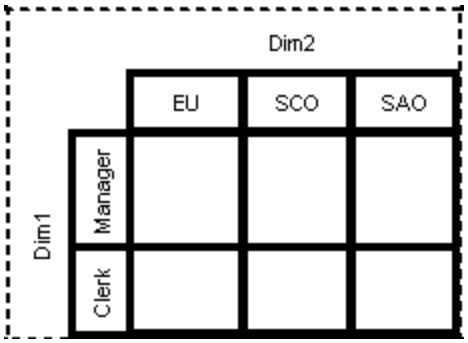
So entfernen Sie die Dim1, Dim2-Beschriftungen:

1. Klicken Sie auf die Dimensionsbezeichnung, die Sie entfernen möchten z.B. Dim1
2. Doppelklicken Sie auf dem Register "Eigenschaften" auf den Eintrag Dim1, löschen Sie ihn und bestätigen Sie den Vorgang mit der Eingabetaste.



Beachten Sie, dass Aktivitätsbereiche geschachtelt sein können:

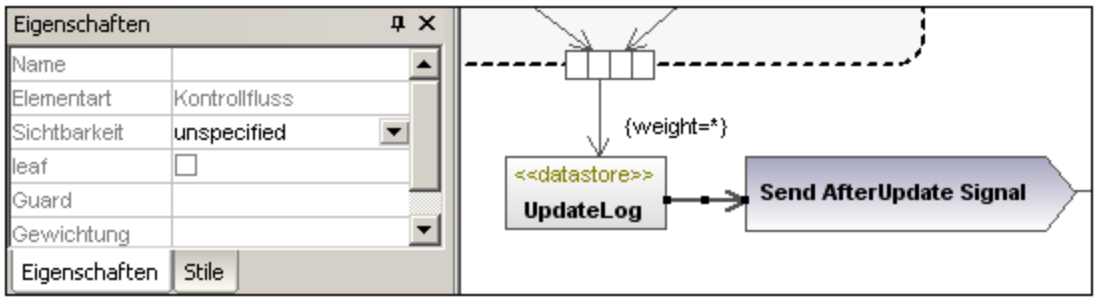
1. Rechtsklicken Sie auf die Beschriftung, wo Sie einen neuen Bereich einfügen möchten.
2. Wählen Sie **Neu | Aktivitätsbereich**.



 © 2016-2022 Altova GmbH

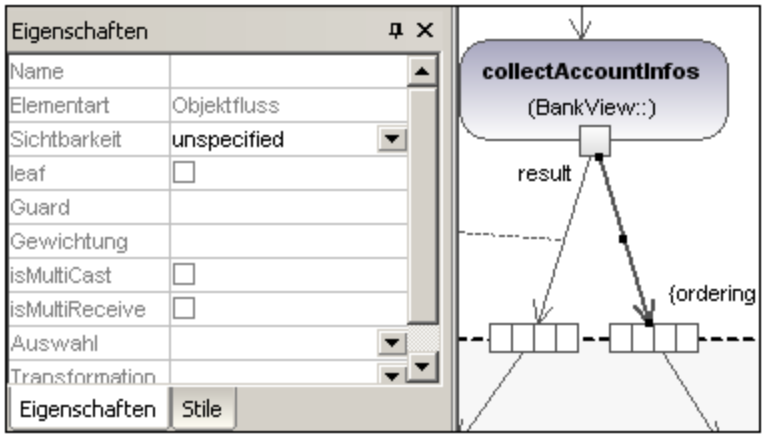
Kontrollfluss

Ein Kontrollfluss ist eine Kante, d.h. ein Pfeil, der zwei Aktivitäten/Verhalten verbindet und eine Aktivität startet, nachdem die vorherige zu Ende geführt wurde.



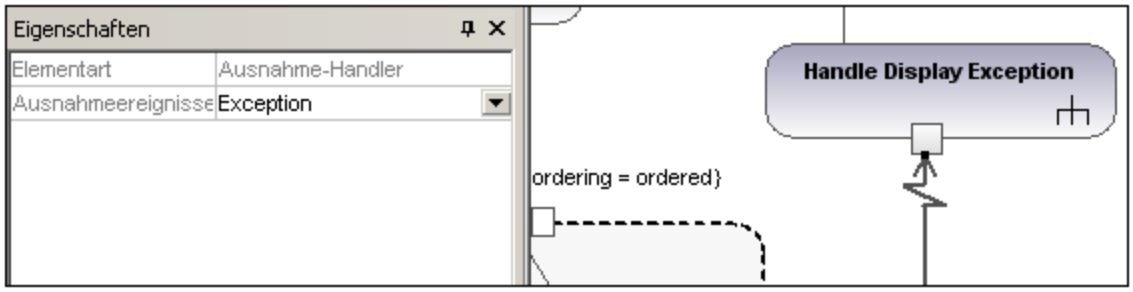
Objektfluss

Ein Objektfluss ist eine Kante, d.h. ein Pfeil, der zwei Aktionen/Objektknoten verbindet und eine Aktivität startet, sobald die vorherige zu Ende geführt worden ist. Entlang eines Objektflusses können Objekte oder Daten übergeben werden.



Ausnahme-Handler

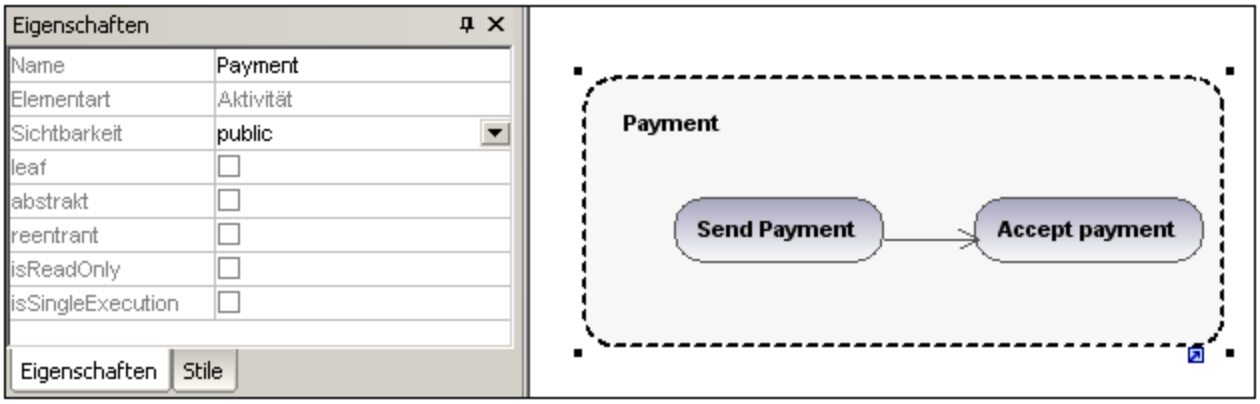
Ein Ausnahme-Handler ist ein Element, das festlegt, welche Aktion ausgeführt werden soll, wenn während der Ausführung des geschützten Knotens ein bestimmtes Ausnahmeereignis auftritt.



Ein Ausnahme-Handler kann nur auf einen Inputpin einer Aktion gezogen werden.

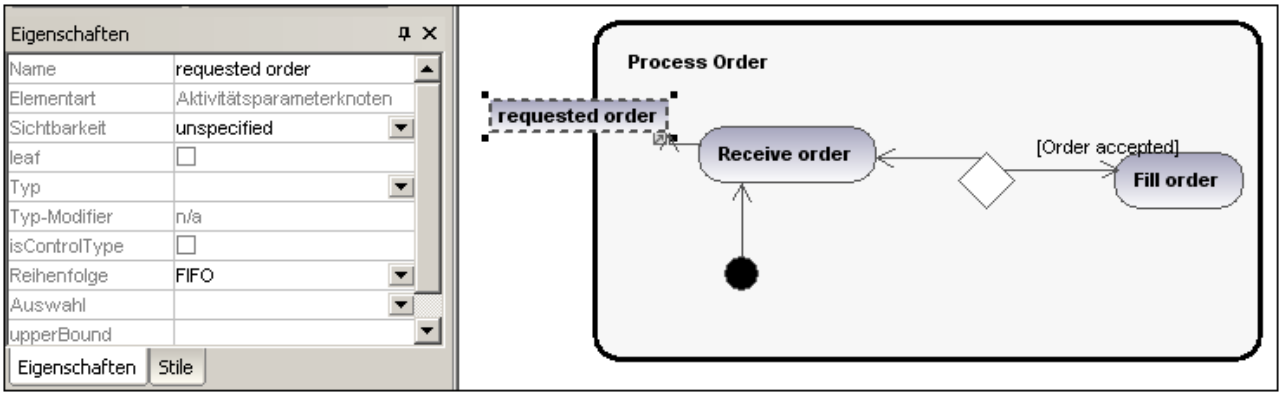
Aktivität

Fügt eine Aktivität in das Aktivitätsdiagramm ein.



Aktivitätsparameterknoten

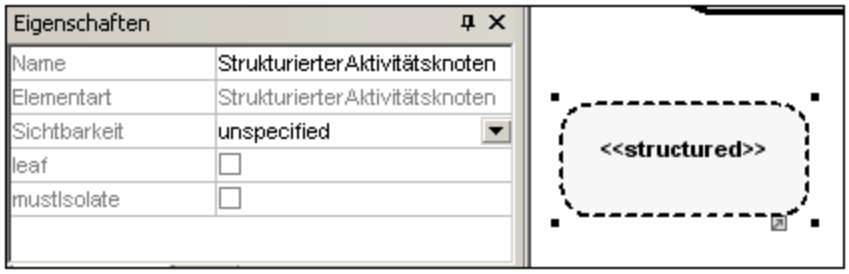
Fügt einen Aktivitätsparameterknoten in eine Aktivität ein. Wenn Sie an eine beliebige Stelle in der Aktivität klicken, wird der Parameterknoten auf den Rand der Aktivität platziert.



StrukturierterAktivitätsknoten

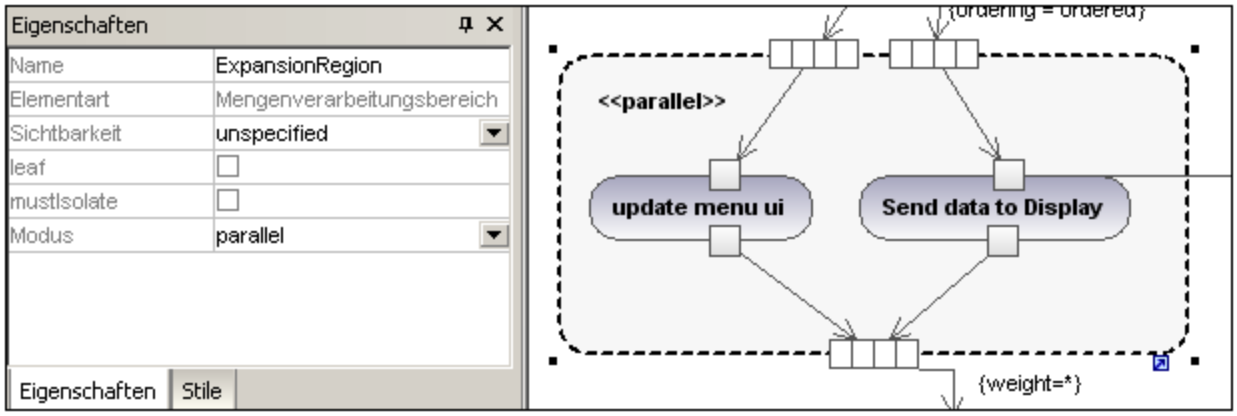
Fügt einen strukturierten Aktivitätsknoten ein. Dabei handelt es sich um einen strukturierten Teil der Aktivität, der mit keinem anderen strukturierten Knoten geteilt wird.

© 2016-2022 Altova GmbH



Mengenverarbeitungsbereich

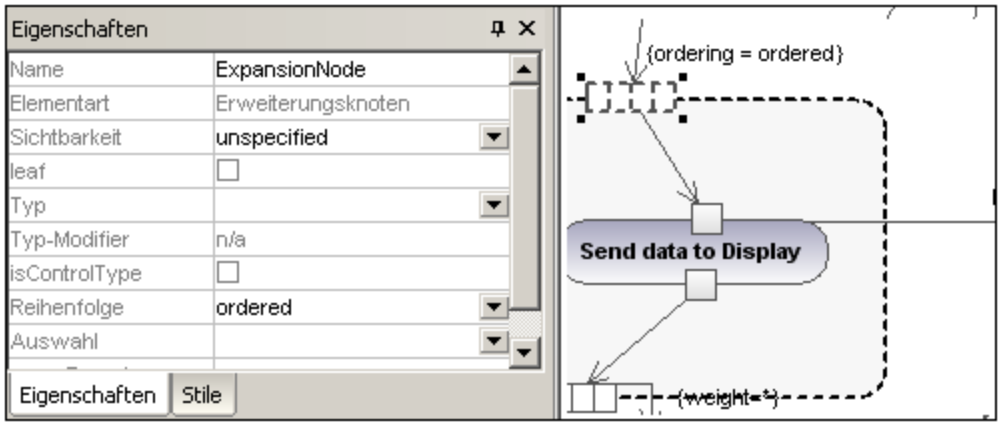
Ein Mengenverarbeitungsbereich ist ein Bereich einer Aktivität mit expliziten Inputs und Outputs (über Erweiterungsknoten). Jeder Input ist eine Sammlung von Werten.



Der Mengenverarbeitungsbereichsknoten wird als Schlüsselwort angezeigt und kann durch Klicken auf die Auswahlliste "Modus" auf dem Register "Eigenschaften" geändert werden. Zur Auswahl stehen die Einstellungen: parallel, iterative oder stream.

Erweiterungsknoten

Fügt einen Erweiterungsknoten in einen Mengenverarbeitungsbereich ein. Erweiterungsknoten sind Eingabe- und Ausgabeknoten für den Mengenverarbeitungsbereich, wobei jeder Input/Output eine Sammlung von Werten ist. Die Pfeile, die in den Mengenverarbeitungsbereich hinein oder daraus heraus weisen , legen die jeweilige Art des Erweiterungsknotens fest.



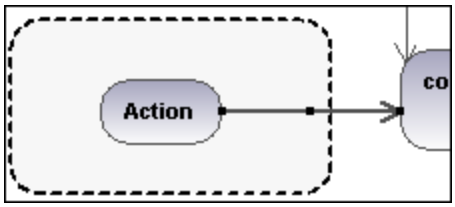
Unterbrechungsbereich

Ein Unterbrechungsbereich enthält Aktivitätsknoten. Wenn ein Kontrollfluss einen Unterbrechungsbereich verlässt, werden alle Flüsse und Verhalten im Bereich beendet.

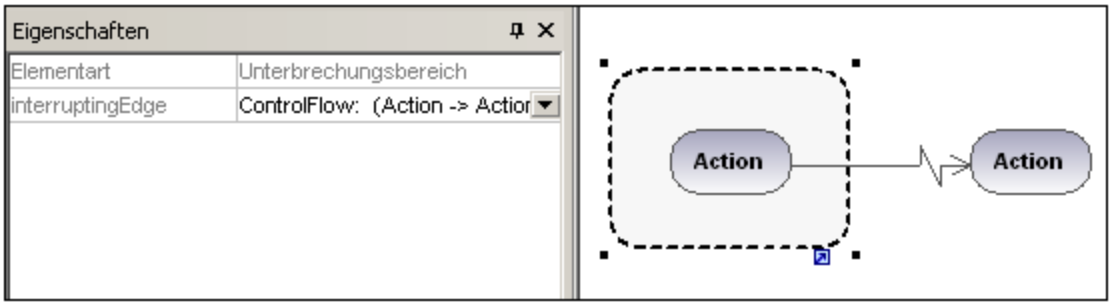
So fügen Sie eine InterruptingEdge hinzu:

Stellen Sie sicher dass:

- der Unterbrechungsbereich ein Aktionselement sowie einen ausgehenden Kontrollfluss zu einer anderen Aktion enthält:



1. Rechtsklicken Sie auf den Kontrollflusspfeil und wählen Sie **Neu | InterruptingEdge**.



Anmerkung:

Sie können eine InterrupingEdge auch durch Klicken auf den Unterbrechungsbereich, Rechtsklick in das Fenster "Eigenschaften" und Auswahl des Popup-Menübefehls "InterruptingEdge hinzufügen" hinzufügen.

© 2016-2022 Altova GmbH

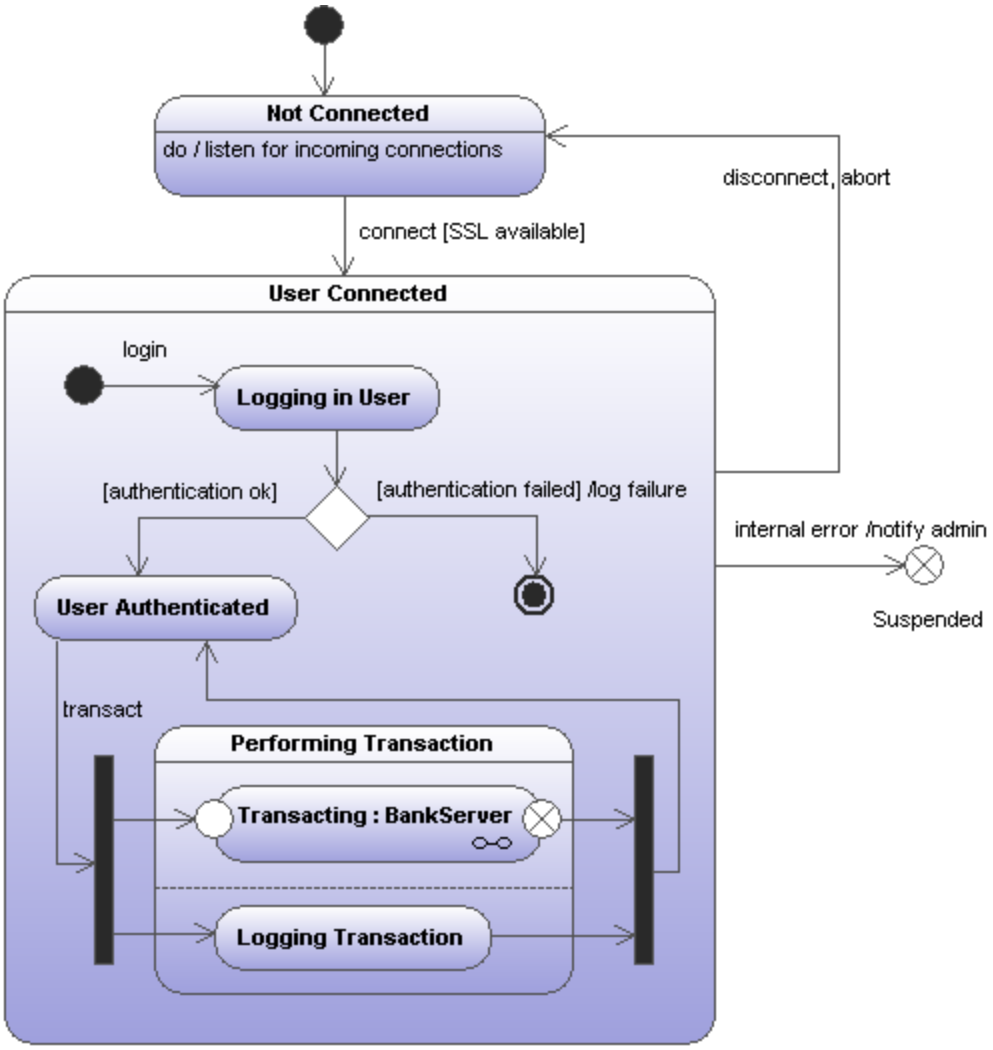
8.1.2 Zustandsdiagramm

Im Zustandsdiagramm wird das Verhalten eines Systems modelliert, indem die Zustände, in denen sich ein Objekt befinden kann und die Übergänge zwischen diesen Zuständen, beschrieben werden. Im Allgemeinen dienen diese Diagramme dazu, das Verhalten eines Objekts in verschiedenen Anwendungsfällen (Use Cases) zu beschreiben.

Dies kann in Form von zwei Arten von Prozessen erfolgen:

1. **Aktionen,** die mit **Transitionen** verknüpft sind, sind kurzfristige Prozesse, die nicht unterbrochen werden können (z.B. **internal error /notify admin** im Diagramm unten)
2. **Zustandsaktivitäten** (Verhalten), welche mit **Zuständen** verknüpft sind, sind länger andauernde Prozesse, die durch andere Ereignisse unterbrochen werden können (z.B. **listen for incoming connections** im Diagramm unten)

Ein Zustandsautomat kann in UModel beliebig viele Zustandsdiagramme haben.



Zustandsbeispieldiagramm

Das oben gezeigte Zustandsdiagramme steht im folgenden UModel-Beispielprojekt zur Verfügung: **C:\Benutzer\<Benutzername>\Dokumente\Altova\UModel2023\UModelExamples\Bank_MultiLanguage.ump**.

8.1.2.1 Einfügen von Zustandsdiagrammelementen

Einfügen über die Symbole der Symbolleiste:

1. Klicken Sie in der Zustandsdiagramm-Symbolleiste auf das entsprechende Zustandsdiagramm-Symbol.

2. Klicken Sie in das Zustandsdiagramm, um das Element einzufügen. Um mehrere Elemente des ausgewählten Typs einzufügen, halten Sie die **Strg-Taste** gedrückt und klicken Sie in das Diagrammfenster.

Ziehen von bestehenden Elementen in das Zustandsdiagramm

Die meisten Elemente, die in anderen Zustandsdiagrammen vorkommen, können in ein bestehendes Zustandsdiagramm eingefügt werden.

1. Suchen Sie das gewünschte Element auf dem Register Modell-Struktur (Sie können dazu das Suchfunktionstextfeld verwenden oder Strg + F drücken, um nach einem beliebigen Element zu suchen).
2. Ziehen Sie das/die Element(e) in das Zustandsdiagramm.

8.1.2.2 Erstellen von Zuständen, Aktivitäten und Transitionen

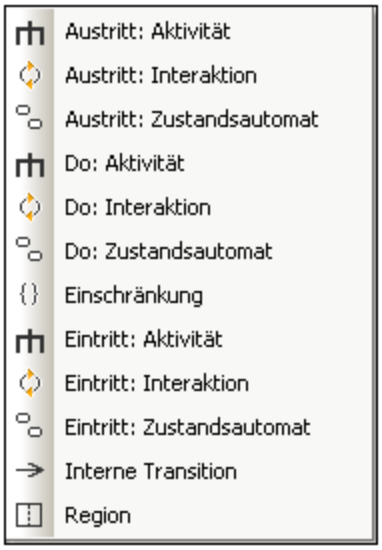
So fügen Sie einen einfachen Zustand hinzu:

1. Klicken Sie in der Symbolleiste auf das Symbol **Zustand** ( ) und anschließend in das Diagramm.
2. Geben Sie den Namen des Zustands ein und drücken Sie zur Bestätigung die **Eingabetaste**.

So fügen Sie eine Aktivität zu einem Zustand hinzu:

- Rechtsklicken Sie auf das Element "Zustand", wählen Sie "Neu" und anschließend einen der Einträge aus dem Kontextmenü.

 © 2016-2022 Altova GmbH

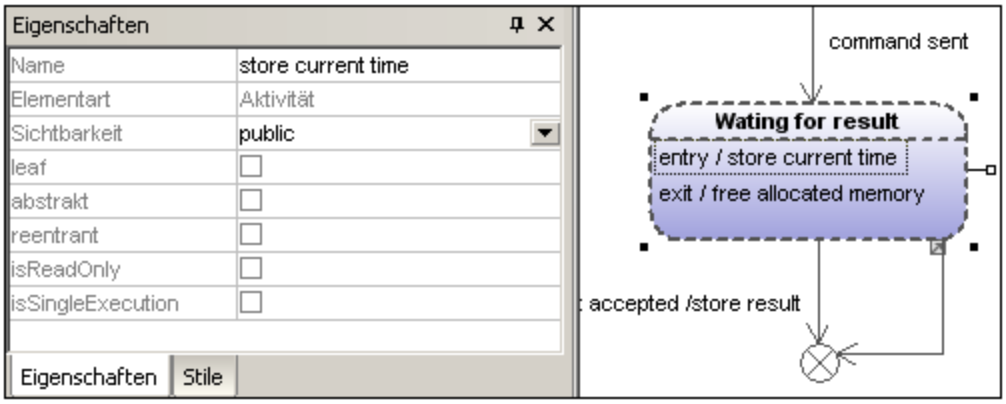


Die Aktivitäten **Eintritt**, **Austritt** und **Do** sind mit einem der folgenden möglichen Verhalten verknüpft: "Aktivität", "Interaktion" und "Zustandsautomat". Daher stehen im Kontextmenü die folgenden Optionen zur Verfügung:

- Austritt: Aktivität
- Austritt: Interaktion
- Austritt: Zustandsautomat
- Do: Aktivität
- Do: Interaktion
- Do: Zustandsautomat
- Eintritt: Aktivität
- Eintritt: Interaktion
- Eintritt: Zustandsautomat

Diese Optionen stammen aus der UML-Spezifikation. Bei jeder dieser internen Aktionen handelt es sich um ein Verhalten und in der UML-Spezifikation sind drei Klassen von der Klasse "Behavior" (Verhalten) abgeleitet: Activity (Aktivität), StateMachine (Zustandsautomat) und Interaction (Interaktion). Es macht im generierten Code keinen Unterschied, welches spezifische Verhalten (Aktivität, Zustandsautomat oder Interaktion) ausgewählt wurde.

Sie haben die Wahl zwischen Aktionen aus der Kategorie: Eintritts, Austritt und Do. Aktivitäten werden im Zustandselement in ihren eigenen Bereich - wenn auch nicht in eine separate Region - platziert. Die Art der ausgewählten Aktivität wird als Präfix für die Aktivität verwendet z.B: **entry / store current time** (=Eintritt / aktuelle Zeit speichern).

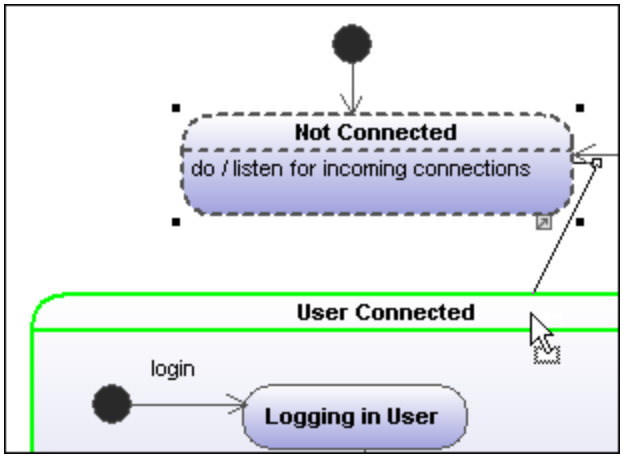


So löschen Sie eine Aktivität:

- Klicken Sie auf die entsprechende Aktivität im Zustandselement und drücken Sie die Entf-Taste.

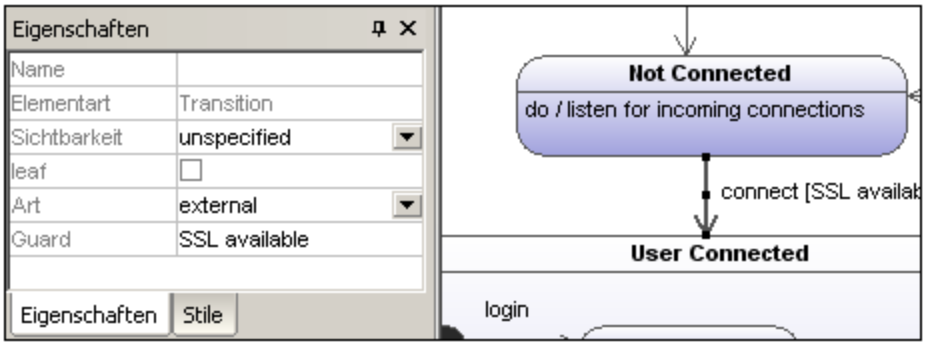
So erstellen Sie eine Transition zwischen zwei Zuständen:

1. Klicken Sie auf den Transition-Ziehpunkt des Ausgangszustands (auf der rechten Seite des Elements).
2. Ziehen Sie den Transition-Pfeil mit der Maus auf den Ziel-Zustand.



Die Eigenschaften der Transition werden nun auf dem Register "Eigenschaften" angezeigt. Wenn Sie auf die Auswahlliste "Art" klicken, können Sie den Typ der Transition definieren: extern, intern oder lokal.

 © 2016-2022 Altova GmbH



Transitionen können einen Event Trigger, eine Guard-Bedingung und eine Aktion in der Form **eventTrigger [Guard-Bedingung] /Aktivität** haben.

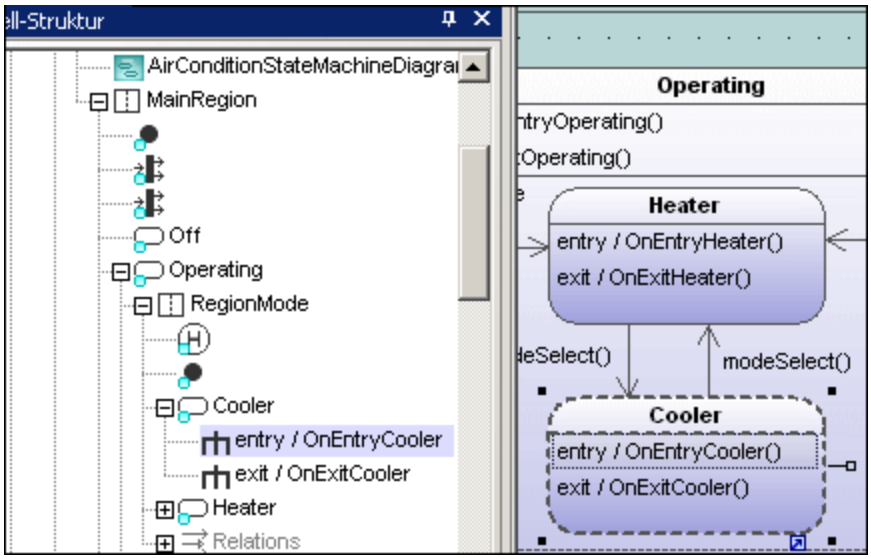
So erstellen Sie automatisch Operationen von Transitionen aus

Wenn Sie die Schaltfläche "Automatische Erstellung von Operationen in der Zielkomponente durch Eingabe von Operationsnamen ein-/ausschalten" aktivieren, wird die entsprechende Operation in der referenzierten Klasse automatisch erstellt, wenn Sie eine Transition erstellen und einen Namen,z. B. myOperation() eingeben.

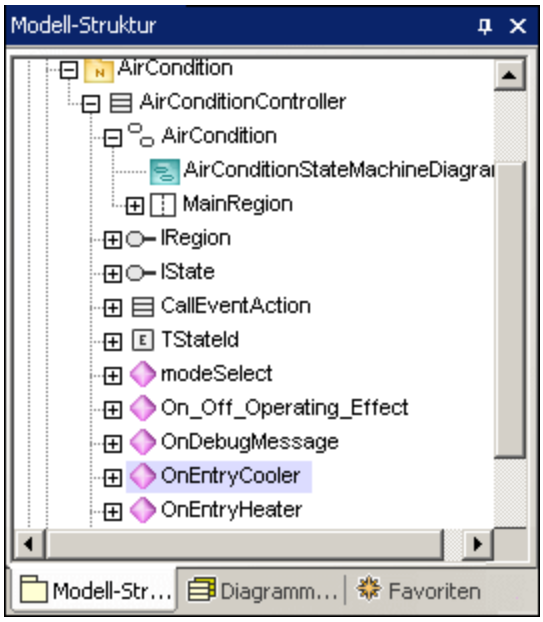
Anmerkung: Operationen können nur dann automatisch erstellt werden, wenn sich der Zustandsautomat innerhalb einer Klasse oder Schnittstelle befindet.

So erstellen Sie automatisch Operationen von Aktivitäten aus:

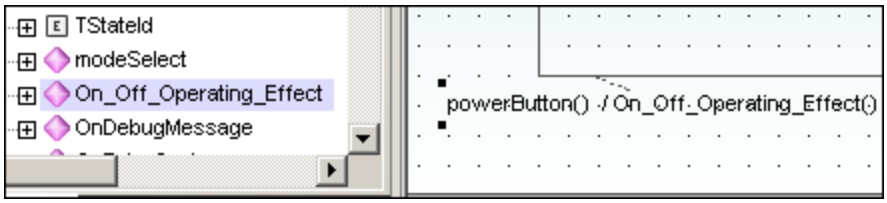
1. Klicken Sie mit der rechten Maustaste auf den Zustand und wählen Sie die entsprechende Aktion/Aktivität aus, z.B. Neu | Eintritt:Aktivität.
2. Geben Sie den Namen der Aktivität ein und stellen Sie sicher, dass Sie am Schluss das "Klammer auf" und "Klammer zu"-Zeichen "**()**", also z.B. **Eintritt / OnEntryCooler()** eingeben.



Das neue Element wird auch in der Modell-Struktur angezeigt.
Wenn Sie einen Bildlauf nach unten durch die Modell-Struktur durchführen, sehen Sie, dass die Operation OnEntryCooler zur übergeordneten Klasse AirConditionController hinzugefügt wurde.



Anmerkung:
Operationen werden automatisch für: Do:Aktivität, Eintritt:Aktivität, Austritt:Aktivität sowie als Guard-Bedingungsaktivitäten und Auswirkungen (in Transitionen) hinzugefügt.



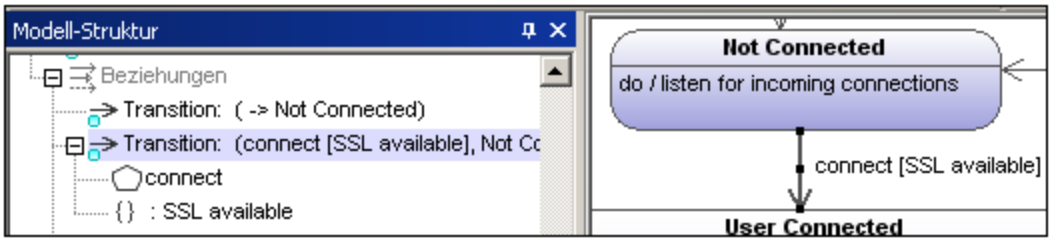
So erstellen Sie einen Transition Trigger:

1. Rechtsklicken Sie auf eine zuvor erstelle Transition (Pfeil).
2. Wählen Sie **Neu | Trigger**.

   Über dem Transition-Pfeil wird in der Transition-Bezeichnung der Buchstabe "a" angezeigt, falls es sich um den ersten Trigger im Zustandsdiagramm handelt. Den Triggern werden Standardwerte in der Form: Buchstabe des Alphabets, Ausgangszustand -> Zielzustand zugewiesen.
3. Doppelklicken Sie auf den neuen Buchstaben und geben Sie die Eigenschaften der Transition in der Form **eventTrigger [Guard-Bedingung] /Aktivität** ein.

> Syntax der Eigenschaft der Transition:
> Der Text vor der eckigen Klammer ist der Trigger, innerhalb der eckigen Klammer befindet sich die Guard-Bedingung und hinter dem Schrägstrich folgt die Aktivität. Bei Bearbeitung des Strings werden automatisch die entsprechenden Elemente in der Modell-Struktur erstellt bzw. gelöscht.

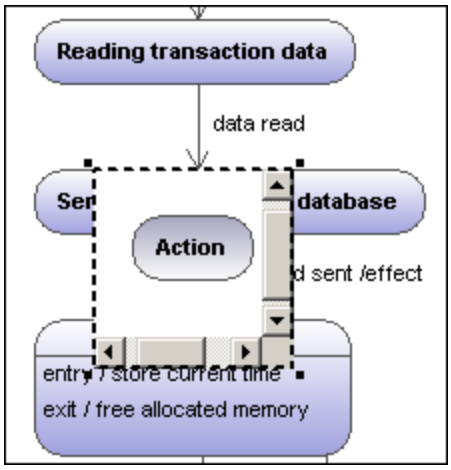
Anmerkung:
Um die Eigenschaften der einzelnen Transition zu sehen, rechtsklicken Sie auf die Transition (also auf den Pfeil) und wählen Sie die Option "In Modell-Struktur auswählen". Das Ereignis, die Aktivität und die Einschränkungselemente werden alle unterhalb der ausgewählten Transition angezeigt.



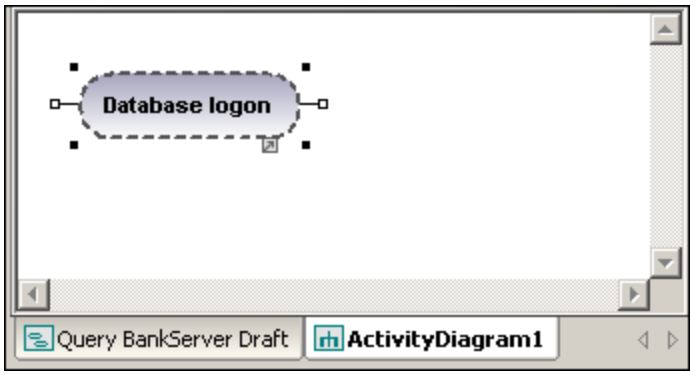
Hinzufügen eines Aktivitätsdiagramms zu einer Transition

UModel bietet die einzigartige Möglichkeit, zur näheren Beschreibung der Transition ein Aktivitätsdiagramm zu einer Transition hinzuzufügen.

1. Rechtsklicken Sie auf einen Transitionspfeil im Diagramm und wählen Sie **Neu | Aktivitätsdiagramm**. Daraufhin wird an der Position des Transitionspfeils ein Aktivitätsdiagrammfenster in das Diagramm eingefügt.
2. Klicken Sie auf das eingefügte Fenster, um es aktiv zu machen. Über die Bildlaufleisten können Sie durch das Fenster scrollen.



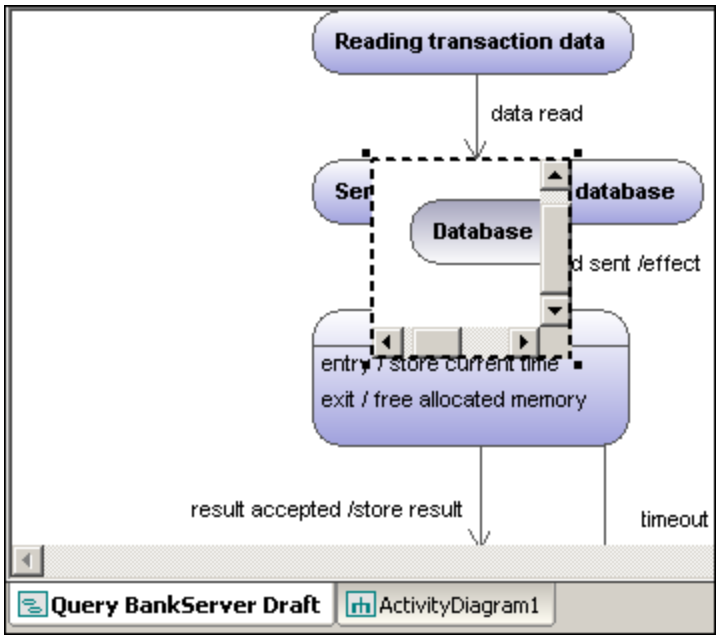
3. Doppelklicken Sie auf das Aktionsfenster um in das Aktivitätsdiagramm zu wechseln und die Transition näher zu definieren, z.B. um den Namen der Aktion in "Database logon" zu ändern.



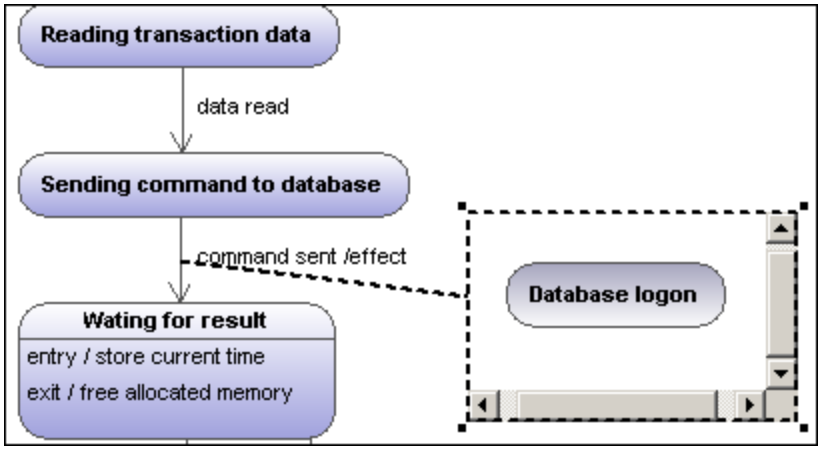
Beachten Sie, dass ein neues Aktivitätsdiagrammregister zum Projekt hinzugefügt wurde. Sie können jedes beliebige Aktivitätsdiagramm-Modellelement zum Diagramm hinzufügen. Nähere Informationen dazu finden Sie unter "Aktivitätsdiagramm 302".

4. Klicken Sie auf das Zustandsdiagrammregister um zurückzuwechseln und die aktualisierte Transition zu sehen.

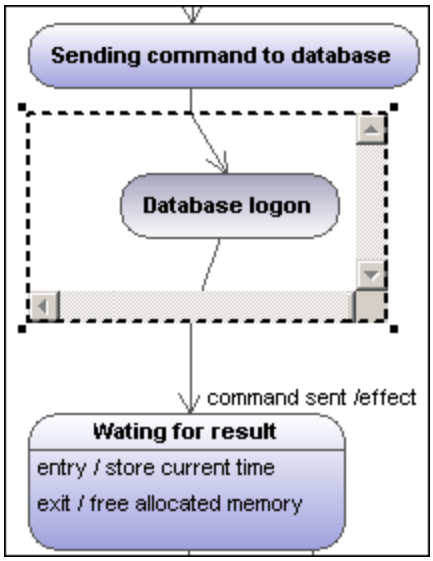
 © 2016-2022 Altova GmbH



5. Ziehen Sie gegebenenfalls das Aktivitätsfenster an die gewünschte Stelle im Diagramm und klicken Sie auf den Ziehpunkt zur Größenanpassung.



Wenn Sie das Aktivitätsfenster zwischen die beiden Zustände ziehen, wird die Transition in die Aktivität und aus der Aktivität heraus angezeigt.



8.1.2.3 Zusammengesetzte Zustände

Zusammengesetzter Zustand

Diese Art von Zustand enthält einen zweiten Bereich bestehend aus einer einzigen Region. Innerhalb dieser Region können beliebig viele Zustände platziert werden.

So fügen Sie eine Region zu einem zusammengesetzten Zustand hinzu:

1. Rechtsklicken Sie auf den zusammengesetzten Zustand und wählen Sie im Kontextmenü den Befehl **Neu | Region**.
   Daraufhin wird eine neue Region zum Zustand hinzugefügt. Regionen werden durch strichlierte Linien voneinander getrennt.

So löschen Sie eine Region:

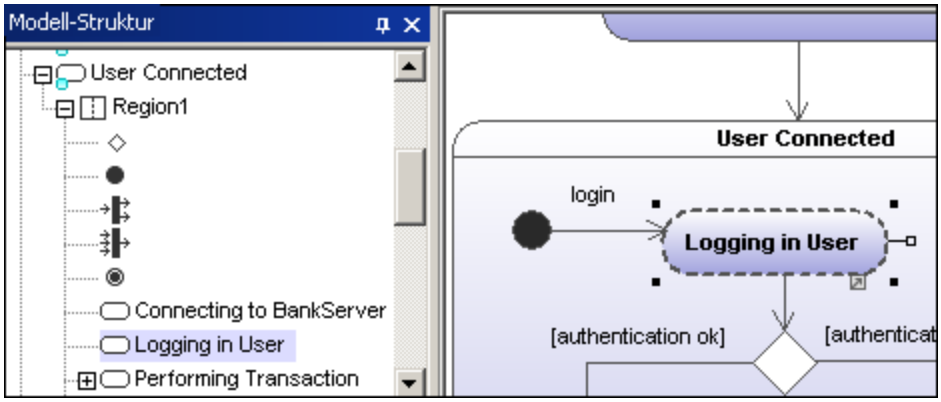
1. Klicken Sie auf die gewünschte Region im zusammengesetzten Zustand und drücken Sie die Entf-Taste.
   Wenn Sie eine Region eines orthogonalen Zustands löschen, so wird dieser wieder zu einem zusammengesetzten Zustand; sobald die letzte Region eines zusammengesetzten Zustands gelöscht wurde, wird dieser wieder zu einem einfachen Zustand.

So platzieren Sie einen Zustand in einen zusammengesetzten Zustand:

1. Klicken Sie auf das gewünschte Zustandselement (z.B. Logging in User) und ziehen Sie es in den Regionsbereich des zusammengesetzten Zustands.

 © 2016-2022 Altova GmbH

Der Regionsbereich erscheint markiert, sobald Sie die Maustaste loslassen können. Das eingefügte Element ist nun Teil der Region und wird im Fenster "Modell-Struktur" als Child-Element der Region angezeigt.

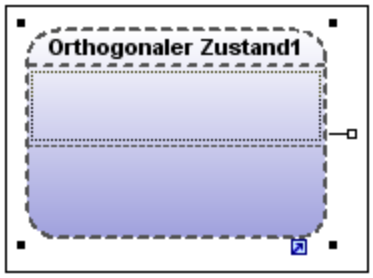


Wenn Sie den zusammengesetzten Zustand verschieben, werden alle darin enthaltenen Zustände mitverschoben.

Orthogonaler Zustand

Diese Art von Zustand enthält einen zweiten Bereich bestehend aus zwei oder mehr Regionen, wobei die einzelnen Regionen auf Gleichzeitigkeit hinweisen.

Wenn Sie mit der rechten Maustaste auf einen Zustand klicken und **Neu | Region** auswählen, können Sie neue Regionen hinzufügen.



So blenden Sie Regionsnamen ein/aus:

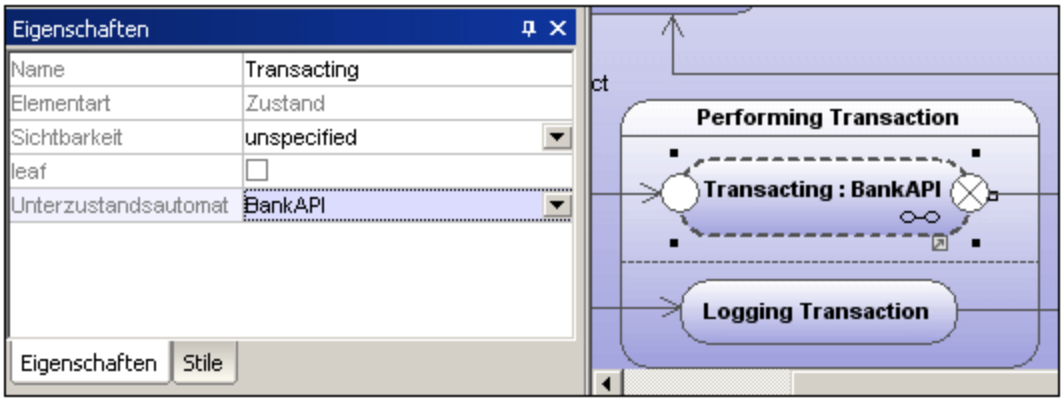
Klicken Sie auf das Register "Stile", scrollen Sie zum Eintrag "Regionsnamen in Zustandselementen anzeigen" und wählen Sie true/false aus.

Unterautomatenzustand

Dieser Zustand dient zum Ausblenden der Einzelheiten eines Zustandsautomaten. Dieser Zustand hat keine Regionen, sondern ist mit einem separaten Zustandsautomaten verknüpft.

So definieren Sie einen Unterautomatenzustand:

1. Wählen Sie zuerst einen Zustand aus und klicken Sie auf dem Register "Eigenschaften" auf die Auswahlliste **Unterautomatenzustand.**
   Es erscheint eine Liste mit den derzeit definierten Zustandsautomaten.
2. Wählen Sie den Zustandsautomaten aus, den dieser Unterautomat referenzieren soll.

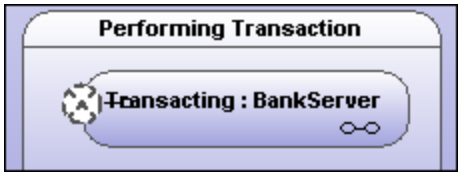


Beachten Sie bitte: Im Unterautomat wird automatisch ein Hyperlink angezeigt. Wenn Sie darauf klicken, wird der referenzierte Zustandsautomat, in diesem Fall BankServer, geöffnet.

So fügen Sie Eintritts- / Austrittspunkte zu einem Unterautomatenzustand hinzu:

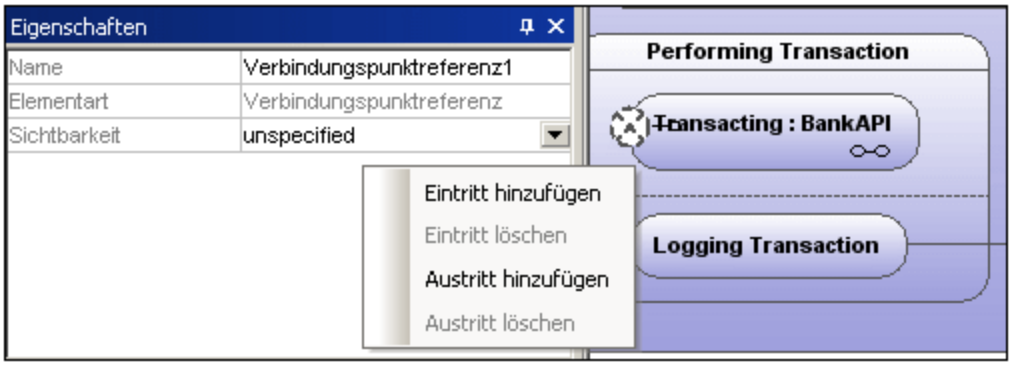
- Der Zustand, mit dem der Punkt verbunden ist, muss selbst einen Unterautomatenzustand (sichtbar auf dem Register "Eigenschaften") referenzieren.
- Dieser Unterautomat muss einen oder mehrere Eintritts- und Austrittspunkte enthalten

1. Klicken Sie in der Titelleiste auf das Symbol **Verbindungspunktreferenz** und klicken Sie anschließend auf den Unterautomatenzustand, zu dem Sie den Eintritts- / Austrittspunkt hinzufügen möchten.

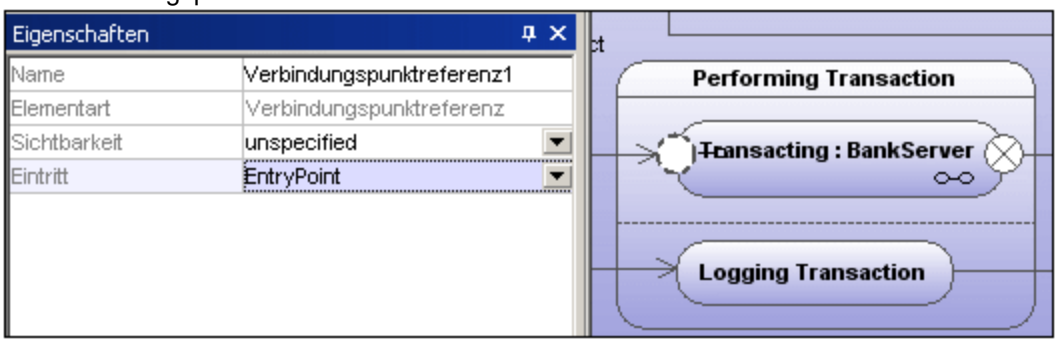


2. Rechtsklicken Sie auf das Register "Eigenschaften" und wählen Sie den Befehl "Eintritt hinzufügen". Bitte beachten Sie, dass es an einer anderen Stelle im Diagramm einen weiteren Eintritts- oder Austrittspunkt geben muss, damit dieses Popup-Menü aktiviert wird.

© 2016-2022 Altova GmbH



Daraufhin wird eine Eintrittspunktzeile zum Register "Eigenschaften" hinzugefügt und das Aussehen des Verbindungspunktreferenz-Elements ändert sich.



3. Fügen Sie auf dieselbe Weise einen Austrittspunkt hinzu. Wählen Sie dazu im Kontextmenü den Befehl "Austritt hinzufügen".

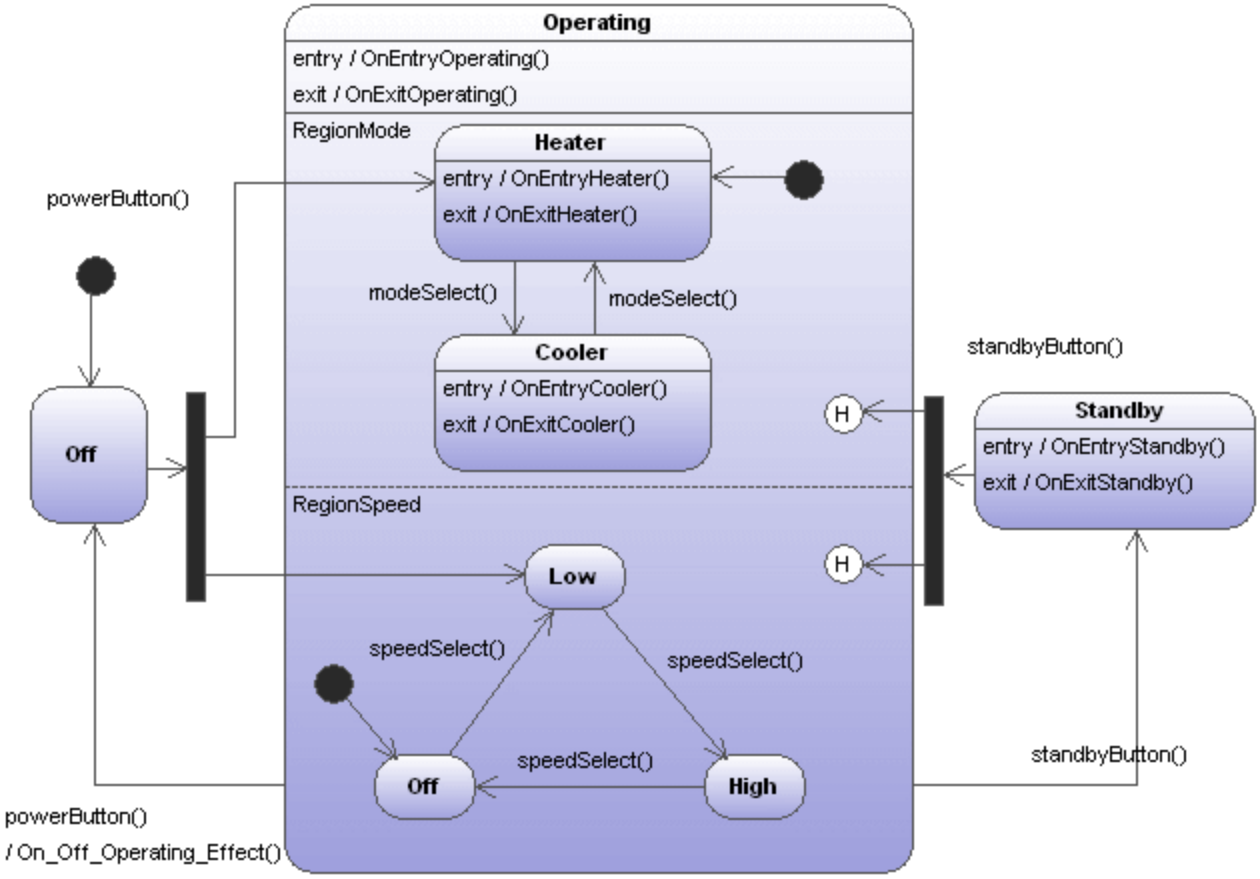
8.1.2.4 Generieren von Code anhand von Zustandsdiagrammen

Sie können in UModel ausführbaren Code (Java, VB.NET) anhand von Zustandsdiagrammen generieren. Es werden beinahe alle Zustandsdiagrammelemente und -funktionen unterstützt:

- Zustand
- Zusammengesetzter Zustand, mit jeder hierarchischen Ebene
- Orthogonaler Zustand, mit beliebig vielen Regionen
- Region
- Anfangszustand
- Endzustand
- Transition
- Guard
- Trigger
- Aufrufereignis
- Gabelung
- Vereinigung
- Entscheidung
- Kreuzung
- DeepHistory
- ShallowHistory

- Eintritts-/Austritts-/Do-Aktion
- Effekt

Die Generierung von Zustandsautomatencode ist in das normale Round Trip Engineering integriert, d.h. der Zustandsautomatencode kann bei jedem Forward Engineering automatisch aktualisiert werden.

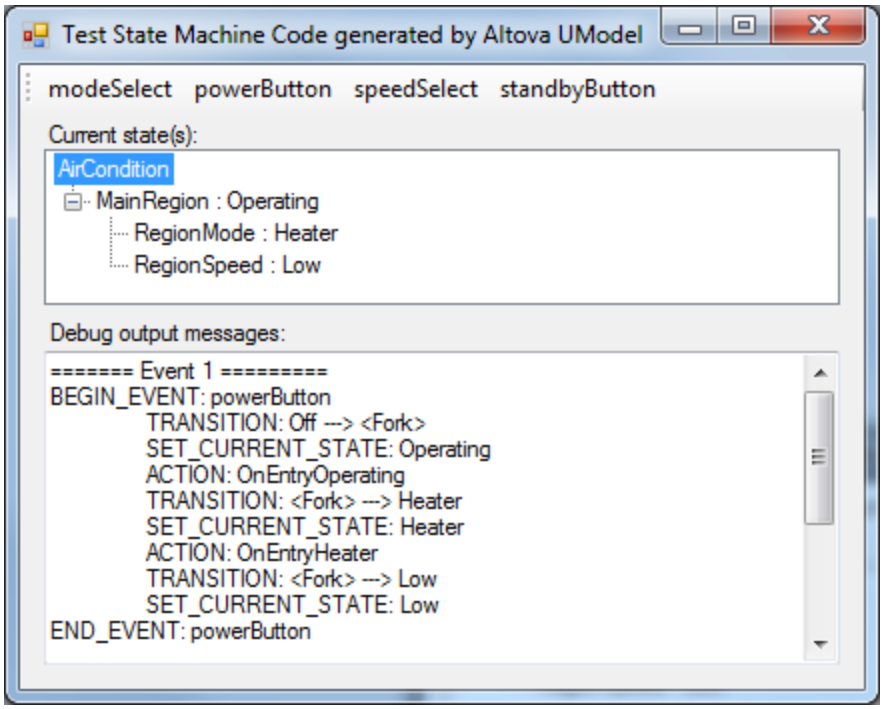


In der Abbildung oben sehen Sie das Zustandsdiagramm AirCondition aus dem Verzeichnis **.. \StateMachineCodeGeneration** unter**...\UModelExamples**. Für jede der Sprachen, in der in UModel Code generiert werden kann, gibt es ein eigenes Verzeichnis.

Jedes Verzeichnis enthält jeweils einen Ordner AirCondition und einen Ordner Complex, die jeweils das entsprechende UModel-Projekt, die Programmiersprachen-Projektdateien sowie die generierten Quelldateien enthalten. Die Projektdatei Complex.ump enthält beinahe alle der von UModel beim Generieren von Zustandsdiagrammcode unterstützten Modellierungselemente und Funktionalitäten.

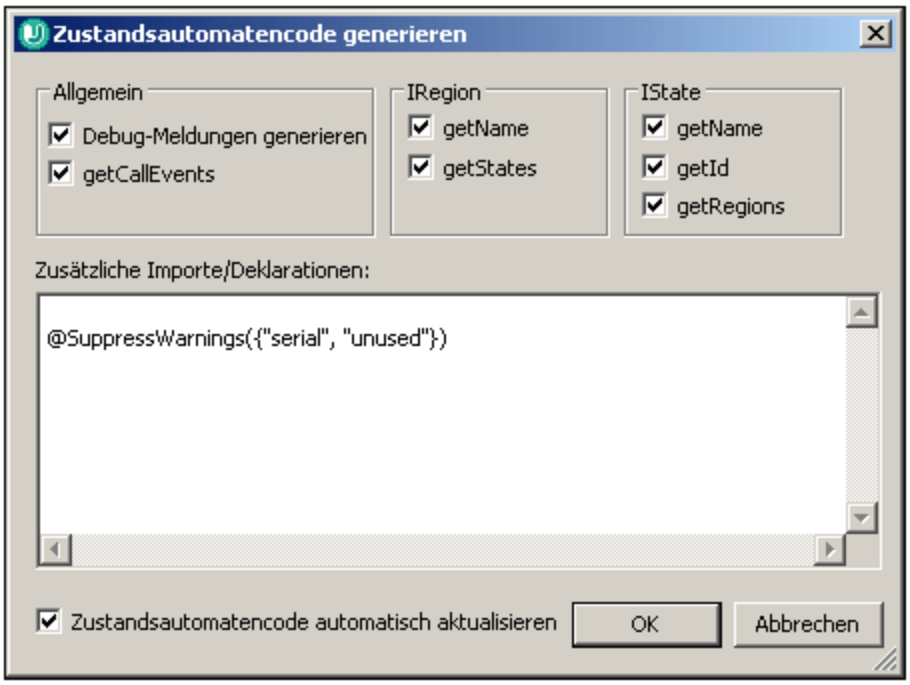
Außerdem enthält jedes Verzeichnis eine Testapplikation, z.B. TestSTMAirCondition.sln für C#, sodass Sie mit den generierten Quellcodedateien sofort arbeiten können.

 © 2016-2022 Altova GmbH



So generieren Sie anhand eines Zustandsdiagramms Code:

- **Klicken Sie mit der rechten Maustaste** in das Zustandsdiagramm und wählen Sie den Befehl "Zustandsautomatencode generieren" oder
- Wählen Sie die Menüoption **Projekt | Zustandsautomatencode generieren.**



In der Abbildung oben sehen Sie die Standardeinstellungen. Klicken Sie zur Codegenerierung auf OK.

Der Zustandsautomatencode wird automatisch aktualisiert, wenn Sie das Forward Engineering starten. Sie können diese Einstellung allerdings ändern, indem Sie auf den Hintergrund des Zustandsdiagramms klicken und das Kontrollkästchen "Code automatisch aktualisieren" deaktivieren.

Am generierten Code sollten keine manuellen Änderungen vorgenommen werden, da diese Änderungen beim Reverse Engineering nicht im Zustandsdiagramm übernommen werden.

Wenn Sie neben dem Feld "Code automatisch aktualisieren" auf das Symbol ... klicken, wird das Dialogfeld "Zustandsautomatencode generieren" geöffnet, wo Sie die Codegenerierungseinstellungen ändern können.

So führen Sie in einem Zustandsdiagramm eine Syntaxüberprüfung durch:

- Klicken Sie mit der rechten Maustaste auf das Diagramm und wählen Sie den Befehl **Zustandsautomatensyntax überprüfen**.

8.1.2.5 Arbeiten mit Zustandsdiagrammcode

Die übergeordnete Klasse des Zustandsdiagramms (d.h. die "Controller-Klasse" oder "Kontextklasse") bildet die einzige Schnittstelle zwischen dem Benutzer des Zustandsdiagramms und der Zustandsdiagrammimplementierung.

Die Controller-Klasse bietet Methoden, mit Hilfe derer die Zustandsdiagramme von "außen" her geändert werden können (z.B. nachdem externe Ereignisse eingetreten sind).

Bei der Implementierung des Zustandsdiagramms werden dagegen Controller-Klassenmethoden ("Callbacks") aufgerufen, um den Benutzer des Zustandsdiagramms über Zustandsänderungen (OnEntry, OnExit, ...), Transitionseffekte und die Möglichkeit Methoden für Bedingungen (Guards) außer Kraft zu setzen und zu implementieren, zu informieren.

UModel kann einfache Operationen (ohne Parameter) für Entry/Exit/Do-Verhalten, Transitionseffekte, ...automatisch erstellen, wenn die entsprechende Option aktiviert ist. (siehe auch Erstellen von Zuständen, Aktivitäten und Transitionen) 320. Diese Methoden können in UModel beliebig (durch Hinzufügen von Parametern, Definieren der Parameter als abstrakt, usw.) geändert werden.

 © 2016-2022 Altova GmbH

Ein Zustandsdiagramm (d.h. seine Controller-Klasse) kann mehrmals instantiiert werden. Alle Instanzen sind unabhängig voneinander.

- Die Ausführung des UML-Zustandsdiagramms ist für das Modell: Bis zur Fertigstellungen ausführen" konzipiert.
- UML-Zustandsautomaten gehen davon aus, dass jedes Ereignis abgeschlossen ist, bevor das nächste verarbeitet wird.
- Dies bedeutet auch, dass keine Entry/Exit/Do-Aktion oder kein Transitionseffekt eine neue Transition/eine Zustandsänderung direkt auslösen darf.

Initialisierung

- Jede Region eines Zustandsdiagramms muss einen Anfangszustand haben.
- Der von UModel generierte Code initialisiert alle Regionen des Zustandsdiagramms automatisch (oder bei Aufruf der Initialize()-Methode der Controller-Klasse).
- Wenn OnEntry-Ereignisse bei der Initialisierung nicht erwünscht sind, können Sie die Initialize()-Methode manuell aufrufen und OnEntry-Ereignisse beim Start ignorieren.

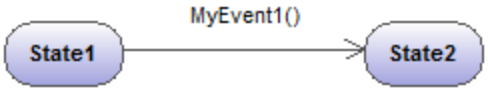
Abrufen des/der aktuellen Zustands/Zustände

UModel unterstützt sowohl zusammengesetzte Zustände als auch orthogonale Zustände, d.h. es gibt nicht nur einen aktuellen Zustand, sondern jede Region (auf jeder hierarchischen Ebene) kann einen aktuellen Zustand haben.

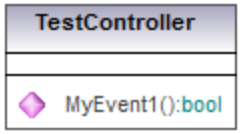
Im Beispiel "AirCondition" wird gezeigt, wie die Regionen des aktuellen Zustands / der aktuellen Zustände durchlaufen werden.

```
TreeNode rootNode = m_CurrentStateTree.Nodes.Add(m_STM.getRootState().getName());
UpdateCurrentStateTree(m_STM.getRootState(), rootNode);

private void UpdateCurrentStateTree(AirCondition.AirConditionController.IState state,
TreeNode node)
{
    foreach (AirCondition.AirConditionController.IRegion r in state.getRegions())
    {
        TreeNode childNode = node.Nodes.Add(r.getName() + " : " + r.getCurrentState().getName());
        UpdateCurrentStateTree(r.getCurrentState(), childNode);
    }
}
```

Beispiel 1 - eine einfache Transition



Die entsprechende Operation wird in UModel automatisch generiert.



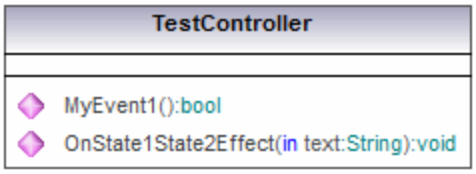
Generierte Methode im Code:

```
private class CTestStateMachine : IState
{
        …
        public bool MyEvent1()
        {
                …
        }
}
```

- Der Benutzer des Zustandsdiagramms sollte die generierte Methode "MyEvent1" aufrufen, wenn das entsprechende Ereignis (außerhalb des Zustandsautomaten) eintritt.
- Der Rückgabeparameter dieser Ereignismethoden liefert Informationen, ob das Ereignis eine Zustandsänderung verursacht hat (d.h. ob es eine Auswirkung auf das Zustandsdiagramm hat) oder nicht. Wenn sich der Automat z.B. im "State1" befindet und "MyEvent1()" eintritt, so ändert sich der aktuelle Status in "State2" und "MyEvent1()" gibt "true" zurück. Wenn "State2" aktiv ist und "MyEvent1()" eintritt, ändert sich nichts am Zustandsdiagramm und MyEvent1() gibt "false" zurück.

Beispiel 2 - eine einfache Transition mit einem Effekt

Die entsprechende Operation wird in UModel automatisch generiert.



Generierte Methode im Code:

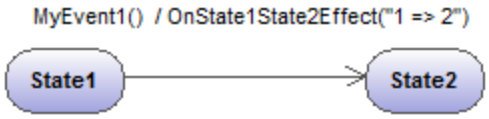
```
private class CTestStateMachine : IState
{
        …
        // Override to handle entry/exit/do actions, transition effects,...:
        public virtual void OnState1State2Effect() {}
}
```

 © 2016-2022 Altova GmbH

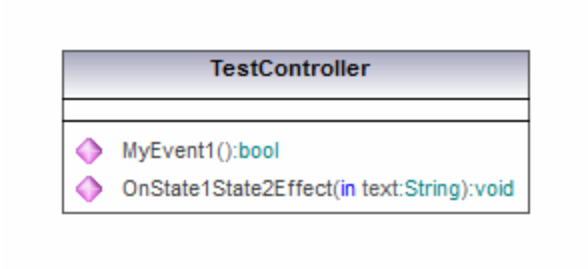
- "OnState1State2Effect()" wird immer dann von der Zustandsdiagrammimplementierung aufgerufen, wenn die Transition zwischen "State1" und "State2" ausgelöst wird.
- Als Reaktion auf diesen Effekt sollte "OnState1State2Effect()" in einer abgeleiteten Klasse von "CTestStateMachine" außer Kraft gesetzt werden.
- "CTestStateMachine:: OnState1State2Effect()" kann auch auf "abstract" gesetzt werden und Sie erhalten Kompilierfehler bis die Methode außer Kraft gesetzt wird.
- Wenn "OnState1State2Effect()" nicht "abstract" ist und die Option "Debug-Meldungen generieren" aktiv ist, generiert UModel die folgende Debug-Ausgabe:

```
// Override to handle entry/exit/do actions, transition effects,...:
public virtual void OnState1State2Effect() {OnDebugMessage("ACTION:
OnState1State2Effect");}
```

Beispiel 3 - eine einfache Transition mit einem Effekt-Parameter



Die entsprechende Operation wird in UModel automatisch generiert.

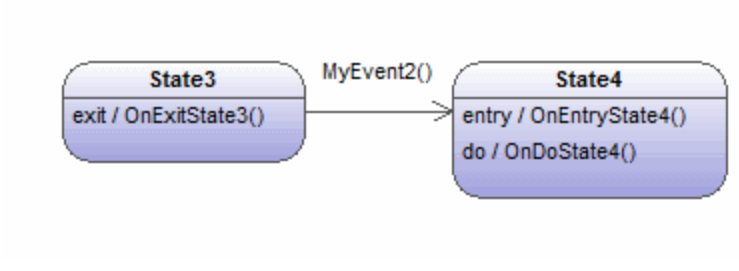


Generierte Methode im Code:

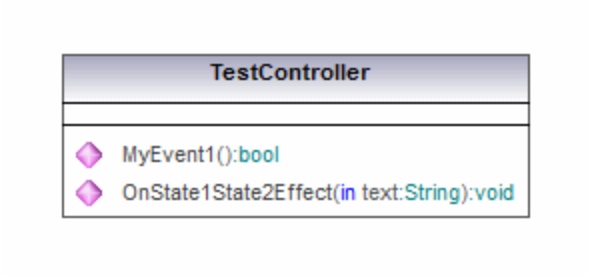
```
private class CTestStateMachine : IState
{
        …
        // Additional defined operations of the controller class:
        public virtual void OnState1State2Effect(String text)
        {
        }
}
```

- Zur Durchführung von (automatisch mit UModel erzeugten) Operationen können Parameter manuell hinzugefügt werden (UModel kann nicht wissen, welcher Typ benötigt wird).
- In diesem Beispiel wurde der Parameter "text:String" zur Effekt-Methode in TestController hinzugefügt. Beim Aufruf dieser Methode muss ein ordnungsgemäßes Argument definiert werden (hier: "1 => 2").
- Eine andere Möglichkeit wäre z.B. die folgende: Aufruf von statischen Methoden ("MyStatic.OnState1State2Effect("1 => 2")") oder Methoden von Singletons ("getSingleton().OnState1State2Effect("1 => 2")").

Beispiel 4 - entry/exit/do-Aktionen



Die entsprechenden Operationen werden in UModel automatisch generiert.



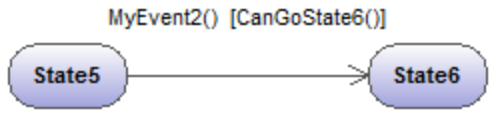
Generierte Methode im Code:

```
private class CTestStateMachine : IState
{
        …
        // Override to handle entry/exit/do actions, transition effects,...:
        public virtual void OnExitState3() {}
        public virtual void OnEntryState4() {}
        public virtual void OnDoState4() {}
}
```

- Zustände können entry/exit/do-Verhalten aufweisen. UModel generiert automatisch die entsprechenden Operationen zu deren Behandlung.
- Wenn im Beispiel oben "MyEvent2()" eintritt, ruft die Zustandsdiagrammimplementierung "OnExitState3()" auf. Wenn "MyEvent2" einen Effekt hätte, würde dieser in der Folge aufgerufen werden. Anschließend würden "OnEntryState4" und "OnDoState4" aufgerufen werden.
- Normalerweise sollten diese Methoden außer Kraft gesetzt werden. Wenn sie nicht abstrakt sind und die Option "Debug-Meldungen generieren" aktiv ist, liefert UModel eine Standard-Debug-Ausgabe, wie in Beispiel 2 beschrieben.
- Diese Methoden können auch Parameter haben, wie in Beispiel 3 gezeigt.

Beispiel 5 - Guards

Transitionen können Guards (Wächterausdrücke) haben, die ermitteln, ob die Transition wirklich ausgelöst werden kann.



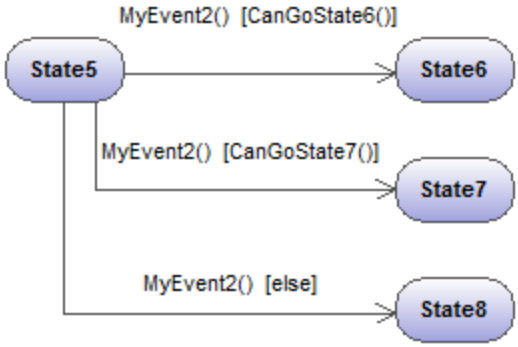
 © 2016-2022 Altova GmbH

Die entsprechende Operation wird in UModel automatisch generiert.

Generierte Methode im Code:

```
private class CTestStateMachine : IState
{
        …
        // Additional defined operations of the controller class:
        public virtual bool CanGoState6()
        {
            return true; // Override!
        }
}
```

- Wenn "State5" der aktive Zustand ist und "MyEvent2" eintritt, so ruft die Zustandsdiagrammimplementierung "CanGoState6" auf und je nach Ergebnis wird die Transition ausgelöst oder nicht.
- Normalerweise sollten diese Methoden außer Kraft gesetzt werden. Wenn sie nicht abstrakt sind und die Option "Debug-Meldungen generieren" aktiv ist, liefert UModel eine Standard-Debug-Ausgabe, wie in Beispiel 2 beschrieben.
- Diese Methoden können auch Parameter haben, wie in Beispiel 3 gezeigt.
- Es sind mehrere Transitionen mit demselben Ereignis aber unterschiedlichen Guards möglich. Die Reihenfolge, in der die verschiedenen Guards abgefragt werden, ist nicht definiert. Wenn eine Transition keinen Guard hat oder der Guard "else" ist, wird sie als der letzte Guard betrachtet (d.h. dieser Guard wird nur dann ausgelöst, wenn alle anderen Transitions-Guards "false" zurückgeben). So ist z.B. im Diagramm unten nicht definiert, ob `CanGoState6()` oder `CanGoState7()` zuerst aufgerufen wird. Die dritte Transition wird nur ausgelöst, wenn `CanGoState6()` und `CanGoState7()` false zurückgeben.



Weitere Konstrukte und Funktionalitäten finden Sie in den Beispielen **AirCondition.ump** und **Complex.ump**.

8.1.2.6 Zustandsdiagramm-Elemente

Startzustand (Pseudozustand)
Der Beginn eines Prozesses.

Endzustand
Das Ende der Abfolge von Prozessen.

Eintrittspunkt (Pseudozustand)
Der Eintrittspunkt eines Zustandsautomaten oder eines zusammengesetzten Zustands.

Austrittspunkt (Pseudozustand)
Der Austrittspunkt eines Zustandsautomaten oder eines zusammengesetzten Zustands.

Entscheidung
Dieses Element stellt eine dynamische bedingte Verzweigung dar, wobei einander gegenseitig ausschließende Guard Trigger ausgewertet werden (OR Operation).

Kreuzung (Pseudozustand)
Dieses Element stellt das Ende der OR-Operation dar, die durch das Element "Entscheidung" definiert ist.

Beendung (Pseudozustand)
Das Anhalten der Ausführung des Zustandsautomaten.

Gabelung (Pseudozustand)
Fügt eine vertikale Gabelungsleiste ein.
Dient zum Aufteilen von Sequenzen in nebenläufige Untersequenzen.

Gabelung horizontal (Pseudozustand)
Fügt eine horizontale Gabelungsleiste ein.
Dient zum Aufteilen von Sequenzen in nebenläufige Untersequenzen.

Vereinigung (Pseudozustand)
Vereinigt/führt zuvor definierte Untersequenzen zusammen. Alle Aktivitäten müssen abgeschlossen sein, bevor Sie fortfahren können.

© 2016-2022 Altova GmbH

Vereinigung, horizontal (Pseudozustand)

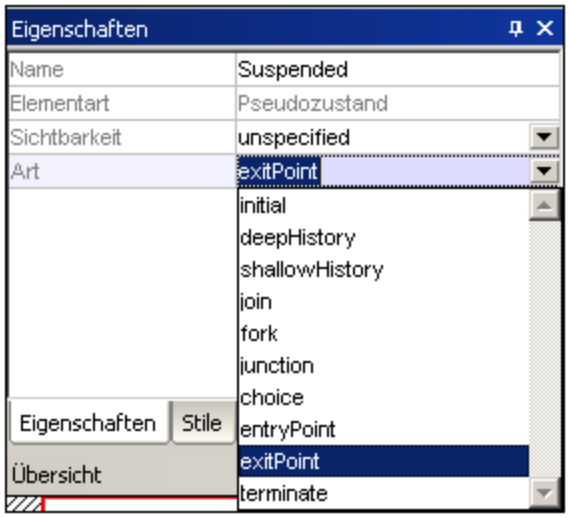
Vereinigt/führt zuvor definierte Untersequenzen zusammen. Alle Aktivitäten müssen abgeschlossen sein, bevor Sie fortfahren können.

DeepHistory

Ein Pseudozustand, der den zuvor aktiven Zustand in einem zusammengesetzten Zustand wiederherstellt.

ShallowHistory

Ein Pseudozustand, der den Ausgangszustand eines zusammengesetzten Zustands wiederherstellt. Alle Pseudozustandselemente können in einen anderen "Typ" geändert werden, indem Sie auf dem Register "Eigenschaften" den Eintrag in der Auswahlliste "**Art**" ändern.



Verbindungspunktreferenz

Eine Verbindungspunktreferenz stellt eine Verwendung (als Teil eines Unterautomatenzustands) eines Eintritts-/Austrittspunkts dar, der in der Zustandsautomatenreferenz durch den Unterautomatenzustand definiert ist.

So fügen Sie Eintritts- oder Austrittspunkte zu einer Verbindungspunktreferenz hinzu:

- Der Zustand, mit dem der Punkt verbunden ist, muss selbst einen Unterautomatenzustand referenzieren. (sichtbar auf dem Register "Eigenschaften").
- Dieser Unterautomat muss einen oder mehrere Eintritts- oder Austrittspunkte enthalten.

Transition

Eine direkte Beziehung zwischen zwei Zuständen. Ein Objekt im ersten Zustand führt eine oder mehrere Aktionen durch und tritt anschließend abhängig von einem Ereignis und der Erfüllung etwaiger Guard-

Bedingungen in den zweiten Zustand ein. Transitionen haben einen Event-Trigger, (eine) Guard-Bedingung(en), eine Aktion (Verhalten) und einen Zielzustand. Die folgenden Ereignis-Unterelemente werden unterstützt:

- ReceiveSignalEvent
- SignalEvent
- SendSignalEvent
- ReceiveOperationEvent
- SendOperationEvent
- ChangeEvent.

Automatische Erstellung von Operationen in der Zielkomponente durch Eingabe von Operationsnamen ein-/ausschalten

Wenn Sie die Schaltfläche "Automatische Erstellung von Operationen in der Zielkomponente durch Eingabe von Operationsnamen ein-/ausschalten" aktivieren, wird die entsprechende Operation in der referenzierten Klasse automatisch erstellt, wenn Sie eine Transition erstellen und einen Namen myOperation() eingeben.

Anmerkung: Operationen können nur dann automatisch erstellt werden, wenn sich der Zustandsautomat innerhalb einer Klasse oder Schnittstelle befindet.

8.1.3 Protokoll-Zustandsautomat

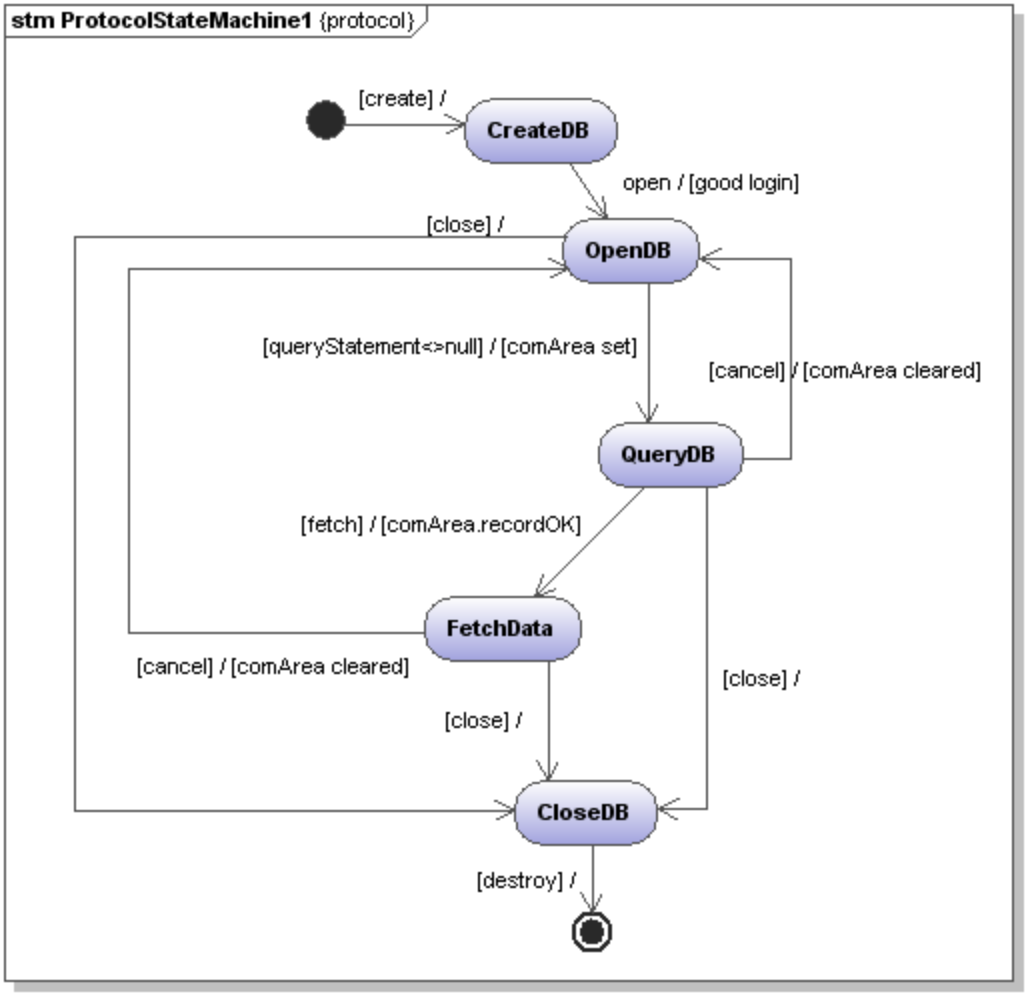
Altova Website: UML-Protokoll-Zustandsdiagramme

Mit Hilfe von Protokoll-Zustandsautomaten wird eine **Sequenz** von Ereignissen dargestellt, auf die ein Objekt reagiert, ohne dass das spezifische Verhalten dargestellt werden muss. In diesem Diagramm werden die benötigte Ereignissequenz und die resultierenden Änderungen am Zustand des Objekts modelliert.

Meist dienen Protokoll-Zustandsautomaten zur Beschreibung komplexer Protokolle, z.B. zur Beschreibung des Datenbankzugriffs über eine bestimmte Schnittstelle oder von Kommunikationsprotokollen wie TCP/IP.

Protokoll-Zustandsautomaten werden auf dieselbe Weise wie Zustandsdiagramme erstellt, haben aber weniger Modellierungselemente. Die Protokoll-Übergänge zwischen Zuständen können Vor- und Nachbedingungen haben, die definieren, was zutreffen, also "true" sein muss, damit der Übergang in einen anderen Zustand erfolgen kann, oder was der resultierende Zustand nach dem Übergang sein muss.

© 2016-2022 Altova GmbH



8.1.3.1 Einfügen von Protokoll-Zustandsdiagramm-Elementen

Einfügen über die Symbole der Symbolleiste:

1. Klicken Sie in der Symbolleiste auf das Symbol "Protokoll-Zustandsautomat".
2. Klicken Sie in das Protokoll-Zustandsdiagramm, um das Element einzufügen. Um mehrere Elemente des ausgewählten Typs einzufügen, halten Sie die **Strg-Taste** gedrückt und klicken Sie in das Diagrammfenster.

Ziehen von bestehenden Elementen in das Protokoll-Zustandsdiagramm:

Die meisten Elemente, die in anderen Protokoll-Zustandsdiagrammen vorkommen, können in ein bestehendes Diagramm eingefügt werden.

1. Suchen Sie das gewünschte Element auf dem Register Modell-Struktur (Sie können dazu das Suchfunktionstextfeld verwenden oder Strg + F drücken, um nach einem beliebigen Element zu suchen).
2. Ziehen Sie das/die Element(e) in das Protokoll-Zustandsdiagramm.

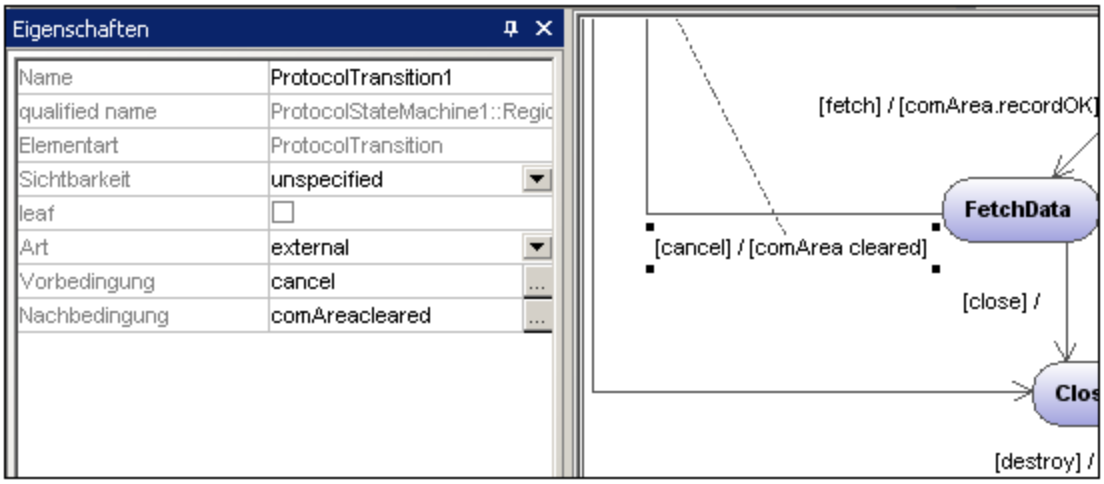
So fügen Sie einen einfachen Zustand ein:

1. Klicken Sie in der Symbolleiste auf das Zustandssymbol und anschließend in das Protokoll-Zustandsdiagramm, um es einzufügen.
2. Geben Sie den Namen des Zustands ein und drücken Sie zur Bestätigung die Eingabetaste. Einfache Zustände haben keine Regionen oder andere Arten von Substrukturen.

So erstellen Sie eine Protokoll-Transition zwischen zwei Zuständen:

1. Klicken Sie auf den Transition-Ziehpunkt des Ausgangszustands (auf der rechten Seite des Elements) oder klicken Sie in der Symbolleiste auf die Schaltfläche "Protocol Transition".
2. Ziehen Sie den Transition-Pfeil mit der Maus auf den Ziel-Zustand.
   Der Textcursor ist automatisch so eingestellt, dass Sie die Vor- und/oder Nachbedingung eingeben können. Verwenden Sie unbedingt die eckigen Klammern [] und den Schrägstrich, wenn Sie die Bedingungen direkt eingeben.

   Wenn Sie die Vor-/Nachbedingung im Fenster "Eigenschaften" eingeben, werden die eckigen Klammern und der Schrägstrich automatisch in das Diagramm eingefügt.



So erstellen Sie Elemente zusammengesetzter Zustände und Unterautomatenzustände bzw. fügen diese ein:

- Siehe Zusammengesetzte Zustände 328

 © 2016-2022 Altova GmbH

8.1.3.2 Protokoll-Zustandsdiagramm-Elemente

Zustand

Ein einfaches Zustandselement mit nur einem Bereich.

Zusammengesetzter Zustand

Diese Art von Zustand enthält einen zweiten Bereich bestehend aus einer einzigen Region. Innerhalb dieser Region können beliebig viele Zustände platziert werden.

Orthogonaler Zustand

Diese Art von Zustand enthält einen zweiten Bereich bestehend aus zwei oder mehreren Regionen, wobei separate Regionen Gleichzeitigkeit kennzeichnen.

Wenn Sie mit der rechten Maustaste auf einen Zustand klicken und den Befehl **Neu | Region** auswählen, können Sie neue Regionen hinzufügen.

Unterautomatenzustand

Mit Hilfe dieses Zustands können Sie die Einzelheiten eines Zustandsautomaten ausblenden. Dieser Zustand hat keine Regionen, ist aber mit einem separaten Zustandsautomaten verknüpft.

Startzustand (Pseudozustand)

Der Beginn eines Prozesses.

Endzustand

Das Ende der Abfolge von Prozessen.

Eintrittspunkt (Pseudozustand)

Der Eintrittspunkt eines Zustandsautomaten oder eines zusammengesetzten Zustands.

Austrittspunkt (Pseudozustand)

Der Austrittspunkt eines Zustandsautomaten oder eines zusammengesetzten Zustands.

Entscheidung

Dieses Element stellt eine dynamische bedingte Verzweigung dar, wobei einander gegenseitig ausschließende Guard Trigger ausgewertet werden (OR Operation).

Kreuzung (Pseudozustand)

Dieses Element stellt das Ende der OR-Operation dar, die durch das Element "Entscheidung" definiert ist.

Beendung (Pseudozustand)

Das Anhalten der Ausführung des Zustandsautomaten.

Gabelung (Pseudozustand)

Fügt eine vertikale Gabelungsleiste ein.
Dient zum Aufteilen von Sequenzen in nebenläufige Untersequenzen.

Gabelung horizontal (Pseudozustand)

Fügt eine horizontale Gabelungsleiste ein.
Dient zum Aufteilen von Sequenzen in nebenläufige Untersequenzen

Vereinigung (Pseudozustand)

Vereinigt/führt zuvor definierte Untersequenzen zusammen. Alle Aktivitäten müssen abgeschlossen sein, bevor Sie fortfahren können.

Vereinigung, horizontal (Pseudozustand)

Vereinigt/führt zuvor definierte Untersequenzen zusammen. Alle Aktivitäten müssen abgeschlossen sein, bevor Sie fortfahren können.

Verbindungspunktreferenz

Eine Verbindungspunktreferenz stellt eine Verwendung (als Teil eines Unterautomatenzustands) eines Eintritts-/Austrittspunkts dar, der in der Zustandsautomatenreferenz durch den Unterautomatenzustand definiert ist.

So fügen Sie Eintritts- oder Austrittspunkte zu einer Verbindungspunktreferenz hinzu:

- Der Zustand, mit dem der Punkt verbunden ist, muss selbst einen Unterautomatenzustand referenzieren. (sichtbar auf dem Register "Eigenschaften").
- Dieser Unterautomat muss einen oder mehrere Eintritts- oder Austrittspunkte enthalten.

Protocol Transition

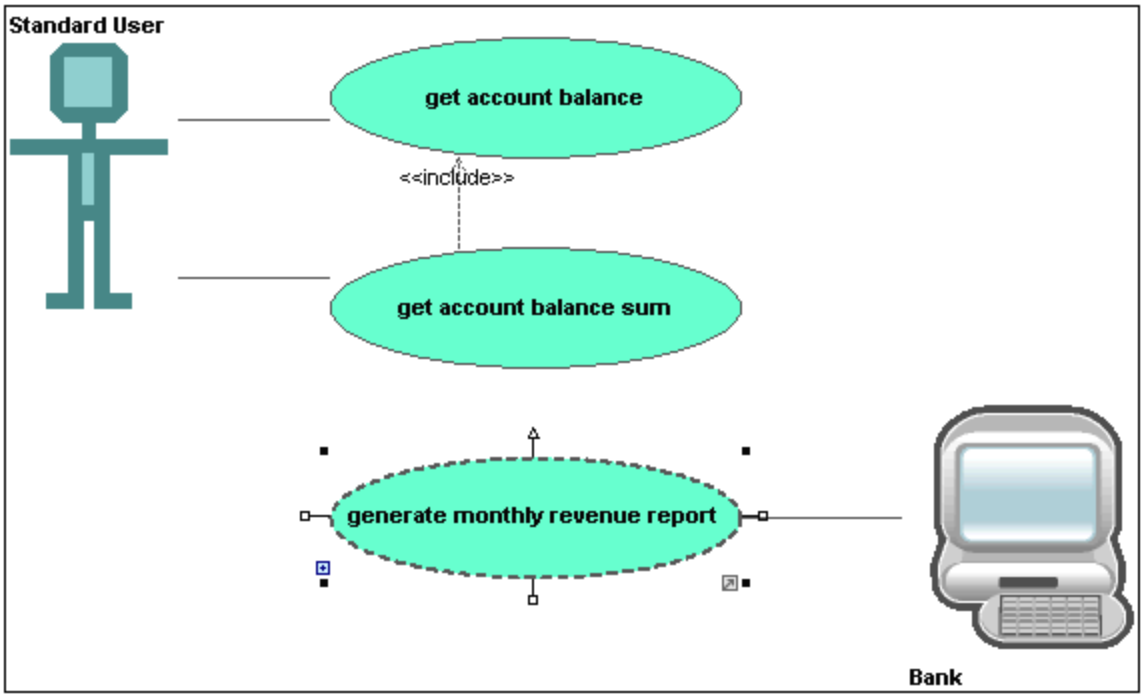
Eine direkte Beziehung zwischen zwei Zuständen. Ein Objekt im ersten Zustand führt eine oder mehrere Aktionen durch und tritt anschließend abhängig von einem Ereignis und der Erfüllung etwaiger Vor- oder Nachbedingungen in den zweiten Zustand ein.

© 2016-2022 Altova GmbH

Nähere Informationen dazu finden Sie unter Einfügen von Protokoll-Zustandselementen 344.

8.1.4 Use Case-Diagramm

Eine Anleitung zum Hinzufügen von Klassen zum Diagramm finden Sie im Tutorial im Abschnitt Use Cases 16.



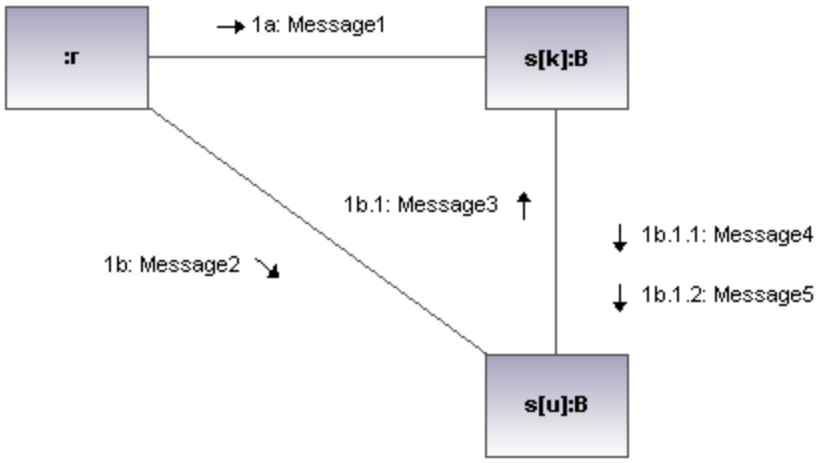
8.1.5 Kommunikationsdiagramm

Altova Website: UML-Kommunikationsdiagramme

In Kommunikationsdiagrammen werden Interaktionen d.h. Nachrichtenflüsse zwischen Objekten zur Laufzeit und die Beziehungen zwischen den miteinander in Wechselbeziehung stehenden Objekten dargestellt. Im Grunde dienen diese Diagramme zum Modellieren des dynamischen Verhaltens von Anwendungsfällen (Use Cases).

Kommunikationsdiagramme weisen denselben Aufbau wie Sequenzdiagramme auf, mit Ausnahme dessen, dass die Notation ein anderes Layout aufweist. Zur Kennzeichnung der Reihenfolge der Nachrichten und der Schachtelung sind die Nachrichten nummeriert.

Sie können in UModel mit einer einzigen einfachen Aktion Kommunikationsdiagramme anhand von Sequenzdiagrammen erstellen und umgekehrt. Nähere Informationen dazu finden Sie unter "Generieren von Sequenzdiagrammen 351".



8.1.5.1 Einfügen von Kommunikationsdiagrammelementen

Verwendung der Symbolleisten-Schaltflächen:

1. Klicken Sie in der Kommunikationsdiagramm-Symbolleiste auf das entsprechende Kommunikationssymbol.

2. Klicken Sie in das Kommunikationsdiagramm, um das Element einzufügen. Um mehrere Elemente des ausgewählten Typs einzufügen, halten Sie die **Strg-Taste** gedrückt und klicken Sie in das Diagrammfenster.

Ziehen vorhandener Elemente in das Kommunikationsdiagramm

Elemente, die in anderen Diagrammen vorkommen, z.B. Klassen, können in ein Kommunikationsdiagramm eingefügt werden.

1. Suchen Sie das gewünschte Element auf dem Register Modell-Struktur (Sie können dazu das Suchfunktionstextfeld verwenden oder Strg + F drücken, um nach einem beliebigen Element zu suchen).
2. Ziehen Sie das/die Element(e) in das Kommunikationsdiagramm.

Lebenslinie

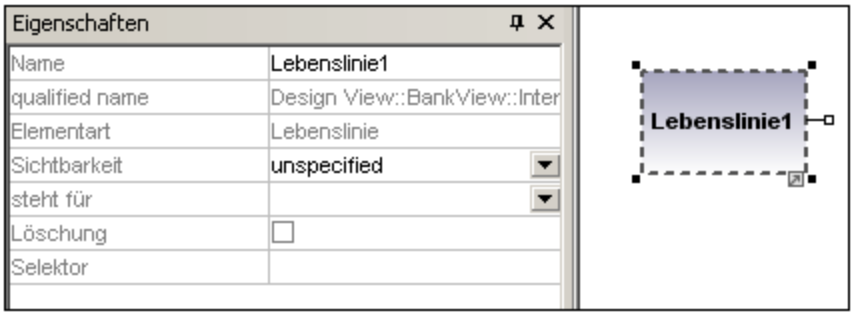
Das Element "Lebenslinie" ist Teil einer Interaktion mehrerer Elemente. Sie haben in UModel die Möglichkeit auch andere Elemente in das Kommunikationsdiagramm einzufügen, z.B. Klassen. Jedes dieser Elemente wird als neue Lebenslinie angezeigt. Sie können die Farben/Schattierung der Lebenslinie über die "Farbverlauf Titel"-Auswahllisten auf dem Register "Stile" neu definieren.

 © 2016-2022 Altova GmbH

Zur Erstellung einer Lebenslinie mit **mehreren Zeilen**, drücken Sie **Strg + Eingabetaste**, um eine neue Zeile zu erstellen.

So fügen Sie eine Kommunikationslebenslinie ein:

1. Klicken Sie in der Titelleiste auf das Symbol "Lebenslinie" und anschließend auf das Kommunikationsdiagramm, um sie einzufügen.



2. Geben Sie einen Namen für die Lebenslinie ein, um den Standardnamen "Lebenslinie1" bei Bedarf zu ändern.

Nachrichten

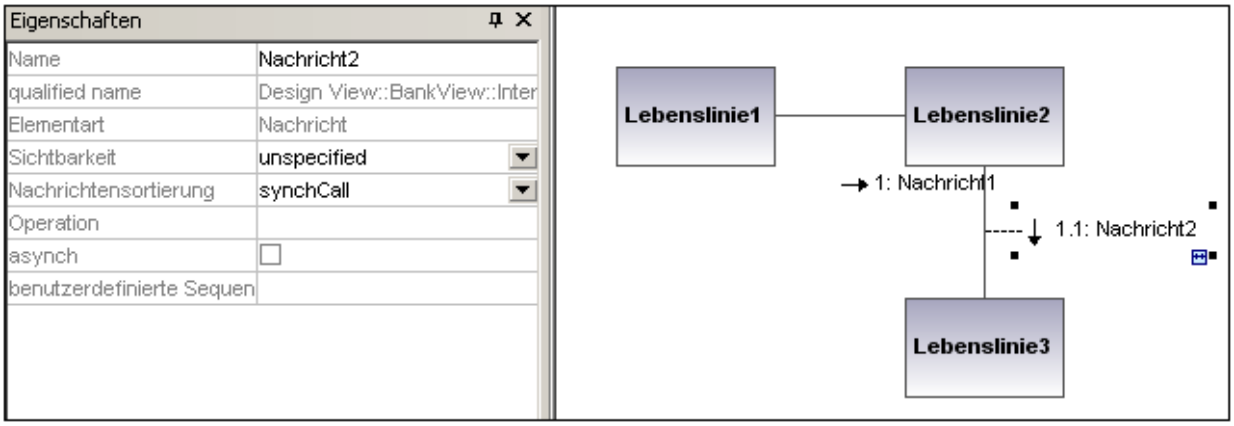
Eine Nachricht ist ein Modellierungselement, das eine bestimmte Art von Kommunikation in einer Interaktion definiert. Bei einer Kommunikation kann es sich z.B. um das Auslösen eines Signals, den Aufruf einer Operation, das Erstellen oder Löschen einer Instanz handeln. Die Nachricht definiert den Absender und Empfänger und um welche Art von Kommunikation es sich handelt.

Nachricht (Aufruf) Nachricht (Antwort) Nachricht (Erstellung) Nachricht (Löschung)

So fügen Sie eine Nachricht ein:

1. Klicken Sie auf das entsprechende Nachrichtensymbol in der Symbolleiste.
2. Ziehen Sie die Nachrichtenlinie auf die empfangenden Objekte.

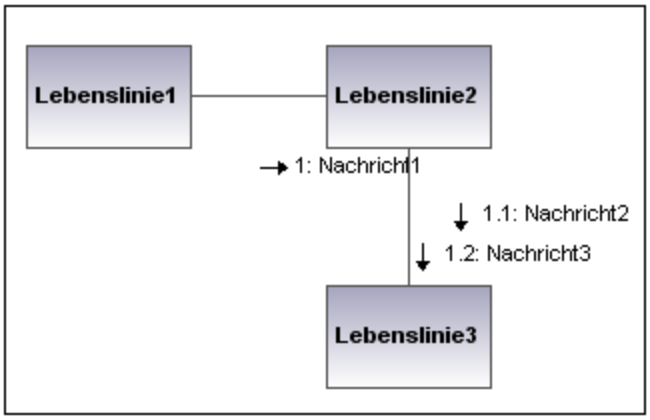
   Die Lebenslinie erscheint markiert, wenn die Nachricht an diese bestimmte Stelle gezogen werden kann.



Anmerkung: Wenn Sie die Strg-Taste gedrückt halten, können Sie mit jedem Klick eine Nachricht einfügen.

So fügen Sie weitere Nachrichten ein:

1. Rechtsklicken Sie auf die vorhandene Kommunikationsverbindung und wählen Sie **Neu | Nachricht**.



- Die Richtung, in die Sie den Pfeil ziehen, bestimmt die Richtung der Nachricht. Antwortnachrichten können in jede der beiden Richtungen weisen.
- Wenn Sie auf ein Nachrichtensymbol klicken und dabei die Strg-Taste gedrückt halten, können Sie mehrere Nachrichten einfügen, indem Sie mehrmals ins Diagramm klicken und die Maus ziehen.

Nachrichtennummerierung

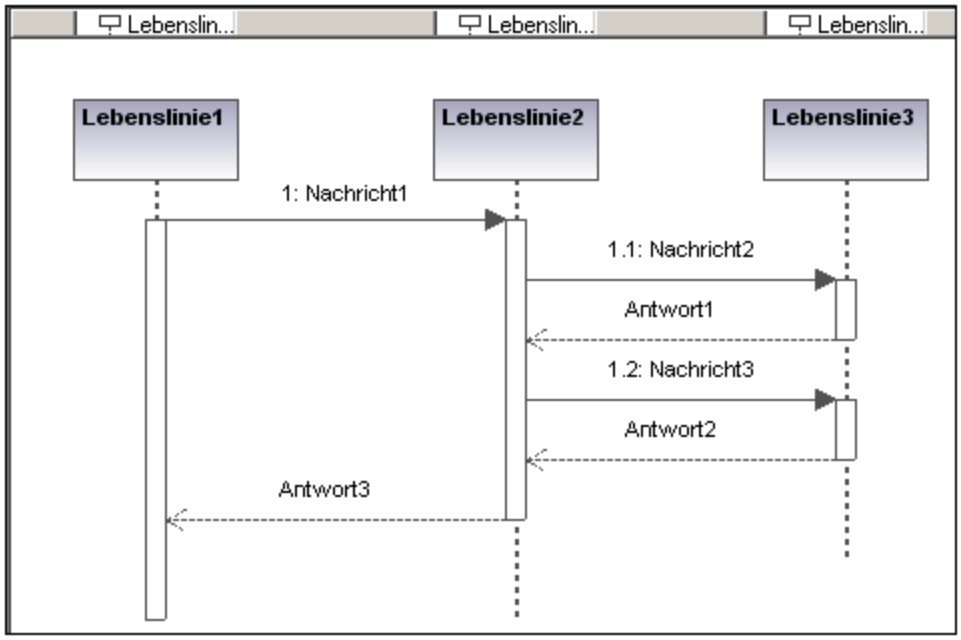
In Kommunikationsdiagrammen wird die Nummerierung in Dezimalschreibweise verwendet, wodurch die hierarchische Struktur der Nachrichten im Diagramm klarer ersichtlich ist. Bei der Reihenfolge handelt es sich um eine durch Punkte getrennte Liste von aufeinander folgenden Zahlen, gefolgt von einem Doppelpunkt und dem Namen der Nachricht.

 © 2016-2022 Altova GmbH

Generieren von Sequenzdiagrammen anhand von Kommunikationsdiagrammen

Sie können in UModel in einer einzigen Aktion Kommunikationsdiagramme anhand von Sequenzdiagrammen generieren und umgekehrt,

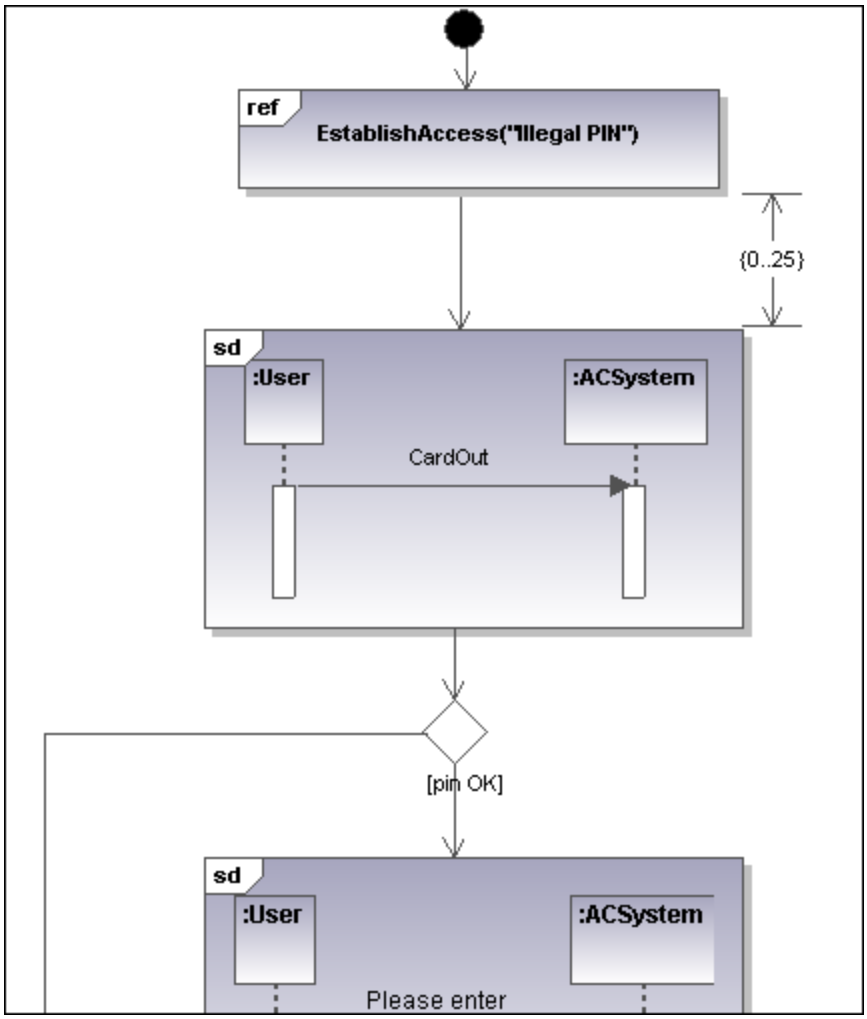
- Rechtsklicken Sie an eine beliebige Stelle in einem Kommunikationsdiagramm und wählen Sie im Kontextmenü den Befehl **Sequenzdiagramm generieren.**



8.1.6 Interaktionsübersichtsdiagramm

Altova Website: UML-Interaktionsübersichtsdiagramme

Interaktionsübersichtsdiagramme sind eine Variante von Aktivitätsdiagrammen und geben Ihnen einen Überblick über die Interaktion zwischen anderen Interaktionsdiagrammen wie z.B. Sequenz-, Aktivitäts, Kommunikations- oder Zeitverlaufsdiagrammen. Interaktionsübersichtsdiagramme werden auf ähnliche Art wie Aktivitätsdiagramme und unter Verwendung derselben Modellierungselemente (Start-/Endpunkte, Gabelungen, Vereinigungen usw.) erstellt.



Anstelle von Aktivitätselementen werden zwei Arten von Interaktionselementen verwendet: Die Elemente "Interaktion" und "Interaktionsverwendung".

Sequenz-, Kommunikations-, Zeitverlaufs- oder Interaktionsübersichtsdiagramme werden als Elemente in Form grafischer Symbole innerhalb eines Rahmens dargestellt, in dessen linker oberer Ecke das Schlüsselwort "SD" angezeigt wird.

Elemente für Interaktionsinstanzen sind Referenzen, die auf vorhandene Interaktionsdiagramme verweisen, wobei im Titelbereich des Rahmens das Schlüsselwort "Ref" und innerhalb des Rahmens der Name der Interaktionsinstanz angezeigt wird.

8.1.6.1 Einfügen von Interaktionsübersichtselementen

Verwendung der Symbolleisten-Schaltflächen:

1. Klicken Sie in der Interaktionsübersichtsdiagramm-Symbolleiste auf das entsprechende Symbol

 © 2016-2022 Altova GmbH

2. Klicken Sie in das Diagramm, um das Element einzufügen. Um mehrere Elemente des ausgewählten Typs einzufügen, halten Sie die **Strg-Taste** gedrückt und klicken Sie in das Diagrammfenster.

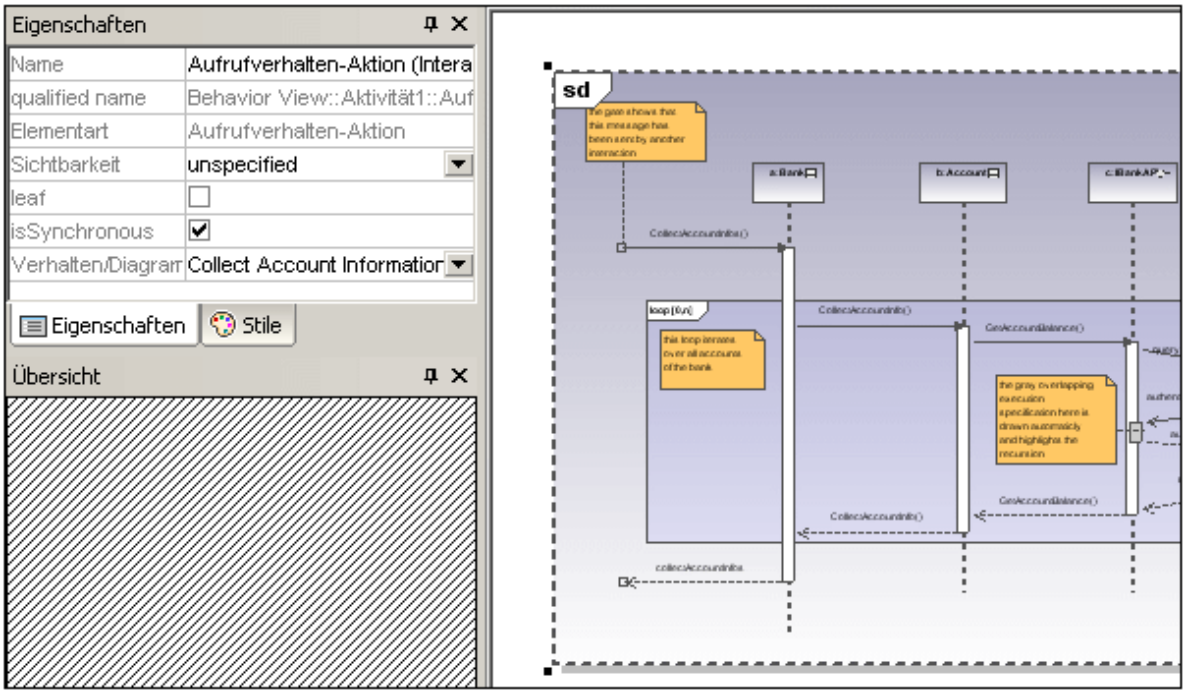
Ziehen vorhandener Elemente in das Interaktionsübersichtsdiagramm

Elemente, die in anderen Diagrammen vorkommen, z.B. Sequenz-, Aktivitäts-, Kommunikations- oder Zeitverlaufsdiagramme, können in ein Interaktionsübersichtsdiagramm eingefügt werden.

1. Suchen Sie das gewünschte Element auf dem Register Modell-Struktur (Sie können dazu das Suchfunktionstextfeld verwenden oder Strg + F drücken, um nach einem beliebigen Element zu suchen).
2. Ziehen Sie das/die Element(e) in das Diagramm.

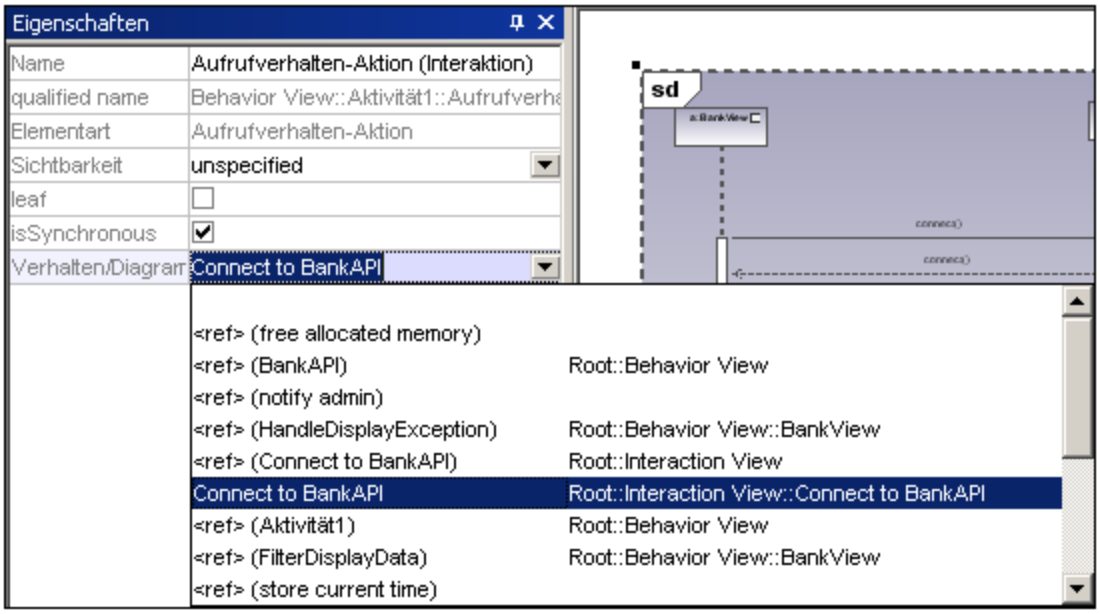
Einfügen eines Interaktionselements

1. Klicken Sie in der Symbolleiste auf das Symbol "Aufrufverhalten-Aktion (Interaktion)" und anschließend in das Interaktionsübersichtsdiagramm, um es einzufügen.

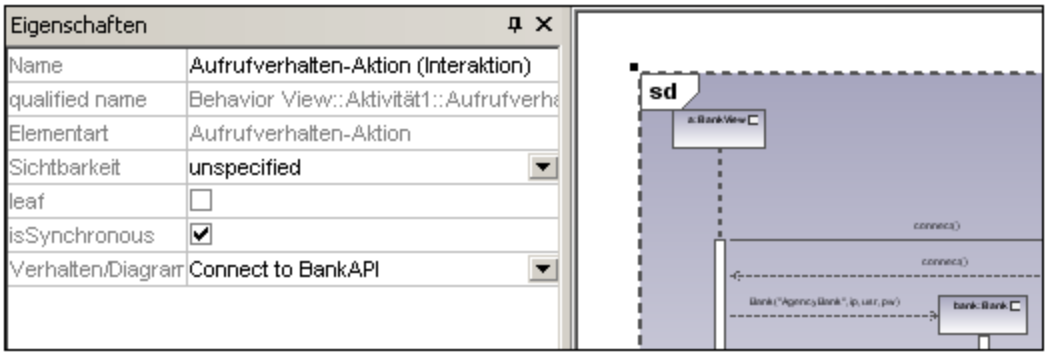


Wenn Sie die Beispieldatei Bank_Multilanguage.ump aus dem Ordner ...\Examples verwenden, wird das "Collect Account Information" Sequenzdiagramm automatisch eingefügt. Standardmäßig wird das erste in der Modell-Struktur gefundene Sequenzdiagramm ausgewählt.

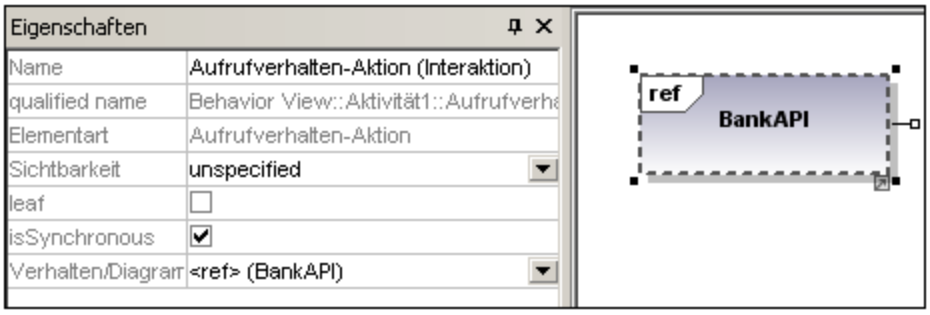
2. Um ein anderes Interaktionselement auszuwählen, klicken Sie auf dem Register "Eigenschaften" auf die Auswahlliste **Verhalten/Diagramm**.
Daraufhin wird eine Liste aller Elemente angezeigt, die eingefügt werden können.



3. Klicken Sie auf das gewünschte Element, z.B. auf Connect to BankAPI.



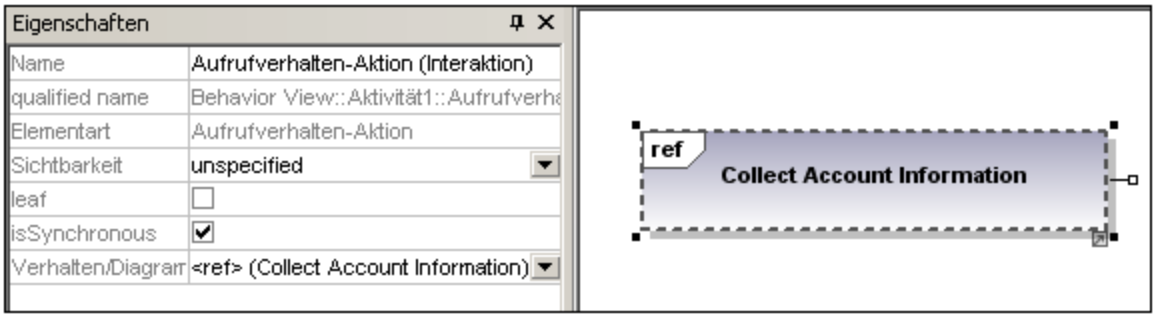
Da es sich dabei auch um ein Sequenzdiagramm handelt, wird das Interaktionselement als Symbol eines Sequenzdiagramms angezeigt.
Bei Auswahl von **<ref> BankAPI** wird die Instanz des Interaktionselements angezeigt.



Einfügen einer Interaktionselementinstanz

1. Klicken Sie in der Symbolleiste auf das Symbol "Aufrufverhalten-Aktion (Interaktionsverwendung) und anschließend in das Interaktionsübersichtsdiagramm, um das Element einzufügen.

   Wenn Sie die Beispieldatei Bank_Multilanguage.ump aus dem Ordner ...**UModelExamples** verwenden, wird das "Collect Account Information" Sequenzdiagramm automatisch als Interaktionselement eingefügt. Standardmäßig wird das erste in der Modell-Struktur gefundene Sequenzdiagramm ausgewählt



2. Um das Interaktionselement zu ändern, doppelklicken Sie auf dem Register "Eigenschaften" auf die Auswahlliste **Verhalten**.
   Daraufhin wird eine Liste aller Elemente angezeigt, die eingefügt werden können.
3. Wählen Sie das gewünschte Element aus.
   Beachten Sie, dass alle mit dieser Methode eingefügten Elemente wie in der Abbildung oben angezeigt werden, d.h. mit "**ref**" im Titelbereich des Rahmens.

Verzweigungsknoten

Fügt einen Verzweigungsknoten mit einer einzigen eingehenden Transition und mehreren mit Guards versehenen ausgehenden Transitionen ein. Nähere Informationen dazu finden Sie unter "Erstellen einer Verzweigung 306".

Verbindungsknoten

Fügt einen Verbindungsknoten ein, der mehrere alternative durch den Verzweigungsknoten definierte Transitionen zusammenführt. Gleichzeitige Prozesse werden dabei von diesem Knoten nicht synchronisiert, sondern es wird einer der Prozesse ausgewählt.

Startknoten

Der Anfang eines Aktivitätsprozesses. Eine Interaktion kann mehrere Startknoten haben.

Aktivitätsendknoten

Das Ende des Interaktionsprozesses. Eine Interaktion kann mehrere Endknoten haben. Alle Flüsse werden gestoppt, wenn der "erste" Endknoten erreicht wird.

Parallelisierungsknoten

Fügt einen vertikalen Parallelisierungsknoten ein.
Dient zum Teilen von Flüssen in mehrere parallele Flüsse.

Parallelisierungsknoten (horizontal)

Fügt einen horizontalen Parallelisierungsknoten ein.
Dient zum Teilen von Flüssen in mehrere parallele Flüsse.

Synchronisationsknoten

Fügt einen vertikalen Synchronisationsknoten ein.
Ein Synchronisationsknoten synchronisiert mehrere durch den Parallelisierungsknoten definierte Flüsse.

Synchronisationsknoten (horizontal)

Fügt einen horizontalen Synchronisationsknoten ein.
Ein Synchronisationsknoten synchronisiert mehrere durch den Parallelisierungsknoten definierte Flüsse.

Zeitdauerbedingung hinzufügen

Eine Zeitdauer definiert eine Wertespezifikation, die eine Zeitdauer zwischen einem Start- und einem Endpunkt angibt. Eine Zeitdauer ist oft ein Ausdruck, der die Anzahl der Uhrticks darstellt, die während dieser Dauer verstreichen.

Kontrollfluss

Ein Kontrollfluss ist eine Kante, d.h. eine mit einem Pfeil versehene Linie, die zwei Verhalten verbindet und eine Interaktion startet, nachdem die vorherige zu Ende geführt wurde.

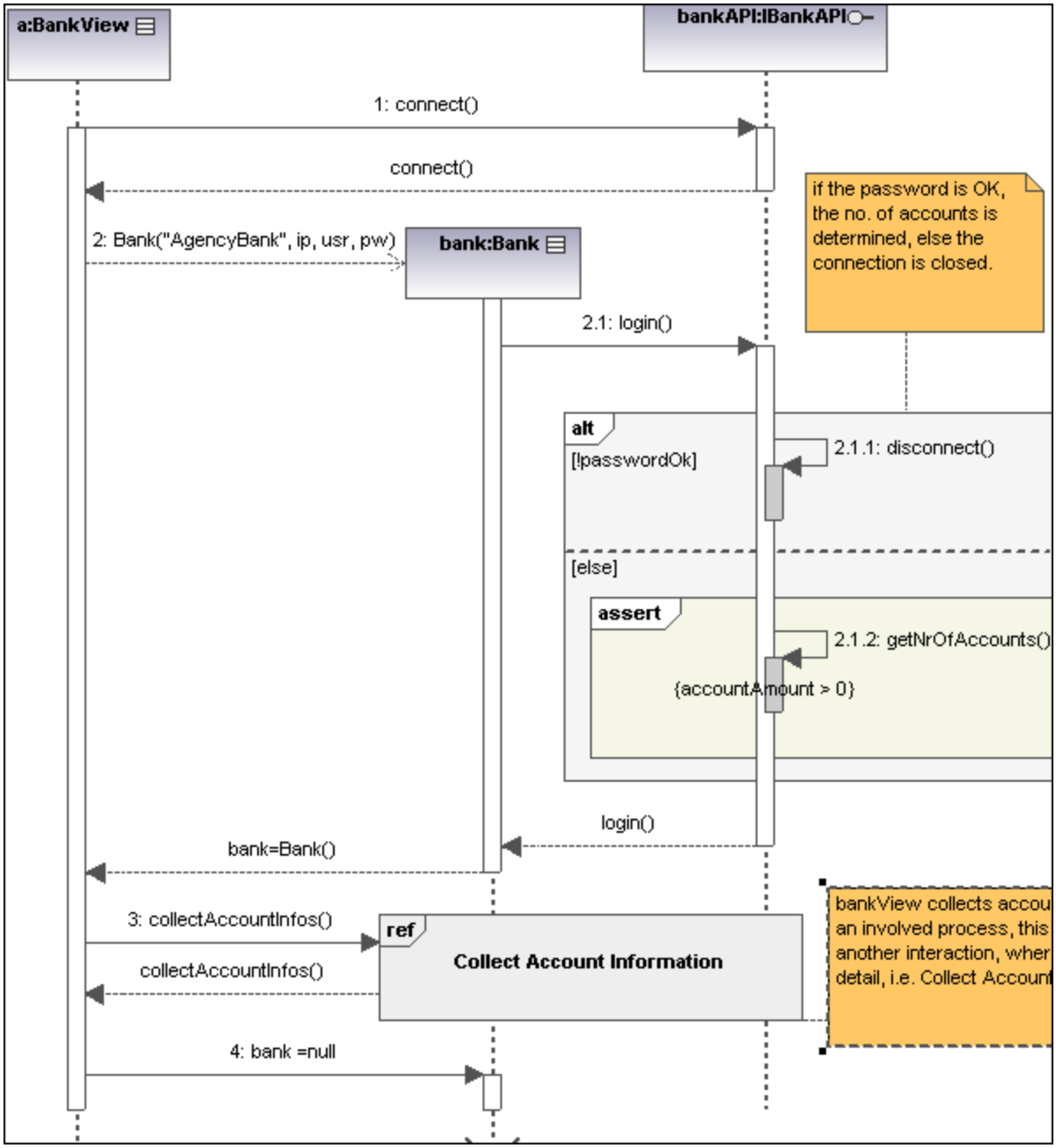
8.1.7 Sequenzdiagramm

Altova Website: UML-Sequenzdiagramme

UModel unterstützt das in UML definierte Standard-Sequenzdiagramm und ermöglicht die einfache Manipulation von Objekten und Nachrichten zur Modellierung von Fallszenarien. Das im folgenden Abschnitt gezeigte Aktivitätsdiagramm steht im UModel-Ordner **...\UModelExamples** in den Beispielen **Bank_Java.ump**, **Bank_CSharp.ump** and **Bank_MultiLanguage.ump** zur Verfügung.

Sie können Sequenzdiagramme manuell modellieren oder diese alternativ dazu anhand von mit Reverse-Engineering erstelltem Quellcode generieren, wie unter Generieren von Sequenzdiagrammen anhand von Quellcode 371 beschrieben.

 © 2016-2022 Altova GmbH



8.1.7.1 Einfügen von Sequenzdiagrammelementen

In einem Sequenzdiagramm werden dynamische Laufzeit-Objektbeziehungen mit Hilfe von Nachrichten modelliert. Sequenzdiagramme dienen im Allgemeinen zur Verdeutlichung einzelner Use Case-Szenarios.

- **Lebenslinien** sind die horizontal angeordneten Kästchen am oberen Rand des Diagramms. Zusammen mit einer strichlierten vertikalen Linie stellen sie das Objekt und seine Interaktionen in

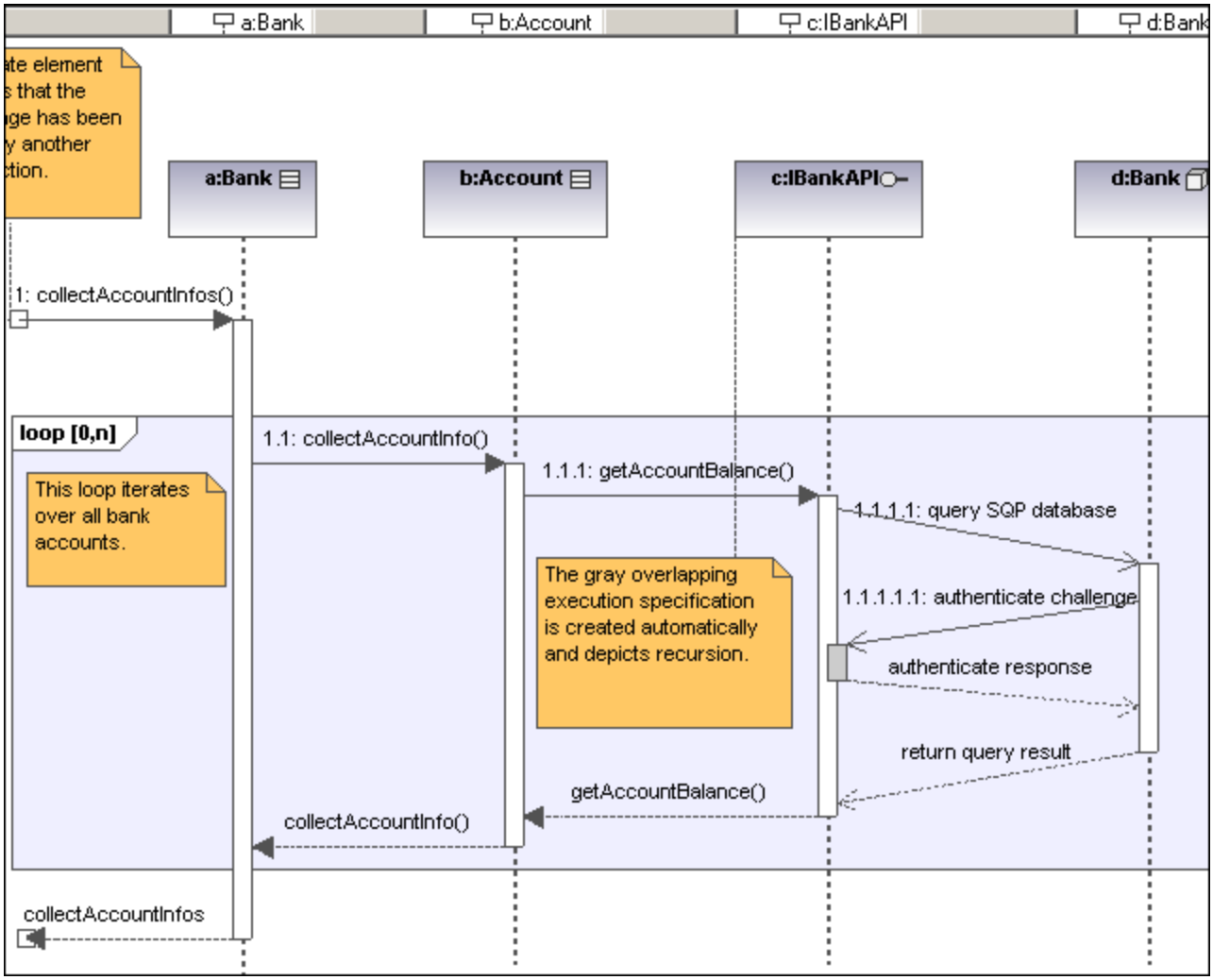
einem zeitlichen Ablauf dar. Nachrichten werden in Form von Pfeilen zwischen den Lebenslinien von zwei oder mehr Objekten dargestellt.

- **Nachrichten** werden zwischen sendenden und empfangenden Objekten gesendet und als Pfeile mit einer Bezeichnung dargestellt. Nachrichten können eine Sequenznummer und verschiedene weitere optionale Attribute haben: argument list usw. Es werden bedingte, optionale und alternative Nachrichten unterstützt. Nähere Informationen dazu finden Sie unter Combined Fragment 361.

Nähere Informationen finden Sie unter den folgenden Themen:

Lebenslinie 359
Combined Fragment 361
Interaction Use 364
Gate 364
Zustandsinvariante 365
Nachrichten 365

Sequenzdiagramm- und andere UModel-Elemente können auf verschiedene Arten in ein Sequenzdiagramm eingefügt werden.



Über die Symbolleistensymbole

1. Klicken Sie in der Sequenzdiagrammsymbolleiste auf das jeweilige Sequenzdiagrammsymbol.
2. Klicken Sie in das Sequenzdiagramm, um das Element einzufügen. Um mehrere Elemente des ausgewählten Typs einzufügen, halten Sie die **Strg-Taste** gedrückt und klicken Sie in das Diagrammfenster.

Ziehen bestehender Elemente in das Sequenzdiagramm

Die meisten Classifier-Arten sowie Elemente, die in anderen Sequenzdiagrammen vorkommen, können in ein bestehendes Sequenzdiagramm eingefügt werden.

1. Suchen Sie das gewünschte Element auf dem Register "Modell-Struktur" (Sie können dazu das Suchfunktionstextfeld verwenden oder Strg + F drücken).
2. Ziehen Sie das Element/die Elemente in das Sequenzdiagramm.

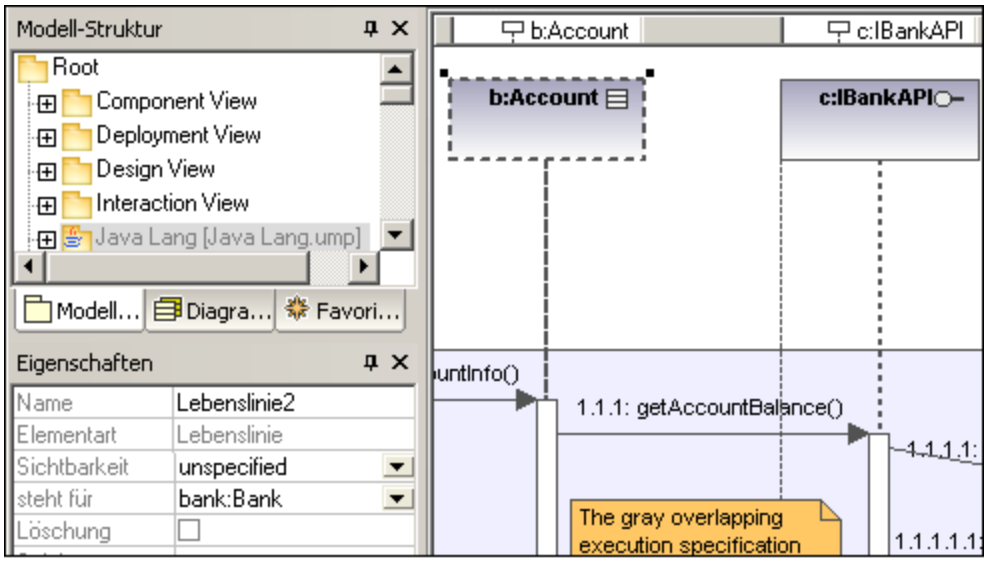
8.1.7.1.1 Lebenslinie

Das Element **Lebenslinie** ist als einzelnes Element an einer Interaktion beteiligt. Sie haben in UModel die Möglichkeit auch andere Elemente in das Sequenzdiagramm einzufügen, z.B. Klassen und Akteure. Jedes dieser Elemente wird als neue Lebenslinie angezeigt, sobald es vom Register "Modell-Struktur" in das Diagrammfenster gezogen wurde.

Die Bezeichnung der Lebenslinie wird in einer Leiste am oberen Rand des Sequenzdiagramms angezeigt. Die Beschriftungen können in der Leiste neu positioniert und in der Größe angepasst werden, wobei die Änderungen sofort im Diagrammfenster zu sehen sind. Auch die Farben/Schattierung der Beschriftung lassen sich über die Farbverlaufsauswahllisten für den Titel auf dem Register "Stile" anpassen.

Zur Erstellung einer Lebenslinie mit **mehreren Zeilen**, drücken Sie **Strg + Eingabetaste**, um eine neue Zeile zu erstellen.

Die meisten Arten von Classifiern können in ein Sequenzdiagramm eingefügt werden. Im Feld "steht für" auf dem Register "Eigenschaften" wird der Elementtyp angezeigt, der als Lebenslinie fungiert. Wenn Sie Eigenschaften, deren Typ definiert ist, in ein Sequenzdiagramm ziehen, wird ebenfalls eine Lebenslinie erstellt.



Ausführungsspezifikation (Objektaktivierung)

Eine Ausführungsspezifikation (Aktivierung) wird in Form eines Kästchens (Rechteck) auf der Objektlebenslinie dargestellt. Eine Aktivierung ist die Ausführung einer Prozedur und die Zeit, die benötigt wird, um etwaige geschachtelte Prozeduren auszuführen. Aktivierungskästchen werden automatisch erstellt, wenn zwischen zwei Lebenslinien eine Nachricht erstellt wird.

Eine rekursive Nachricht oder Selbstnachricht (self message), also eine Nachricht, die eine andere Methode in derselben Klasse aufruft), erstellt gestapelte Aktivierungskästchen.

Ein-/Ausblenden von Aktivierungskästchen:

1. Klicken Sie auf das Register **Stile** und scrollen Sie zum unteren Ende der Liste.
   Über das Auswahlfeld "**Ausführungsspezifikationen anzeigen**" können Sie die Aktivierungskästchen im Sequenzdiagramm ein- und ausblenden.

Lebenslinienattribute

Über das Kontrollkästchen **Löschung** können Sie einen Lösch-Marker oder Stopp zur Lebenslinie hinzufügen, ohne eine Löschnachricht verwenden zu müssen.

Über das Feld **Selektor** können Sie einen Ausdruck eingeben, der den durch die Lebenslinie dargestellten Teil spezifiziert, wenn das Element, das verbunden werden kann, mehrere Werte hat, d.h. eine Multiplizität größer als eins aufweist.

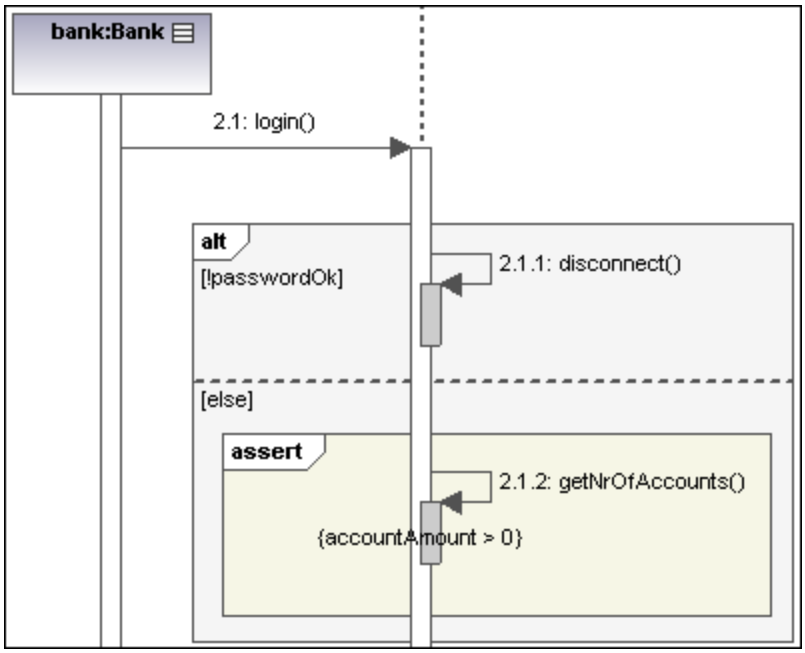
Gehe zu Lebenslinienelement

Wenn Sie mit der rechten Maustaste auf eine Lebenslinie klicken, können Sie die Option "Gehe zu XXX" auswählen, wobei XXX für den Typ der speziellen Lebenslinie steht, auf die Sie geklickt haben. Das Element wird daraufhin im Fenster "Model-Struktur" angezeigt.

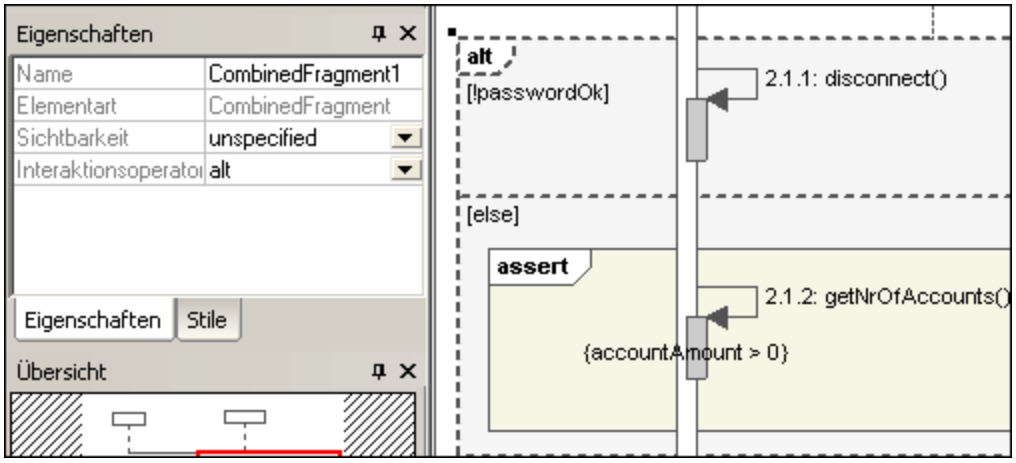
 © 2016-2022 Altova GmbH

8.1.7.1.2 Combined Fragment

Combined Fragments sind Untereinheiten oder Abschnitte einer Interaktion. Der Interaktionsoperator, der im Fünfeck in der linken oberen Ecke angezeigt wird, definiert, um welche spezifische Art von Combined Fragment es sich handelt. Die Einschränkung definiert daher das jeweilige Fragment z.B. loop fragment, alternative fragment usw., welches in der Interaktion verwendet wird.



Über die Symbole für Combined Fragments in der Symbolleiste können Sie bestimmte Combined Fragments einfügen: seq, alt oder loop. Auch durch Klicken auf die **Interaktionsoperator-**Auswahlliste können Sie das spezifische Interaktionsfragment definieren.



Interaktionsoperatoren

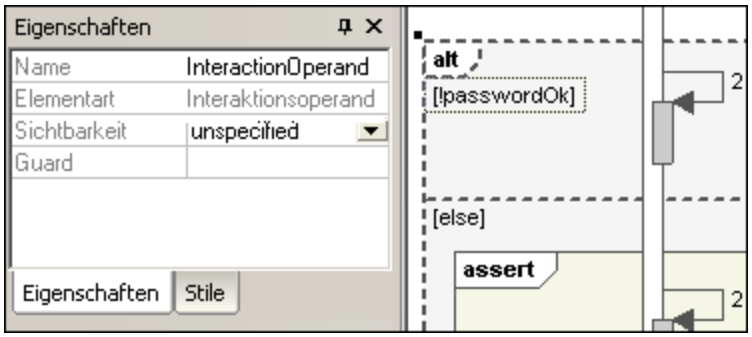
Lose Ordnung	**seq**	seq Das Combined Fragment stellt eine lose Ordnung zwischen dem Verhalten der Operanden dar.
Alternativen	**alt**	Nur einer der definierten Operanden wird ausgewählt. Der Operand muss eine "Guard-Expression" haben, deren Auswertung "true" ergibt. alt [else] Wenn in einem der Operanden der Guard-Ausdruck "else" verwendet wird, wird dieser Operand ausgeführt, wenn alle anderen Guards den Wert "false" zurückgeben. Die Guard-Expression kann unmittelbar beim Einfügen eingegeben werden und wird innerhalb der beiden eckigen Klammern angezeigt. CombinedFragment1 InteractionOperand [] : {!passwordOk} InteractionOperand Model ... Diagra... Favorites Properties Name: InteractionOperand qualified name: Interaction View::Connec Elementart: Interaktionsoperand Sichtbarkeit: unspecified guard: !passwordOk alt [!passwordOk] [else] assert {accountAm Collect Account Inform Bei der **Interaktionsbedingung** handelt es sich eigentlich um den Guard-Ausdruck zwischen den eckigen Klammern.
Option	**opt**	Option steht für eine Auswahl, wobei entweder nur der Operand ausgeführt wird oder nichts passiert.
Abbruchfragment	**break**	Der Break-Operator wird ausgewählt, wenn Guard "true" ist. Der Rest des umschließenden Fragments wird ignoriert.
Parallel	**par**	Zeigt an, dass das Combined Fragment eine parallele Zusammenführung von Operanden darstellt.

© 2016-2022 Altova GmbH

Strenge Ordnung	**strict**	Das Combined Fragment stellt eine strenge Ordnung zwischen dem Verhalten der Operanden dar.
Schleife	**loop**	Der Schleifen-Operand wird so oft wiederholt, wie in der Guard-Expression definiert. **loop [0,n]** Nachdem Sie diesen Operand ausgewählt haben, können Sie den Ausdruck (im Schleifen-Fünfeck) direkt bearbeiten, indem Sie darauf doppelklicken.
Kritischer Bereich	**critical**	Das Combined Fragment stellt einen kritischen Bereich dar. Die Sequenz(en) darf/dürfen nicht durch andere Prozesse unterbrochen werden.
Negativ	**neg**	Definiert, dass das Fragment ungültig ist und dass alle anderen als gültig gelten.
Sicherstellung	**assert**	Kennzeichnet das gültige Combined Fragment und seine Sequenzen. Wird oft in Kombination mit consider- oder ignore-Operanden verwendet.
Ignore	**ignore**	Definiert, welche Nachricht in der Interaktion ignoriert werden soll. Wird oft im Zusammenhang mit "Sicherstellung" oder "Consider"-Operanden verwendet.
Consider	**consider**	Definiert, welche Nachricht in der Interaktion berücksichtigt werden soll.

Hinzufügen von Interaktionsoperanden zu einem Combined Fragment

1. Rechtsklicken Sie auf das Combined Fragment und wählen Sie **Neu | Interaktionsoperand**. Der Textcursor wird automatisch so gesetzt, dass Sie die Guard Condition eingeben können.
2. Geben Sie die Guard Condition für den Interaktionsoperanden ein, z.B. **!passwordOK** und bestätigen Sie mit der Eingabetaste. Mit Strg + Eingabetaste können Sie einen **mehrzeiligen** Interaktionsoperanden erstellen.



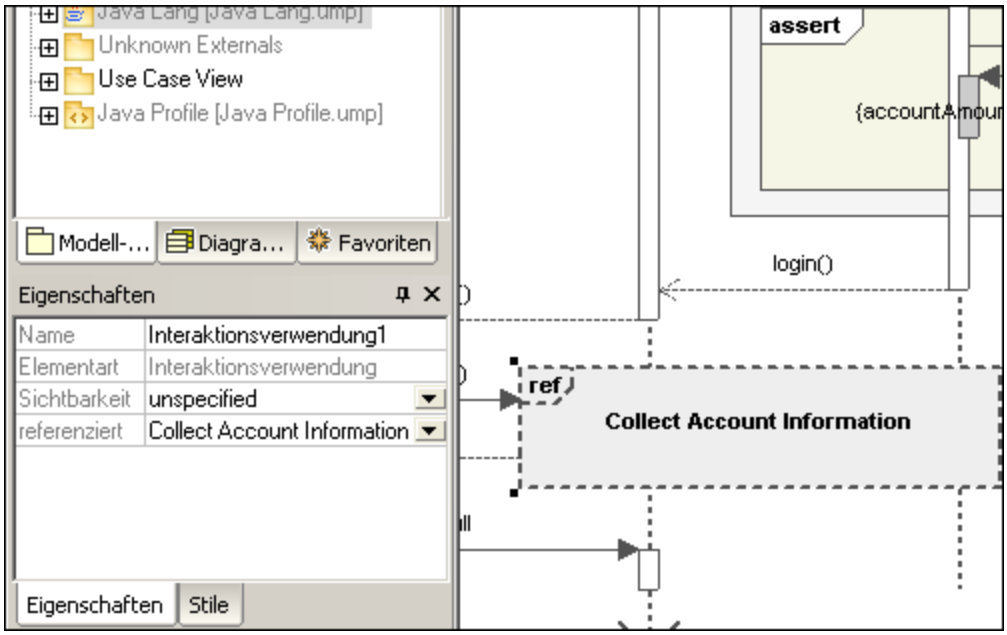
3. Fügen Sie nun auf dieselbe Art den zweiten Interaktionsoperanden mit der Guard Condition "else" ein. Die einzelnen Operanden im Fragment werden durch strichlierte Linien getrennt.

Löschen von Interaktionsoperanden

1. Doppelklicken Sie im Diagramm (nicht auf dem Register "Eigenschaften") auf die Guard-Expression im Combined Fragment.
2. Löschen Sie die Guard-Expression zur Gänze und drücken Sie zur Bestätigung die Eingabetaste.
   Die Guard Expression/der Interaktionsoperand wird entfernt und die Größe des Combined Fragment wird automatisch angepasst.

8.1.7.1.3 Interaction Use

Das Element "Interaction Use" ist eine Referenz auf ein Interaktionselement. Mit Hilfe dieses Elements können Sie Abschnitte einer Interaktion mit verschiedenen anderen Interaktionen gemeinsam verwenden.



Wenn Sie auf die Auswahlliste "referenziert" klicken, können Sie die Interaktion auswählen, auf die Sie referenzieren möchten. Der Name der ausgewählten Interaktion wird im Element angezeigt.

Anmerkung:
Sie können auch ein vorhandenes Interaction Use-Element aus der Modell-Struktur ins Diagrammfenster ziehen.

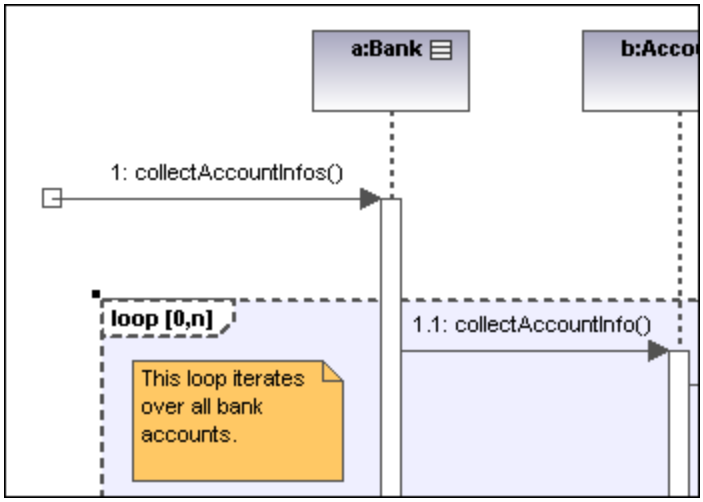
8.1.7.1.4 Gate

Ein **Gate** ist ein Verbindungspunkt, über den Nachrichten in und aus Interaktionsfragmente/n übertragen werden können. Gates werden mittels Nachrichten miteinander verbunden.

1. Fügen Sie das Gate-Element in das Diagramm ein.

© 2016-2022 Altova GmbH

2. Erstellen Sie eine neue Nachricht und ziehen Sie den Cursor vom Gate zur Lebenslinie oder von einer Lebenslinie auf ein Gate.
   Dadurch werden die beiden Elemente miteinander verbunden. Das Quadrat, das das Gate darstellt, wird nun kleiner angezeigt.

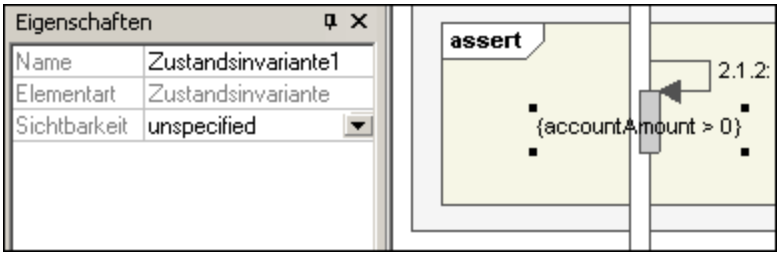


8.1.7.1.5 Zustandsinvariante

Eine **Zustandsinvariante** {s} ist eine Bedingung oder eine Einschränkung, die auf eine Lebenslinie angewendet wird. Die Bedingung muss zutreffen, damit die Lebenslinie vorhanden sein kann.

So definieren Sie eine Zustandsinvariante:

1. Klicken Sie auf das Symbol "Zustandsinvariante" und anschließend auf die Lebenslinie oder auf eine Objektaktivierung, um diese einzufügen.
2. Geben Sie die gewünschte Bedingung/Einschränkung ein, z.B. accountAmount > 0 und drücken Sie zur Bestätigung die Eingabetaste.



8.1.7.1.6 Nachrichten

Nachrichten werden von sendenden an empfangende Lebenslinien gesendet und in Form beschrifteter Pfeile angezeigt. Nachrichten können eine Sequenznummer und verschiedene andere optionale Attribute aufweisen: argument list usw. Nachrichten werden von oben nach unten angezeigt, d.h. die vertikale Linie stellt den zeitlichen Ablauf im Sequenzdiagramm dar.

- Ein **Aufruf** ist eine synchrone oder asynchrone Kommunikation, die eine Operation aufruft, über die die Kontrolle zum sendenden Objekt zurückkehren kann. Ein Aufrufpfeil zeigt zum **Anfang** der Aktivierung, die der Aufruf initiiert.
- Rekursionen oder Aufrufe einer anderen Operation desselben Objekts werden durch Übereinanderstapeln von Aktivierungskästchen angezeigt (Ausführungsspezifikationen).

So fügen Sie eine Nachricht ein:

1. Klicken Sie in der Sequenzdiagramm-Symbolleiste auf das jeweilige Nachrichtensymbol.
2. Klicken Sie auf die Lebenslinie oder das Aktivierungskästchen des Sender-Objekts.
3. Ziehen Sie die Nachrichtenlinie auf die Lebenslinie oder das Aktivierungskästchen des Empfängerobjekts. Objektlebenslinien erscheinen markiert, wenn Sie die Nachricht dorthin ziehen können.

- Die Richtung, in die Sie die Pfeile ziehen, bestimmt die Richtung der Nachricht. Antwortnachrichten können in jede der beiden Richtungen weisen.

- Ein oder mehrere Aktivierungskästchen wird bzw. werden automatisch auf den Sender-/Empfänger-Objekten erstellt oder in der Größe angepasst. Sie können deren Größe auch manuell durch Ziehen der entsprechenden Ziehpunkte anpassen.

- Abhängig davon, welche Einstellungen Sie für die Nummerierung der Nachrichten aktiviert haben, wird die Nummerierungsreihenfolge aktualisiert.

- Wenn Sie auf ein Nachrichtensymbol klicken und dabei die Strg-Taste gedrückt halten, können Sie mehrere Nachrichten einfügen. Ziehen Sie dazu einfach mehrere Nachrichten in das Diagrammfenster.

So löschen Sie eine Nachricht:

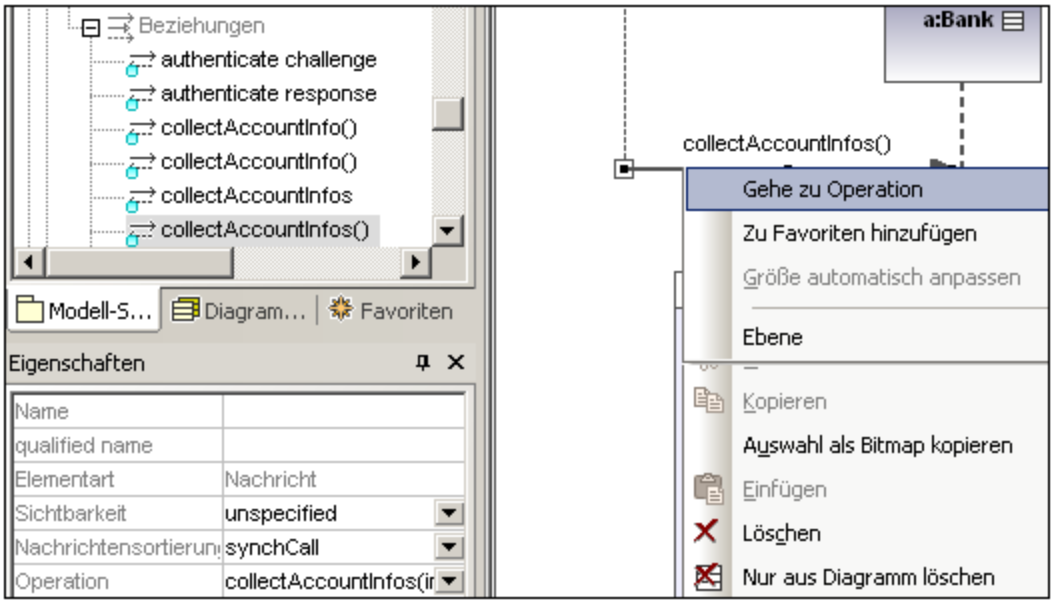
1. Klicken Sie auf die jeweilige Nachricht, um sie auszuwählen.
2. Drücken Sie die Entf-Taste, um die Nachricht aus dem Modell zu löschen oder rechtsklicken Sie auf die Nachricht und wählen Sie den Befehl "Aus Diagramm löschen".
   Die Nummerierung der Nachrichten und die Aktivierungskästchen der verbleibenden Objekte werden aktualisiert.

"Gehe zu-Operation" für Call-Nachrichten

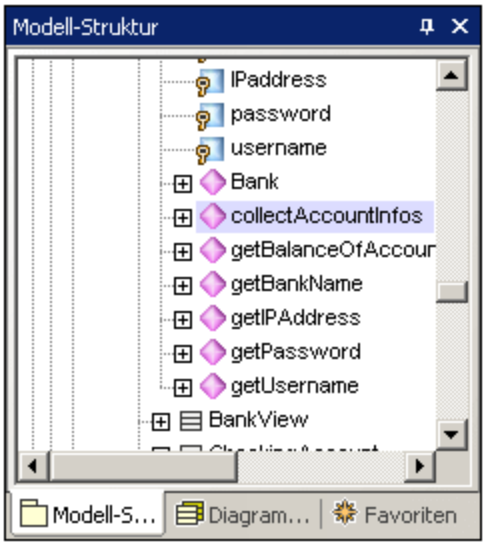
Die durch Call-Nachrichten referenzierten Operationen werden in Sequenz- und Kommunikationsdiagrammen verwendet.

1. Rechtsklicken Sie auf eine Call-Nachricht und wählen Sie den Option "Gehe zu Operation".

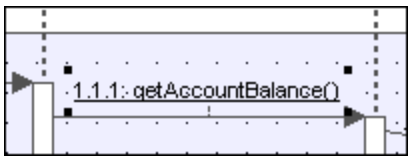
© 2016-2022 Altova GmbH



Die Anzeige ändert sich und die die Verbindungsoperation wird auf dem Register "Modell-Struktur" angezeigt.



Anmerkung:
Statische Operationsnamen werden in Sequenzdiagrammen unterstrichen angezeigt.



So positionieren Sie zusammengehörige Nachrichten:

1. Klicken Sie auf die entsprechende Nachricht und ziehen Sie sie in vertikaler Richtung, um sie neu zu positionieren. Standardmäßig werden beim Neupositionieren einer Nachricht alle mit der aktiven in Zusammenhang stehenden Nachrichten mitverschoben.

   Mit Strg + Klick können Sie mehrere Nachrichten auswählen.

So positionieren Sie Nachrichten einzeln:

1. Klicken Sie auf das Symbol "**Verschieben zusammengehöriger Nachrichten ein/aus**", um diese Funktion zu deaktivieren.
2. Klicken Sie auf die gewünschte Nachricht und ziehen Sie sie, um sie zu verschieben.
   Nur die ausgewählte Nachricht wird beim Ziehen verschoben. Sie können die Nachricht an einer beliebigen Stelle auf der vertikalen Achse zwischen den Objektlebenslinien positionieren.

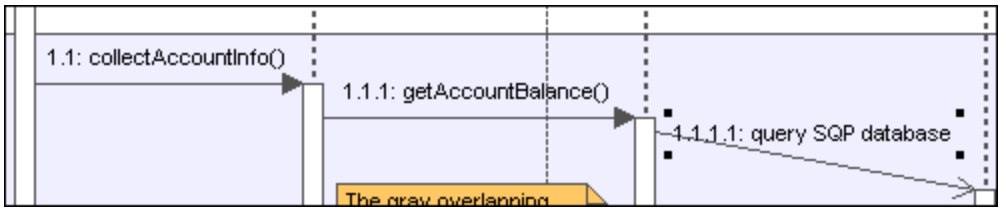
So erstellen Sie automatisch Antwortnachrichten:

1. Klicken Sie auf die Schaltfläche "**Automatische Erstellung von Antworten auf Nachrichten (Call) ein/aus**".
2. Erstellen Sie eine neue Nachricht zwischen den beiden Lebenslinien. Daraufhin wird automatisch eine Antwortnachricht eingefügt.

Nachrichtennummerierung

UModel unterstützt verschiedene Methoden der Nachrichtennummerierung: hierarchisch, einfach und keine.

- **Keine** entfernt alle Nachrichtennummern.
- **Einfach** nummeriert alle Nachrichten von oben nach unten, also in der Reihenfolge, in der sie auf der Zeitachse vorkommen, durch.
- **Hierarchisch** verwendet eine Dezimaldarstellung, sodass die hierarchische Struktur der Nachrichten im Diagramm deutlich wird. Die Hierarchie wird in Form einer durch Punkte getrennten Nummerierung, gefolgt von einem Doppelpunkt und dem Namen der Nachricht dargestellt.



Es gibt zwei Methoden, um das Nummerierungssystem auszuwählen:

- Klicken Sie auf das jeweilige Symbol in der Symbolleiste.
- Wählen Sie das Nummerierungssystem über das Register **Stile** aus.

© 2016-2022 Altova GmbH

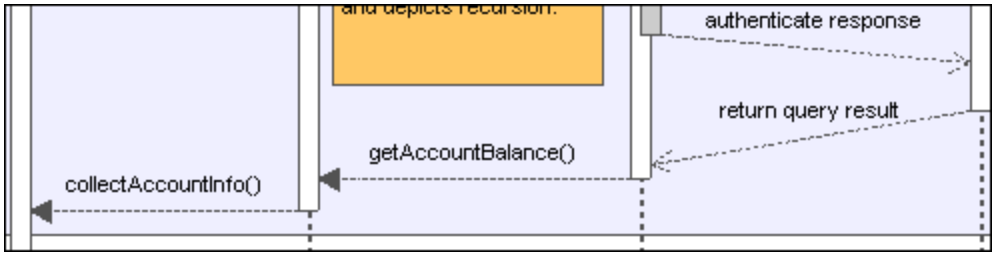
Um das Nummerierungssystem über das Register "Stile" auszuwählen:

1. Klicken Sie auf das Register **Stile** und scrollen Sie hinunter zum Feld **Nachrichtennummern anzeigen**.
2. Klicken Sie auf das Auswahllistenfeld und wählen Sie die gewünschte Nummerierungsoption aus. Die gewählte Nummerierungsoption wird sofort im Sequenzdiagramm angezeigt.

Anmerkung:
Bei Vorhandensein nicht eindeutiger Spuren werden mit dem gewählten Nummerierungssystem nicht immer alle Nachrichten richtig nummeriert. Fügen Sie in diesem Fall Antwortnachrichten hinzu, um Unklarheiten zu beseitigen.

Antwortnachrichten

Zum Erstellen von Antwortnachrichten stehen Nachrichtenantwortsymbole zur Verfügung. Diese werden als strichlierte Pfeile dargestellt.

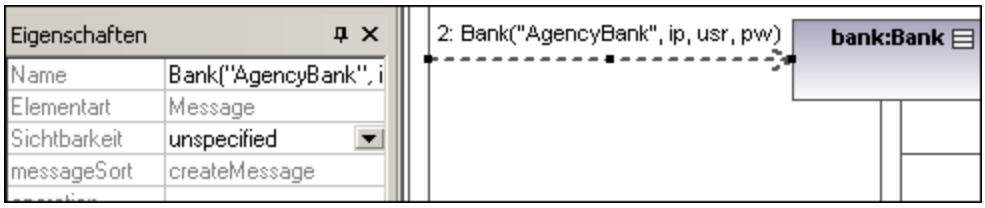


Im Allgemeinen geht es außerdem aus dem unteren Bereich der Aktivierungskästchen - falls diese aktiviert sind - hervor, ob Antwortnachrichten erwartet werden. Wenn Aktivierungskästchen deaktiviert wurden (Register "Stile" | Show Execution Specifics=false), sollten aus Gründen der Übersichtlichkeit Pfeile verwendet werden.

Wenn Sie die Schaltfläche "Automatische Erstellung von Antworten auf Nachrichten (Call) ein/aus" aktivieren, wird automatisch eine syntaktisch korrekte Antwortnachricht erstellt, wenn Sie eine Call-Nachricht zwischen Lebenslinien/Aktivierungskästen erstellen.

Erstellen von Objekten mit Hilfe von Nachrichten

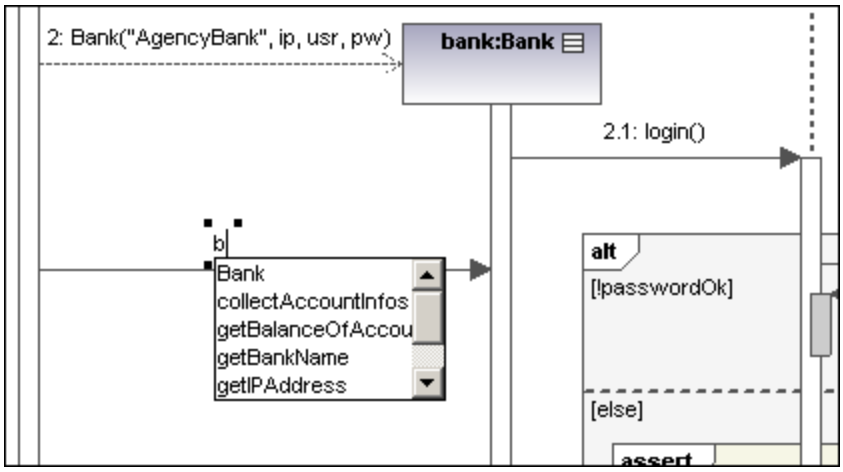
1. Nachrichten können neue Objekte erstellen. Dies erfolgt über das Symbol "Nachricht (Erstellung)" .
2. Ziehen Sie den Nachrichtenpfeil zur Lebenslinie eines bestehenden Objekts, um das Objekt zu erstellen. Diese Nachrichtenart endet in der Mitte eines Objektrechtecks und positioniert das Objektkästchen oft in vertikaler Richtung neu.



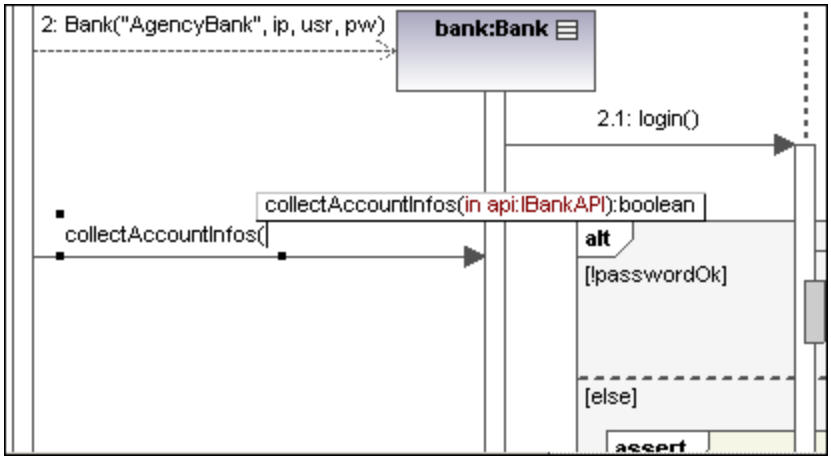
Senden von Nachrichten an bestimmte Klassenmethoden/-operationen in Sequenzdiagrammen

Nachdem Sie eine Klasse aus der Modellstruktur in ein Sequenzdiagramm eingefügt haben, können Sie anschließend mit Hilfe der UModel Syntaxhilfe und der Autokomplettierungsfunktionen eine Nachricht von einer Lebenslinie zu einer bestimmten Methode der Empfängerklasse (Lebenslinie) erstellen.

1. Erstellen Sie eine Nachricht zwischen zwei Lebenslinien, wobei es sich beim Empfängerobjekt um eine Klassenlebenslinie (Bank) handeln muss.Nachdem Sie den Nachrichtenpfeil gezogen haben, erscheint der Name der Nachricht automatisch markiert.
2. Geben Sie über die Tastatur ein Zeichen ein, z.B. "b". Es erscheint ein Popup-Fenster mit einer Liste der verfügbaren Klassenmethoden.



3. Wählen Sie eine Operation aus der Liste aus, z.B. collectAccountinfos und drücken Sie die Eingabetaste.
4. Drücken Sie die Leertaste und anschließend die Eingabetaste, um das automatisch vorgegebene Klammerzeichen auszuwählen.
   Es erscheint ein Syntaxhilfefenster, über das Sie den Parameter korrekt eingeben können.



Erstellen von Operationen in referenzierten Klassen

Wenn Sie die Schaltfläche "Automatische Erstellung von Operationen in der Zielkomponente durch Eingabe von Operationsnamen ein-/ausschalten" aktivieren, wird die entsprechende Operation in der referenzierten Klasse automatisch erstellt, wenn Sie eine Transition erstellen und einen Namen myOperation() eingeben.

Anmerkung: Operationen können nur dann automatisch erstellt werden, wenn sich der Zustandsautomat innerhalb einer Klasse oder Schnittstelle befindet.

Nachrichtensymbole

Nachricht (Aufruf)

Nachricht (Antwort)

Nachricht (Erstellung)

Nachricht (Löschung)

Asynchrone Nachricht (Aufruf)

Asynchrone Nachricht (Antwort)

Asynchrone Nachricht (Löschung)

Verschieben zusammengehöriger Nachrichten ein/aus

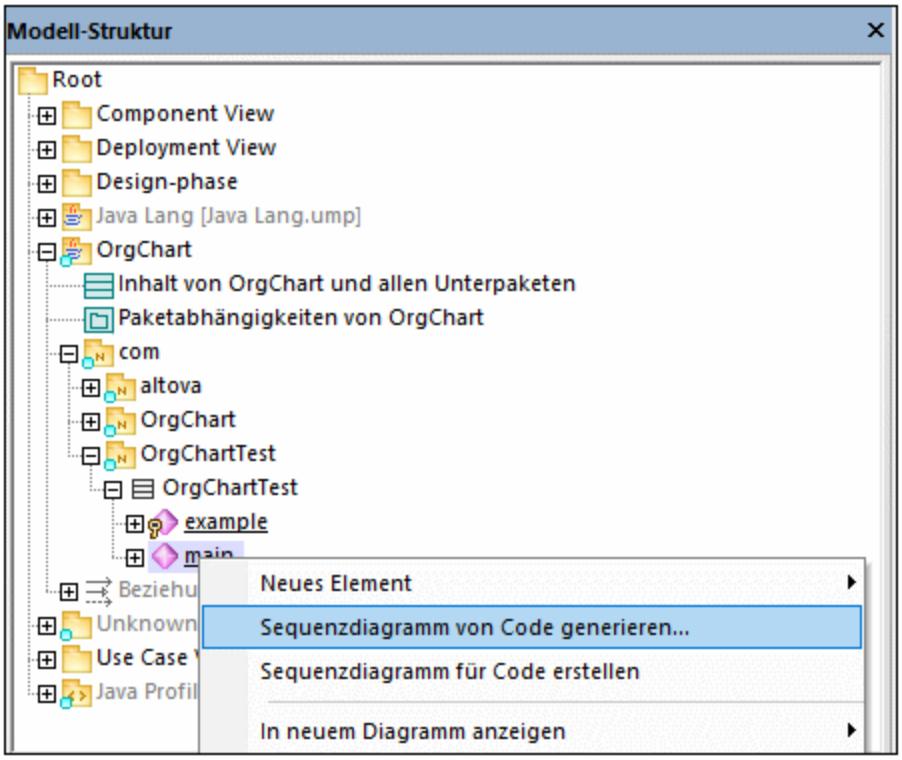
Automatische Erstellung von Antworten auf Nachrichten (Call) ein/aus

Automatische Erstellung von Operationen in der Zielkomponente durch Eingabe von Operationsnamen ein-/ausschalten

8.1.7.2 Generieren von Sequenzdiagrammen anhand von Quellcode

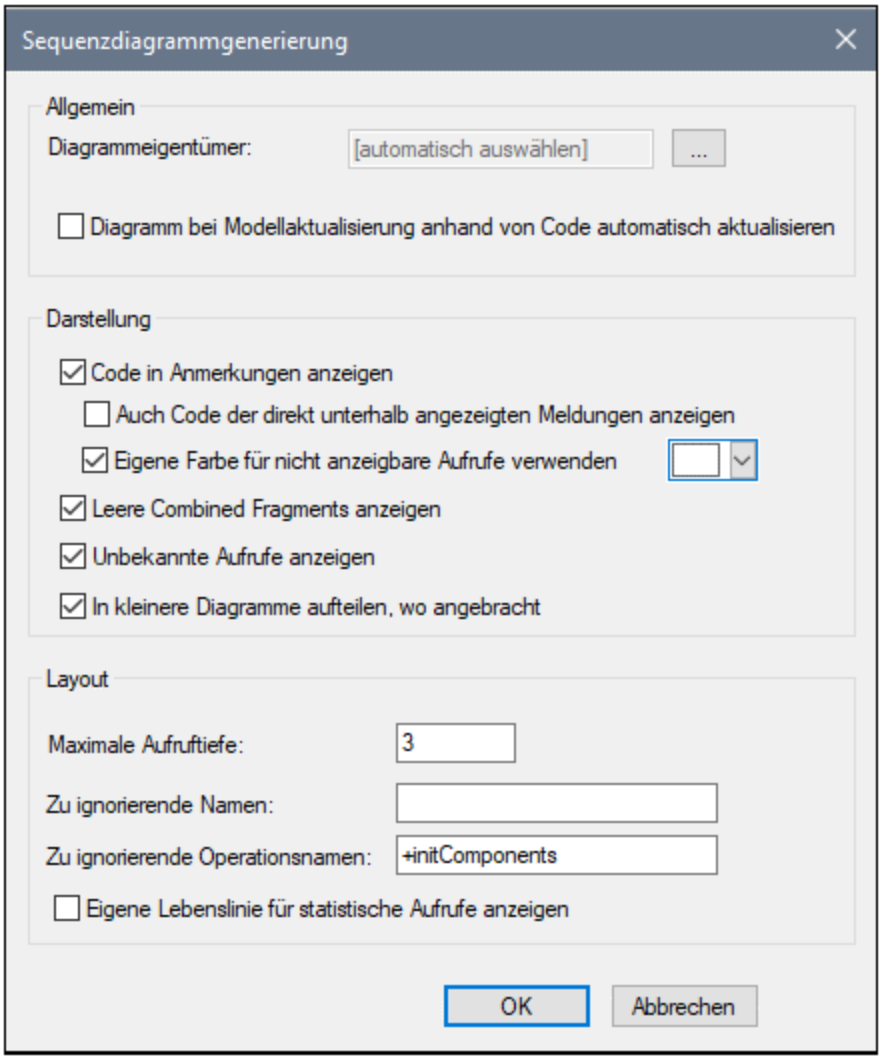
In diesem Beispiel wird gezeigt, wie Sie ein Sequenzdiagramm anhand von einer Methode generieren. Das Projekt, das diese Methode enthält, wird mittels Reverse Engineering anhand von Java-Quellcode erstellt. Sie finden den Java-Quellcode unter dem folgenden Pfad: **C:\Benutzer\<Benutzer>\Dokumente\Altova\UModel2023\UModelExamples\OrgChart.zip**. Entpacken Sie das **OrgChart.zip**-Archiv zuerst in denselben Ordner (Klicken Sie z.B. mit der rechten Maustaste in Windows Explorer auf das Archiv und wählen Sie **Extract All**).

1. Klicken Sie im Menü **Projekt** auf **Quellverzeichnis importieren** und wählen Sie das zuvor entpackte Verzeichnis aus.
2. Befolgen Sie die Anweisungen des Assistenten, um den Quellcode als Java-Projekt zu importieren. Nähere Informationen zu diesem Schritt finden Sie unter Reverse Engineering (Code zu Modell) 69.
3. Nachdem Sie den Code importiert haben, klicken Sie mit der rechten Maustaste in der Modellstruktur auf die Methode `main` der Klasse `OrgChartText` und wählen Sie im Kontextmenü den Befehl **Sequenzdiagramm von Code generieren**.

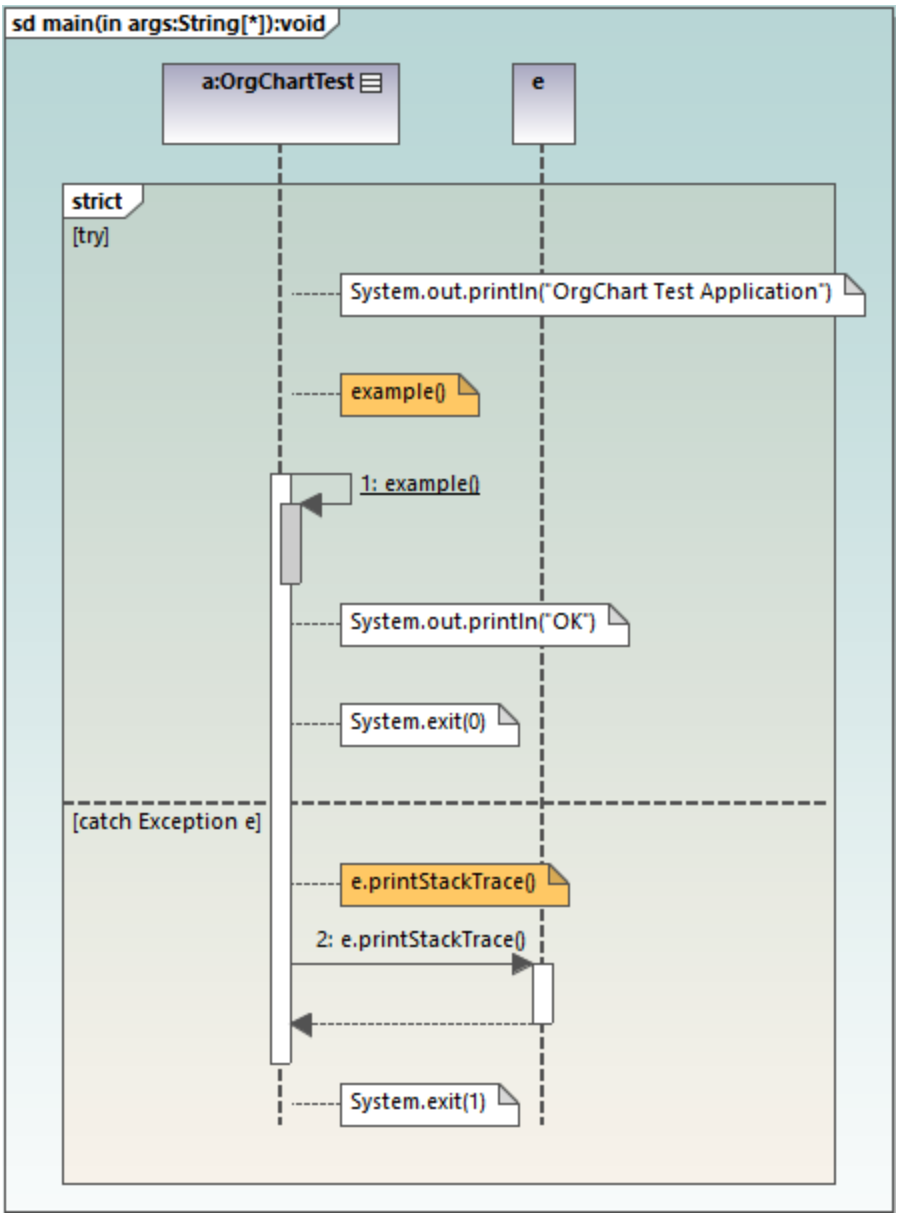


Daraufhin wird das Dialogfeld "Sequenzdiagrammgenerierung" geöffnet, in dem Sie die Einstellungen für die Generierung definieren können.

 © 2016-2022 Altova GmbH



4. Wählen Sie die Darstellungs- und Layout-Optionen aus und klicken Sie anschließend auf **OK**, um das Diagramm zu generieren. Mit den oben gewählten Einstellungen wird das unten gezeigte Sequenzdiagramm erzeugt.

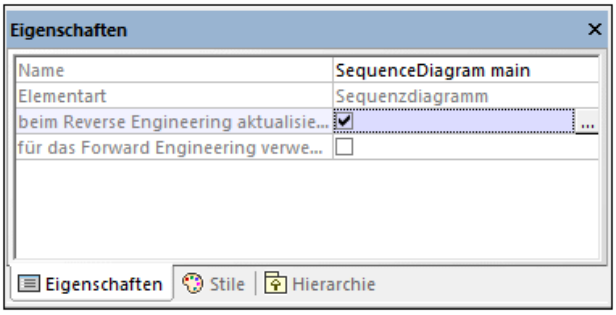


Optionen für die Generierung von Sequenzdiagrammen

In der unten stehenden Tabelle werden die Optionen für die Generierung von Sequenzdiagrammen aufgelistet.

Option	Aufgabe
Diagrammeigentümer	Diese Option kann definiert werden, wenn das Diagramm zum ersten Mal generiert wird. Bei vorhandenen Diagrammen ist diese Information schreibgeschützt.

 © 2016-2022 Altova GmbH

Option	Aufgabe
	Klicken Sie auf die **Auslassungszeichen**, um das Owner-Paket des Diagramms auszuwählen. Andernfalls platziert die Option **[automatisch auswählen]** das Diagramm in das Standardpaket.
Diagramm bei Modellaktualisierung anhand von Code automatisch aktualisieren.	Beim Reverse Engineering (von Code zu Modell) werden Sequenzdiagramme im Modell automatisch neu generiert, vorausgesetzt, Sie haben die Option **Diagramm bei Modellaktualisierung anhand von Code automatisch aktualisieren** aktiviert, als Sie das Diagramm zum ersten Mal generiert haben. Bei bestehenden Diagrammen können Sie diese Option folgendermaßen ändern: 1. Wählen Sie das Sequenzdiagramm in der Modellstruktur oder der Diagrammstruktur aus. 2. Aktivieren Sie im Fenster "Eigenschaften" die Option **beim Rerverse Engineering aktualisieren**.  Wenn Sie das Kontrollkästchen **für das Forward Engineering verwenden** aktivieren, wird bei der Durchführung eines Forward Engineering (von Modell zu Code) bei der Synchronisierung von Modell zu Code Code anhand des Sequenzdiagramms generiert, siehe auch Generieren von Code anhand von Sequenzdiagrammen 378. Wenn die beiden "Engineering"-Kontrollkästchen fehlen, ist das Diagramm wahrscheinlich nur ein Fragment eines größeren Diagramms oder wurde eventuell anhand einer nicht mit Reverse Engineering erstellten Operation generiert.
Code in Anmerkungen anzeigen	Aktivieren Sie dieses Kontrollkästchen, um das Diagramm mit Anmerkungen (Beschriftungen), die Programmcode enthalten, zu generieren.

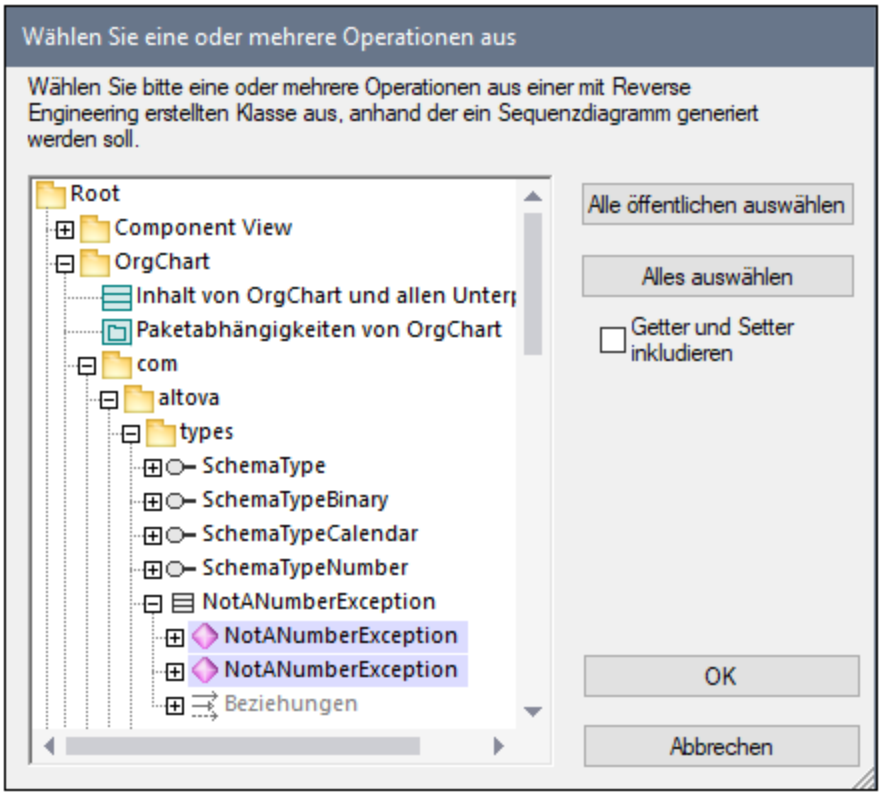
Option	Aufgabe
Auch Code der unterhalb angezeigten Meldungen anzeigen	Selbst in Fällen, in denen ein Codefragment im Diagramm als UML-Meldung angezeigt werden kann, wird der Code dieser Meldung mit dieser Option dennoch als Anmerkung angezeigt.
Eigene Farbe für nicht anzeigbare Aufrufe verwenden	Weist nicht anzeigbaren Aufrufen eine Farbe Ihrer Wahl zu.
Leere Combined Fragments anzeigen	Die Combined Fragment 361-Blöcke werden auch dann im Diagramm beibehalten, wenn sie keinen Inhalt haben.
Unbekannte Aufrufe anzeigen	Wenn diese Option aktiviert ist, werden auch Meldungen für Operationen oder Konstruktoren, die nicht aufgelöst werden konnten (d.h. die im Modell nicht gefunden wurden) angezeigt.
In kleinere Diagramme aufteilen, wo angebracht	Damit werden Sequenzdiagramme in kleinere Subdiagramme aufgeteilt, zwischen denen zur einfacheren Navigation automatisch Hyperlinks angelegt werden.
Maximale Aufruftiefe	Definiert die Aufruftiefe, die im Diagramm verwendet werden soll. Wenn z.B. `method1() method2()` aufruft, die wiederum `method3()` aufruft, und als Aufruftiefe **2** definiert ist, wird nur `method2` angezeigt. `method3` wird nicht mehr angezeigt.
Zu ignorierende Typnamen	Damit können Sie eine kommagetrennte Liste von Typen definieren, die im Sequenzdiagramm nicht aufscheinen sollen, wenn es generiert wird.
Zu ignorierende Operationsnamen	Damit können Sie eine kommagetrennte Liste von Operationen definieren, die im generierten Sequenzdiagramm nicht aufscheinen sollen. Wenn Sie die Operationsnamen zur Liste hinzufügen, wird die komplette Operation ignoriert. Wenn Sie der Operation in der Liste ein + (Pluszeichen) voranstellen, z.B. **+InitComponent**, werden die Operationsaufrufe im Diagramm angezeigt, jedoch ohne Inhalt.
Eigene Lebenslinie für statische Aufrufe anzeigen	Wenn statische Methodenaufrufe vorhanden sind und es im Diagramm bereits eine Instanz dieses Objekts gibt, werden die Meldungen normalerweise mit dieser vorhanden Lebenslinie verbunden. Wenn diese Option aktiviert ist, wird bei der Diagrammgenerierung eine eigene neue Lebenslinie nur für statische Methodenaufrufe für diesen Classifier verwendet.

© 2016-2022 Altova GmbH

8.1.7.2.1 Generieren von Sequenzdiagrammen anhand von Eigenschaften

Sie können auch mehrere Sequenzdiagrammmodelle anhand mehrerer Operationen generieren. Gehen Sie dazu folgendermaßen vor:

1. Wählen Sie die Menüoption **Projekt | Sequenzdiagramme von Code generieren**.



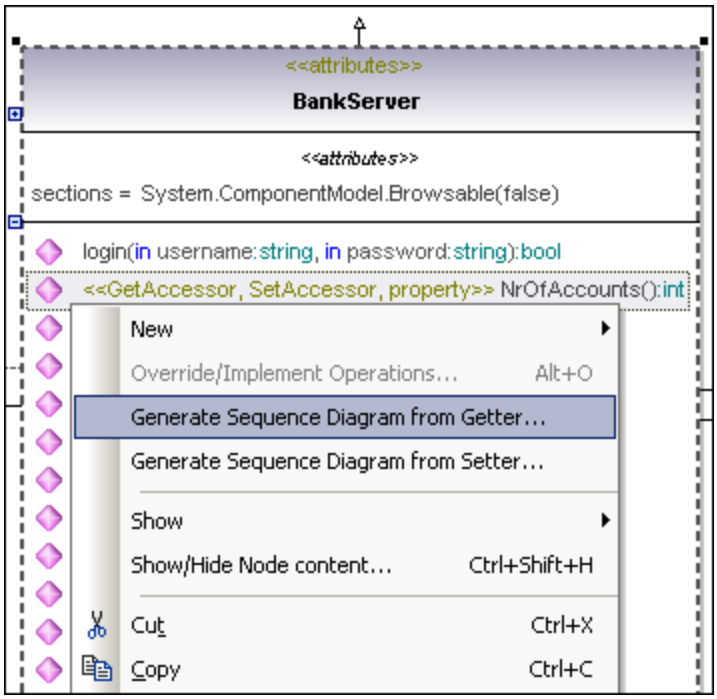
2. Aktivieren Sie die Operationen, für die ein Sequenzdiagramm generiert werden soll, aus und klicken Sie auf **OK**. (Klicken Sie, wo nötig, auf die Schaltflächen **Alle öffentlichen auswählen** und **Alle auswählen**.
3. Aktivieren Sie optional das Kontrollkästchen **Getter und Setter inkludieren**, um Sequenzdiagramme für C#/VB.NET Getter und Setter zu generieren.
4. Klicken Sie auf **OK.** Daraufhin wird ein Dialogfeld geöffnet, in dem Sie die Optionen für die Generierung von Sequenzdiagrammen 374 definieren können.
5. Klicken Sie auf **OK.** Für jede der ausgewählten Operationen wird ein Sequenzdiagramm generiert und automatisch in UModel geöffnet.

Wenn Ihr Projekt groß ist, dauert die Erstellung mehrerer Sequenzdiagramme wahrscheinlich etwas länger. Beachten Sie, dass die ersten 10 Diagramme von UModel automatisch geöffnet werden. Der Rest wird generiert, ohne geöffnet zu werden.

8.1.7.2.2 Generieren von Sequenzdiagrammen anhand von Gettern/Settern

Sie können ein Sequenzdiagramm auch anhand von Getter/Setter-Eigenschaften (in C#, VB .NET) generieren. Gehen Sie dazu folgendermaßen vor:

1. Klicken Sie mit der rechten Maustaste auf eine Operation mit einem `GetAccessor/SetAccessor` Stereotyp.



2. Wählen Sie im Kontextmenü den Befehl **Sequenzdiagramm von Code (Getter / Setter) generieren** aus. Daraufhin wird ein Dialogfeld geöffnet, in dem Sie die Optionen für die Generierung von Sequenzdiagrammen 374 definieren können.
3. Klicken Sie auf **OK**, um das Sequenzdiagramm zu generieren.

8.1.7.3 Generieren von Code anhand eines Sequenzdiagramms

UModel kann Code anhand eines mit mindestens einer Operation verknüpften Sequenzdiagramms generieren.

Die Generierung von Code anhand von Sequenzdiagrammen steht zur Verfügung für:

- VB.NET, C# und Java
- UModel Standalone Edition, Eclipse und Visual Studio Edition
- alle drei UModel Editions

Code anhand von Sequenzdiagrammen kann auf zwei Arten erstellt werden:

- durch einen Reverse Engineering Vorgang. Siehe dazu Generieren von Sequenzdiagrammen anhand von Quellcode 371,

 © 2016-2022 Altova GmbH

- indem ein mit einer Operation verknüpftes Sequenzdiagramm von Grund auf **neu** erstellt wird. Mit einem Rechtsklick auf die Operation in der Modell-Struktur können Sie über das Kontextmenü den Befehl Sequenzdiagramm für Code erstellen[381] aufrufen.

Wenn Sie ein mit Reverse Engineering erstelltes Sequenzdiagramm als Basis verwenden, vergewissern Sie sich, dass die Option "Code in Anmerkungen anzeigen" beim Reverse Engineering aktiviert ist, damit kein Code verloren geht, wenn Sie wieder mit dem Forward Engineering beginnen. Dies ist darauf zurückzuführen, dass in einem UML-Sequenzdiagramm nicht alle Funktionen von VB.NET, Java und C# angezeigt werden können. Daher werden diese Codeabschnitte in Form von Code-Anmerkungen angezeigt.

So fügen Sie bei der Erstellung eines Sequenzdiagramms einfachen Text als Code hinzu:

1. Hängen Sie eine Anmerkung an eine Lebenslinie in einem Sequenzdiagramm an.
2. Geben Sie den Code ein, der in den endgültigen Quellcode geschrieben werden soll.
   Aktivieren Sie im Bereich "Eigenschaften" das Kontrollkästchen "Ist Code", damit diese Anmerkung zur Verfügung steht.

Ein Beispiel dazu finden Sie unter Hinzufügen von Code zu Sequenzdiagrammen[382].

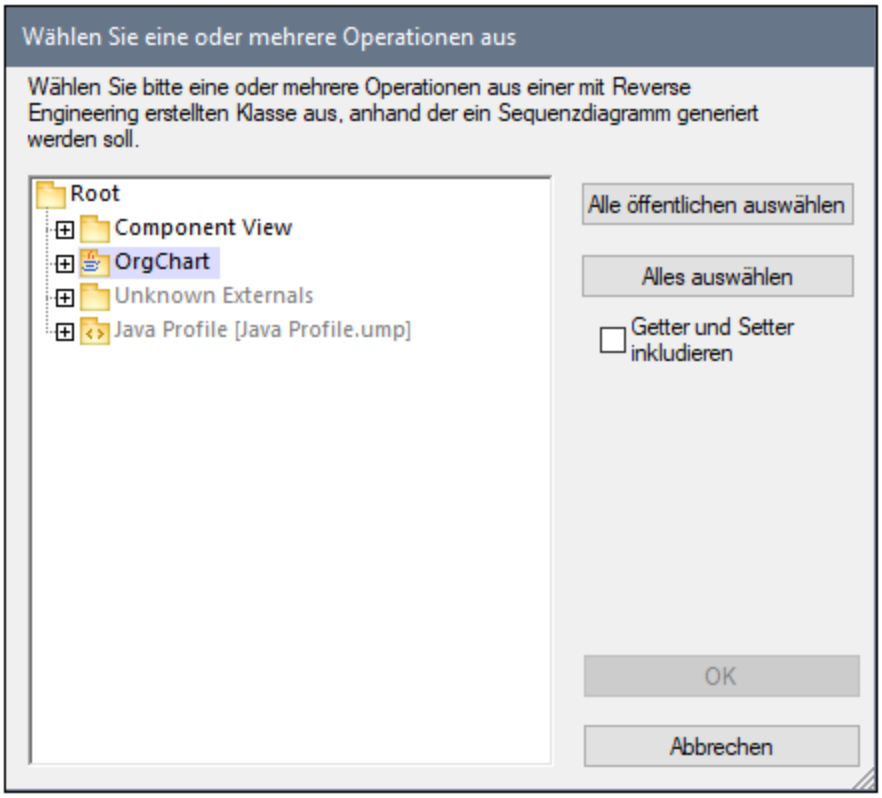
Wenn bei jedem Start eines Code Engineering automatisch ein Sequenzdiagramm für das Code Engineering verwendet werden soll:

1. Wählen Sie das Diagramm im Fenster "Modell-Struktur" oder "Diagramm-Struktur" aus.
2. Aktivieren Sie im Bereich **Eigenschaften** das Kontrollkästchen **Für das Forward Engineering verwenden**.

Der alte Code geht beim Forward Engineering von Code anhand eines Sequenzdiagramms immer verloren, da er durch den neuen Code überschrieben wird.

So generieren Sie über das Menü "Projekt" Code:

1. Wählen Sie die Menüoption **Projekt | Code von Sequenzdiagrammen generieren**.
   Daraufhin werden Sie aufgefordert, das/die gewünschten Sequenzdiagramm(e) auszuwählen.

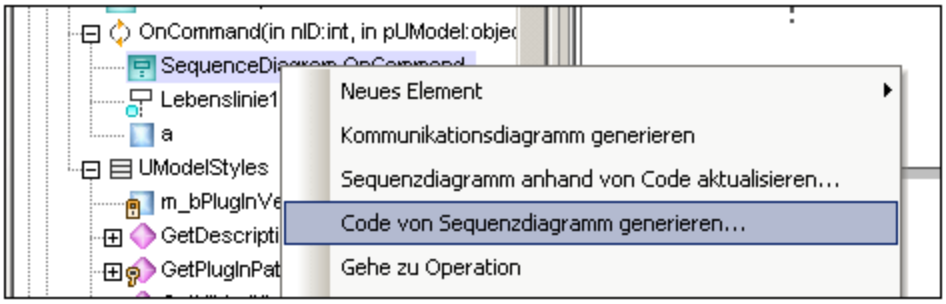


Wenn Sie auf die Schaltfläche "Alles auswählen" klicken, werden alle Sequenzdiagramme im UModel-Projekt ausgewählt.

2. Klicken Sie auf OK, um den Code zu generieren.
   Im Fenster "Meldungen" wird der Status der Codegenerierung angezeigt.

So generieren Sie über die Modell-Struktur Code:

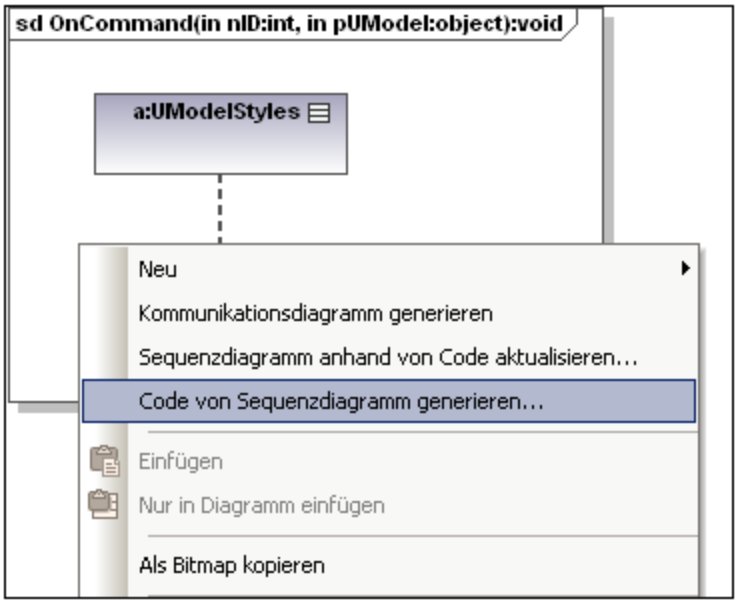
- Klicken Sie mit der rechten Maustaste auf ein Sequenzdiagramm und wählen Sie den Befehl "Code von Sequenzdiagramm generieren".



 © 2016-2022 Altova GmbH

So generieren Sie ein Sequenzdiagramm, das Code einer Operation enthält:

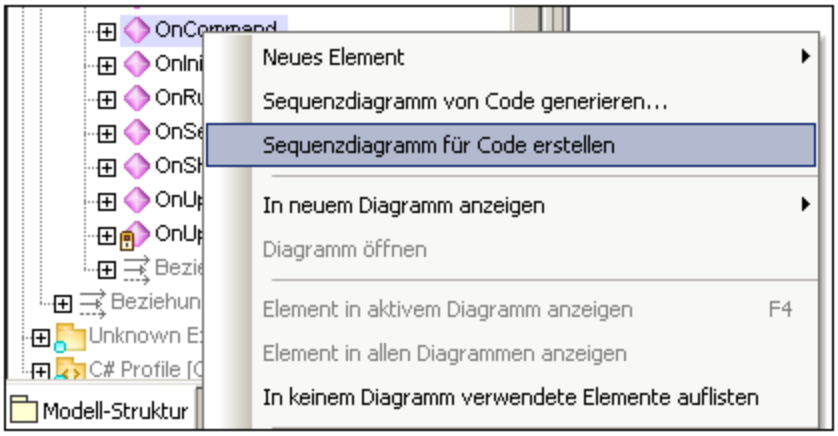
1. Klicken Sie in den leeren Bereich des Sequenzdiagramms, das Code einer Operation enthält.
2. Wählen Sie den Befehl "Code von Sequenzdiagramm generieren".



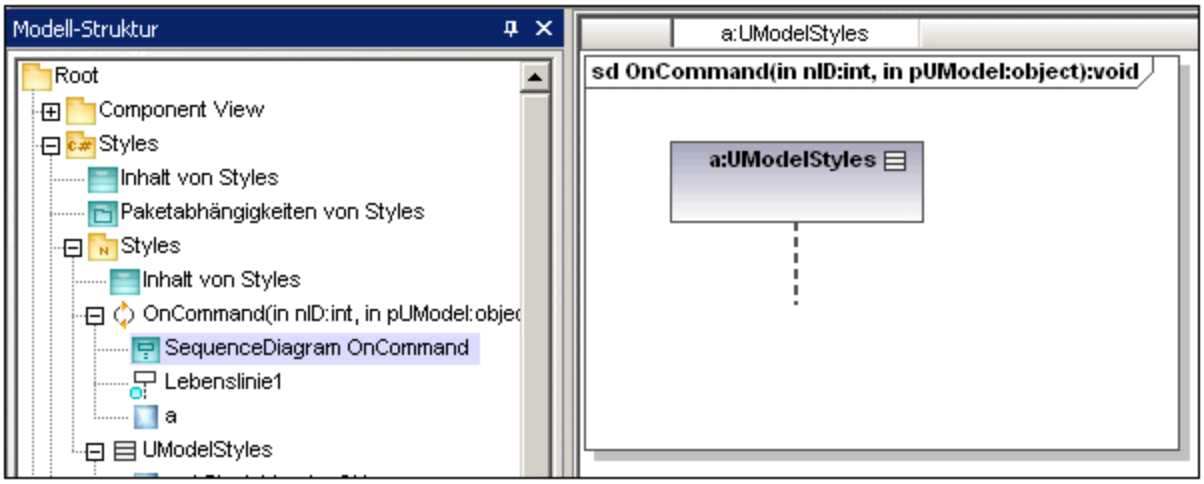
Mit diesem Befehl wird das Forward Engineering gestartet.

So erstellen Sie ein Sequenzdiagramm für Code (Engineering):

1. Klicken Sie in der Model-Struktur mit der rechten Maustaste auf eine Operation und wählen Sie den Befehl "Sequenzdiagramm für Code erstellen".



Sie werden daraufhin gefragt, ob Sie das neue Diagramm für das Forward Engineering verwenden möchten.



Das Ergebnis ist ein neues Sequenzdiagramm, das die Lebenslinie dieser Klasse enthält.

8.1.7.3.1 Hinzufügen von Code zu einem Sequenzdiagramm

Programmcode kann anhand eines neuen und eines mit Reverse Engineering erstellten Sequenzdiagramms generiert werden, allerdings nur für ein Sequenzdiagramm, das mit der "Hauptoperation" verknüpft ist.

Beim Reverse Engineering werden Standardelemente des Sequenzdiagramms, wie z.B. CombinedFragments Codierungselementen (z.B. "if" Anweisungen, Schleifen usw.) "zugewiesen" bzw. auf diese "gemappt".

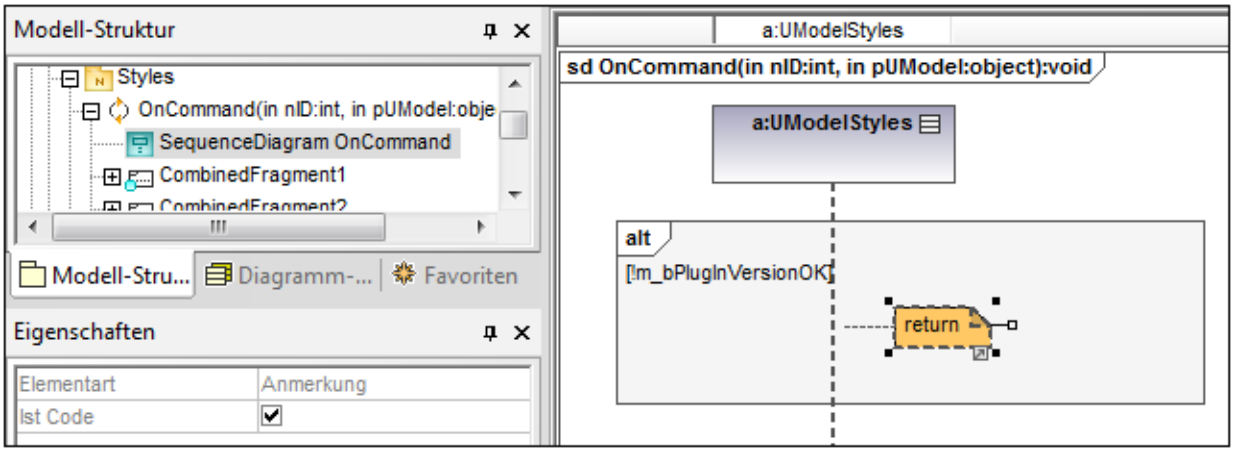
Bei Programmanweisungen, für die es keine entsprechenden Sequenzdiagrammelemente gibt, wie z.B. "i = i+1" werden in UModel "Code- Anmerkungen" verwendet, um Code zu Diagrammen hinzuzufügen. Diese Anweisungen müssen anschließend mit der Lebenslinie verknüpft werden.

Beachten Sie, dass UModel diese Codefragmente nicht überprüft oder parst. Sie müssen vorher sicherstellen, dass die Codefragmente korrekt und kompilierbar sind.

So fügen Sie Code zu einem Sequenzdiagramm hinzu:

1. Klicken Sie auf die Schaltfläche "Anmerkung" und klicken Sie anschließend auf das Modellelement, an dem diese eingefügt werden soll, z.B. CombinedFragment.
2 Geben Sie das Codefragment ein, z.B. return.
3. Klicken Sie auf den Ziehpunkt der eingefügten Anmerkung und ziehen Sie den Cursor auf die Lebenslinie.
4. Aktivieren Sie auf dem Register "Eigenschaften" das Kontrollkästchen "Ist Code", damit dieses Codefragment bei der Codegenerierung inkludiert wird.

 © 2016-2022 Altova GmbH

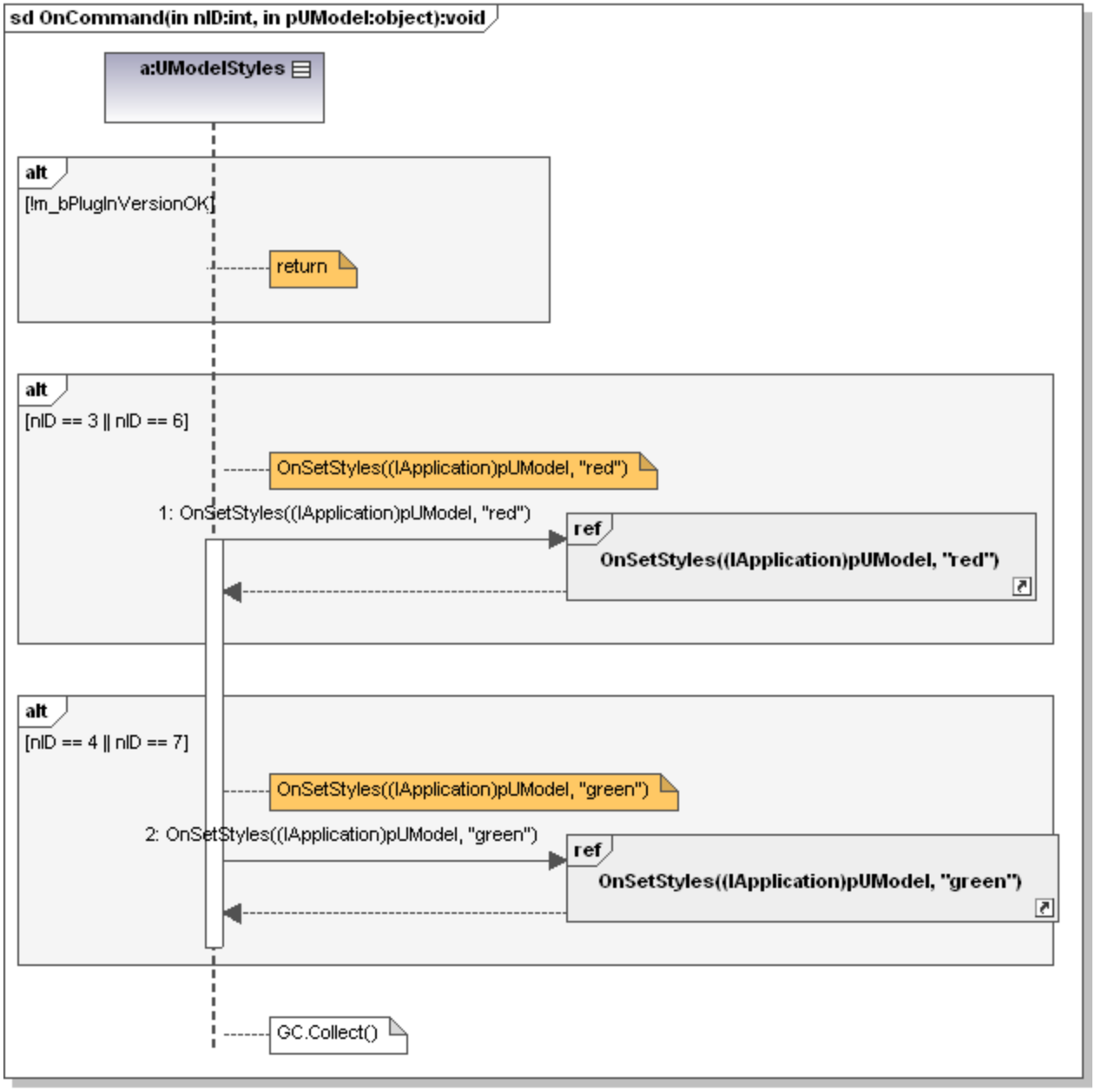


Wenn Sie eine Anmerkung in einem Sequenzdiagramm auswählen, die zur Codegenerierung verwendet werden kann, so steht die Eigenschaft "Ist Code" im Fenster "Eigenschaften" zur Verfügung. Durch Aktivieren/Deaktivieren des Kontrollkästchens können Sie zwischen "normalen" Anmerkungen und Codegenerierungsanmerkungen wechseln.

Normale Anmerkungen:	return
Codegenerierungsanmerkungen	return - werden mit einem schattierten Eck angezeigt

Wenn das Kontrollkästchen "Für das Forward Engineering verwenden" aktiviert ist, wird der Code automatisch bei jedem Forward Engineering aktualisiert. Wenn Änderungen am Sequenzdiagramm vorgenommen wurden, wird der Code der Operation immer überschrieben.

Das unten gezeigte Sequenzdiagramm wurde durch Rechtsklick auf die OnCommand-Operation und Auswahl des Befehls "Sequenzdiagramm von Code generieren" generiert. Der C# Code dieses Beispiels steht im OrdnerC:\Benutzer\<benutzer>\Dokumente\Altova\UModel2023\UModelExamples\IDEPlugIn\Styles\ zur Verfügung. Über die Option "Projekt | Quellprojekt importieren können Sie das Projekt importieren.



Anhand dieses Sequenzdiagramms wird der unten gezeigte Code generiert.

```
Public void OnCommand(int nID, object pUModel)
{
   //Generated by UModel. This code will be overwritten when you re-run code generation.

   if (!m_bPlugINVersionOK)
   {
   return;
   }

   if (nID == 3 || nID == 6)
```

```
    {
    OnSetStyles((IApplication)pUModel, "red");
    }

    if (nID == 4 || nID == 7)
    {
    OnSetStyles((IApplication)pUModel, "green");
    }
    GC.Collect();

}
```

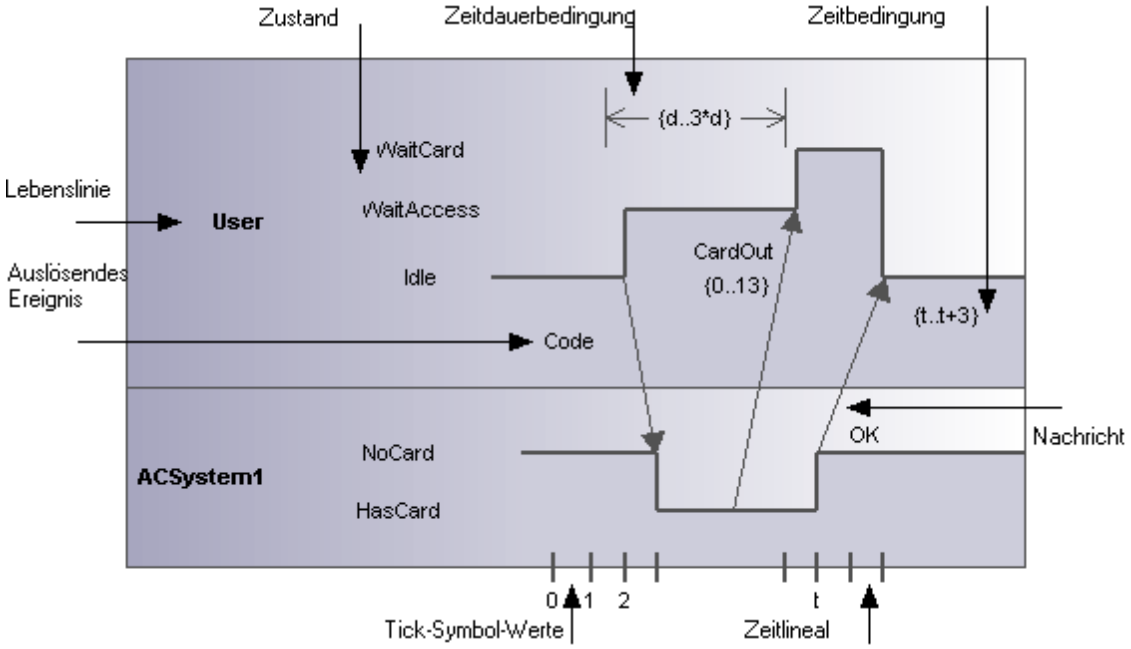
8.1.8 Zeitverlaufsdiagramm

Altova Website: UML-Zeitverlaufsdiagramme

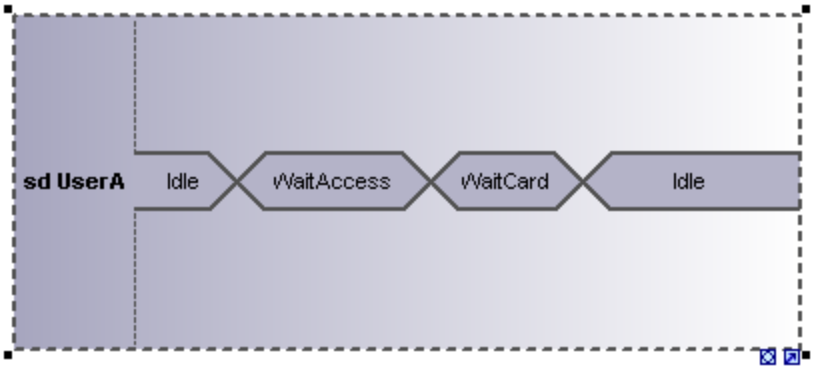
Zeitverlaufsdiagramme dienen zum Darstellen von Änderungen am Status oder Zustand von einem oder mehreren miteinander in Wechselbeziehung stehenden Objekten über einen bestimmten Zeitraum. Zustände werden in Form von Zeitlinien dargestellt, die auf Message Events reagieren, wobei eine Lebenslinie eine Classifier Instance oder Classifier Role darstellt.

Bei Zeitverlaufsdiagrammen handelt es sich um eine Sonderform eines Sequenzdiagramms. Im Unterschied zum Sequenzdiagramm sind beim Zeitverlaufsdiagramm die Achsen vertauscht, d.h. der Zeitverlauf wird aufsteigend von links nach rechts dargestellt und Lebenslinien werden in separaten vertikal angeordneten Bereichen angezeigt.

Zeitverlaufsdiagramme werden im Allgemeinen zum Entwerfen von eingebetteter Software oder Echtzeitsystemen verwendet.



Es gibt zwei verschiedene Arten von Zeitverlaufsdiagrammen: Die eine Art enthält, wie oben gezeigt, eine Zustands-Zeitlinie, die andere allgemeinere Art enthält, wie unten gezeigt, eine Lebenslinie.



8.1.8.1 Einfügen von Elementen des Zeitverlaufsdiagramms

Verwendung der Symbolleisten-Schaltflächen

1. Klicken Sie in der Zeitverlaufsdiagramm-Symbolleiste auf das entsprechende Zeitverlaufselement.

2. Klicken Sie zum Einfügen des Elements in das Zeitverlaufsdiagramm. Um mehrere Elemente des ausgewählten Typs einzufügen, halten Sie die **Strg-Taste** gedrückt und klicken Sie in das Diagrammfenster.

Ziehen vorhandener Elemente in das Zeitverlaufsdiagramm

Sie können Elemente aus anderen Diagrammen, z.B. Klassen, in ein Zeitverlaufsdiagramm einfügen.

1. Suchen Sie das gewünschte Element auf dem Register "Modell-Struktur" (über das Suchfunktionstextfeld oder durch Drücken der Tasten Strg + F).
2. Ziehen Sie das Element/die Elemente in das Diagramm.

8.1.8.2 Lebenslinie

Das Element "Lebenslinie" bildet einen Teil einer Interaktion und steht in zwei Formen zur Verfügung:

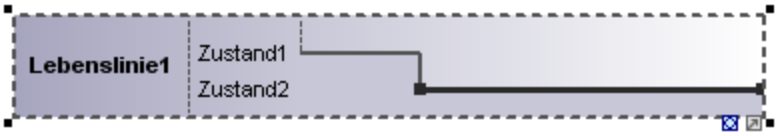
1. als Zustands-Lebenslinie
2. als allgemeine Wertverlaufslinie

Um eine Lebenslinie **mit mehreren** Zeilen zu erstellen, drücken Sie Strg + Eingabetaste, um eine neue Zeile anzulegen.

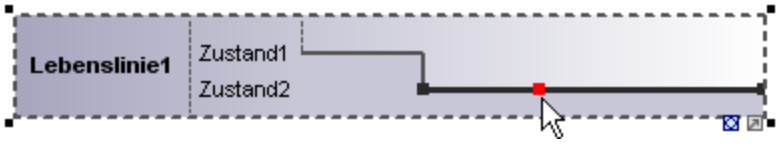
 © 2016-2022 Altova GmbH

So fügen Sie eine Zustands-Lebenslinie (Zustandsinvariante) ein und definieren Zustandsänderungen:

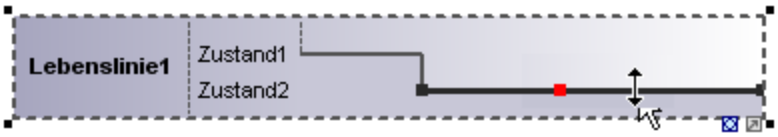
1. Klicken Sie auf das Symbol "Lebenslinie (Zustand)" in der Titelleiste und anschließend in das Zeitverlaufsdiagramm, um die Lebenslinie einzufügen.



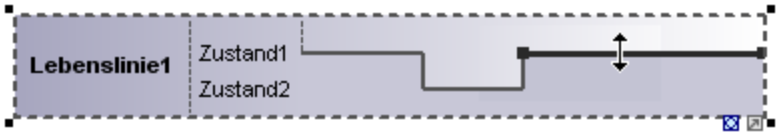
2. Geben Sie einen Namen für die Lebenslinie ein, um den Standardnamen (Lebenslinie1) wenn nötig zu ändern.
3. Platzieren Sie den Mauszeigen über einen Abschnitt einer der Zeitlinien und klicken Sie auf die linke Maustaste. Daraufhin wird die Linie ausgewählt.
4. Verschieben Sie den Mauszeiger an die Stelle, an der die Zustandsänderung eintreten soll, und klicken Sie nochmals auf die linke Maustaste. Beachten Sie: Während dieses Vorgangs wird ein Pfeil mit zwei Spitzen angezeigt. An der Klickposition erscheint ein rotes Kästchen, das die Zeile an diesem Punkt teilt.



5. Ziehen Sie den Cursor auf der Linie nach rechts und ziehen Sie die Linie nach oben.



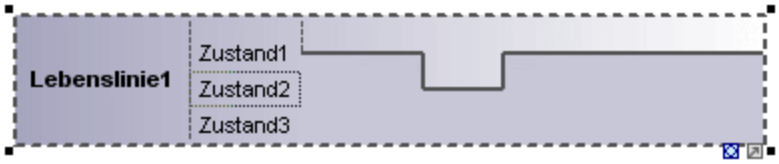
Beachten Sie, dass die Linien nur zwischen bestehenden Zuständen der aktuellen Lebenslinie verschoben werden kann.



Sie können beliebig viele Zustandsänderungen pro Lebenslinie definieren. Wenn auf einer Linie ein rotes Kästchen angezeigt wird, können Sie an eine beliebige Stelle im Diagramm klicken, um es zu löschen.

So fügen Sie einen neuen Zustand zur Lebenslinie hinzu:

1. Rechtsklicken Sie auf die Lebenslinie und wählen Sie **Neu | Zustand (Zustandsinvariante)**. Daraufhin wird ein neuer Zustand, z.B. Zustand2 zur Lebenslinie hinzugefügt.



So verschieben Sie einen Zustand innerhalb einer Lebenslinie:

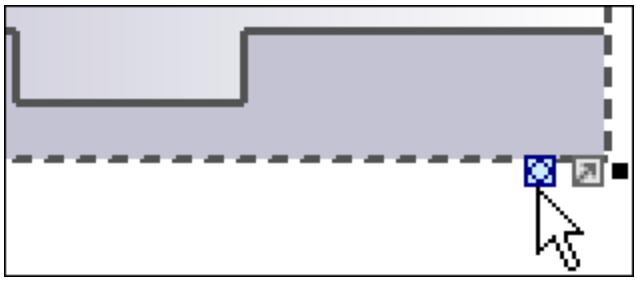
1. Klicken Sie auf die Beschriftung des Zustands, den Sie verschieben möchten.
2. Ziehen Sie den Zustand an eine andere Stelle innerhalb der Lebenslinie.

So löschen Sie einen Zustand aus einer Lebenslinie:

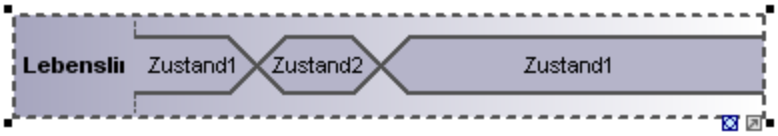
1. Klicken Sie auf den Zustand und drücken Sie die Entf-Taste oder klicken Sie auf die rechte Maustaste und wählen Sie den Befehl "Löschen".

So wechseln Sie zwischen Zeitverlaufsdiagrammtypen:

1. Klicken Sie auf das Symbol "Darstellung wechseln" rechts unterhalb der Lebenslinie.



Daraufhin wird stattdessen eine Wertverlaufslebenslinie angezeigt, der Kreuzungspunkt stellt eine Änderung eines Zustands/Werts dar.

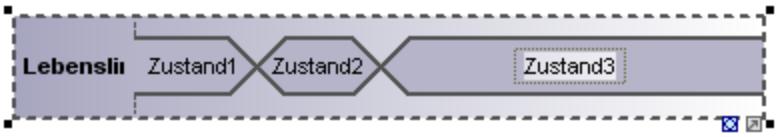


Anmerkung: Wenn Sie auf das Symbol "**Lebenslinie (allgemeiner Wert)** klicken, wird die oben gezeigte Lebenslinie eingefügt. Sie können jederzeit zwischen den beiden Darstellungen wechseln.

So fügen Sie einen neuen Zustand zur Wertverlaufslebenslinie hinzu:

1. Rechtsklicken Sie auf die Lebenslinie und wählen Sie **Neu | Zustand (Zustandsinvariante)**.
2. Bearbeiten Sie den neuen Namen, z.B. Zustand3 und bestätigen Sie die Änderung mit der Eingabetaste.

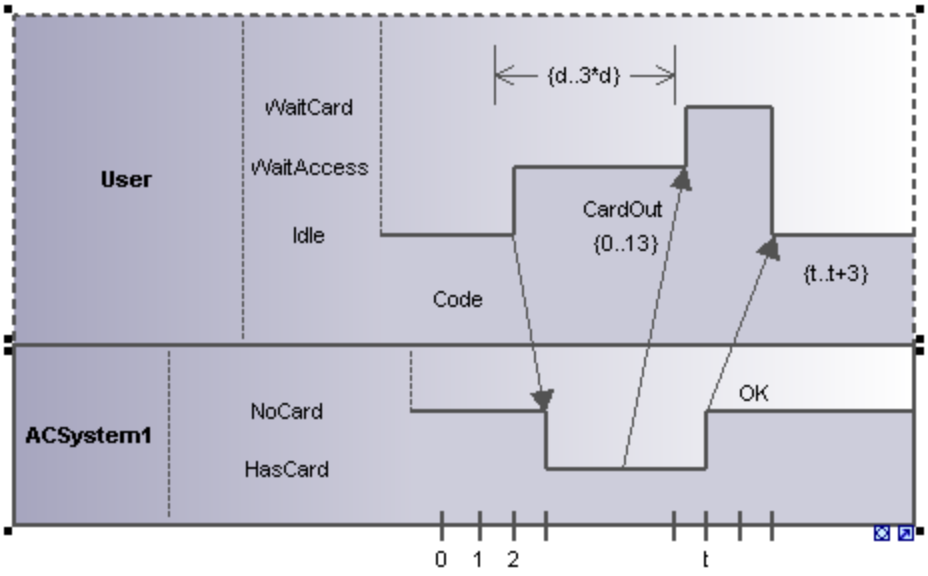
 © 2016-2022 Altova GmbH



Daraufhin wird ein neuer Zustand zur Lebenslinie hinzugefügt.

Gruppieren von Lebenslinien

Beim Platzieren oder Stapeln von Lebenslinien werden diese automatisch korrekt angeordnet, wobei auch hinzugefügte Häkchen erhalten bleiben. Durch Ziehen des entsprechenden Nachrichtenobjekts können auch Nachrichten zwischen separaten Lebenslinien erstellt werden.

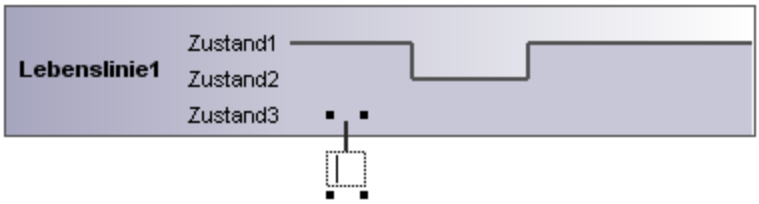


8.1.8.3 Tick-Symbol

Das **Tick-Symbol** dient zum Einfügen der Tick-Symbole eines Zeitverlaufslineals auf einer Lebenslinie.

So fügen Sie ein Tick-Symbol ein:

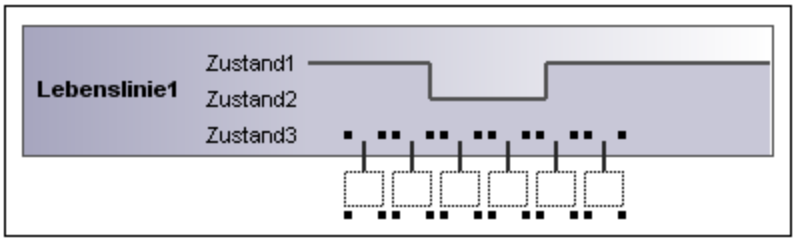
1. Klicken Sie auf das Tick-Symbol in der Symbolleiste und anschließend auf die Lebenslinie, um es einzufügen.



2. Um mehrere Tick-Symbole einzufügen, halten Sie die Strg-Taste gedrückt, während Sie auf verschiedene Positionen am Rand der Lebenslinie klicken.
3. Geben Sie in das dafür vorgesehene Feld eine Beschriftung für das Tick-Symbol ein.
   Zum Neupositionieren der Tick-Symbole auf der Lebenslinie ziehen Sie die Symbole mit der Maus an die gewünschte Stelle.

So ordnen Sie die Tick-Symbole in gleichmäßigen Abständen auf einer Lebenslinie an:

1. Ziehen Sie im Hauptfenster ein Rechteck auf, um die einzelnen Tick-Symbole zu markieren.
2. Klicken Sie in der Symbolleiste auf das Symbol **Waagrecht anordnen** .

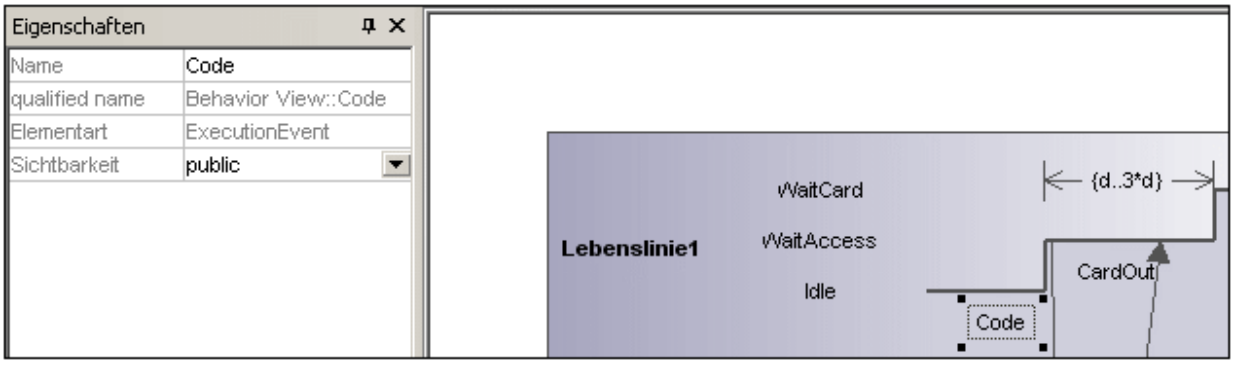


8.1.8.4 Auslösendes Ereignis

Das ExecutionEvent "Auslösendes Ereignis" dient dazu, eine Änderung im Zustand eines Objekts anzuzeigen, die durch das entsprechende auslösende Ereignis verursacht wurde. Die eingehenden Ereignisse werden mit einer Beschriftung versehen, um das Ereignis zu kennzeichnen, das die Zustandsänderung verursacht.

So fügen Sie ein auslösendes Ereignis ein:

1. Klicken Sie auf das Symbol "Auslösendes Ereignis" und anschließend auf die entsprechende Stelle auf der Zeitlinie, an der die Veränderung stattfindet.



2. Geben Sie einen Namen für das Ereignis ein. In diesem Beispiel heißt das Ereignis "Code".

© 2016-2022 Altova GmbH

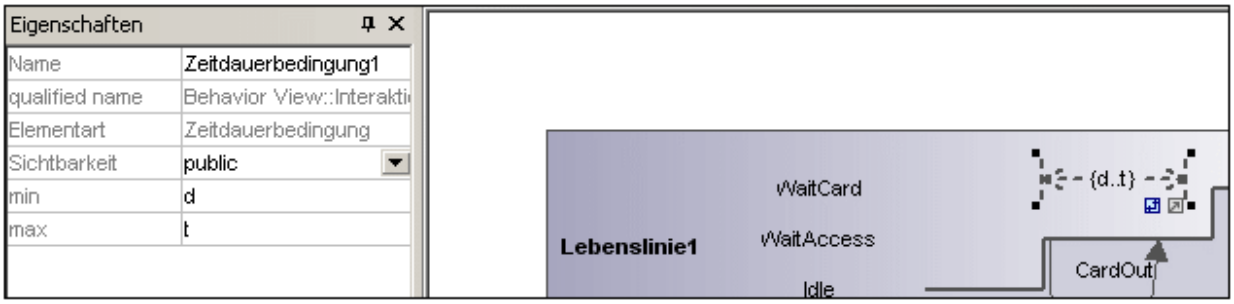
Beachten Sie, dass die Eigenschaften des Ereignisses auf dem Register "Eigenschaften" angezeigt werden.

8.1.8.5 Zeitdauerbedingung

Eine **Zeitdauerbedingung** definiert eine Wertespezifikation zur Angabe einer Zeitdauer zwischen einem Start- und einem Endpunkt. Eine Zeitdauer ist oft ein Ausdruck zur Darstellung der Ticks der Uhr, die während dieser Zeitdauer verstreichen.

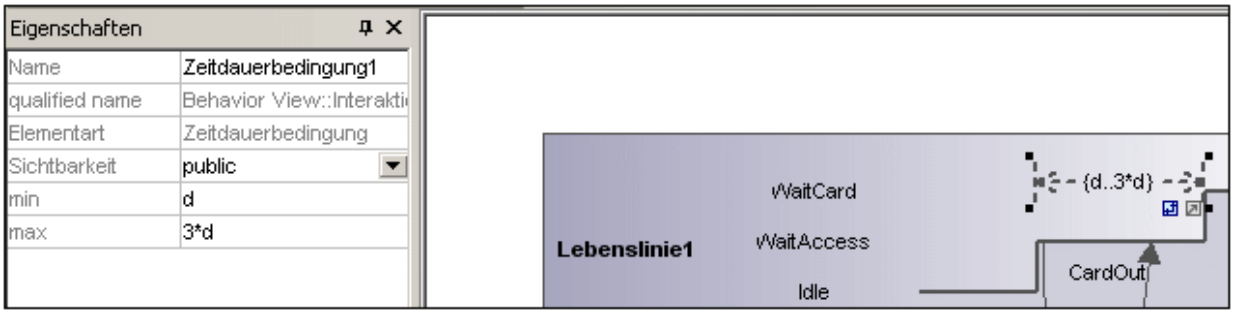
So fügen Sie eine Zeitdauerbedingung ein:

1. Klicken Sie auf das Symbol "Zeitdauerbedingung" und anschließend auf die Stelle auf der Lebenslinie, an der die Bedingung angezeigt werden soll.



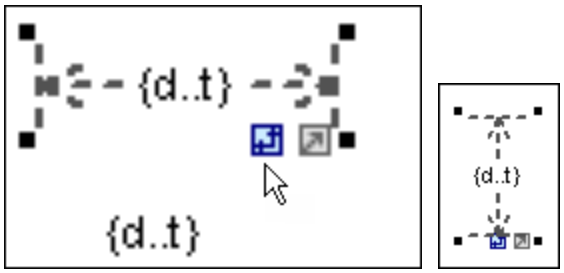
Die Standard-Minimum- und Maximum-Werte "d..t" werden automatisch vorgegeben. Durch Doppelklick auf die Zeitdauerbedingung oder durch Bearbeitung der Werte im Fenster "Eigenschaften" können Sie diese Werte bearbeiten.

2. Mit Hilfe der Ziehpunkte können Sie die Größe des Objekts gegebenenfalls anpassen.



Ändern der Ausrichtung der Zeitdauerbedingung:

1. Klicken Sie auf das "Umdrehen"-Symbol, zum die Bedingung vertikal auszurichten.

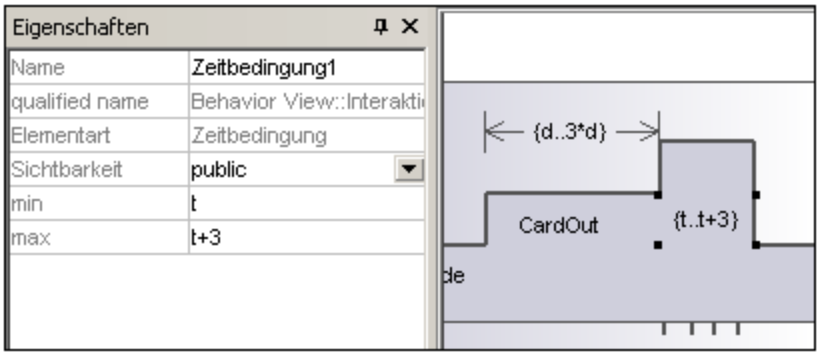


8.1.8.6 Zeitbedingung

Eine **Zeitbedingung** wird im Allgemeinen als grafische Assoziation zwischen einem Zeitintervall und dem Konstrukt, das es einschränkt, dargestellt. Normalerweise handelt es sich um eine grafischen Assoziation zwischen dem Auftreten eines Ereignisses und einem Zeitintervall.

So fügen Sie eine Zeitbedingung ein:

1. Klicken Sie auf das Symbol "Zeitbedingung" und anschließend auf die entsprechende Position auf der Lebenslinie, an der die Bedingung angezeigt werden soll.



Die Standard-Minimum- und Maximum-Werte "d..t" werden automatisch vorgegeben. Durch Doppelklick auf die Zeitbedingung oder durch Bearbeitung der Werte im Fenster "Eigenschaften" können Sie diese Werte bearbeiten.

8.1.8.7 Nachricht

Eine Nachricht ist ein Modellierungselement, das eine bestimmte Art von Kommunikation in einer Interaktion definiert. Dabei kann es sich z.B. um das Auslösen eines Signals, den Aufruf einer Operation, das Erstellen oder Löschen einer Instanz handeln. Die Nachricht definiert die Art der durch die absendende Ausführungsspezifikation definierten Kommunikation sowie den Absender und den Empfänger.

Verwenden Sie die folgenden Symbolleisten-Schaltflächen, um bestimmte Nachrichtentypen hinzuzufügen:

Message (Aufruf)

 © 2016-2022 Altova GmbH

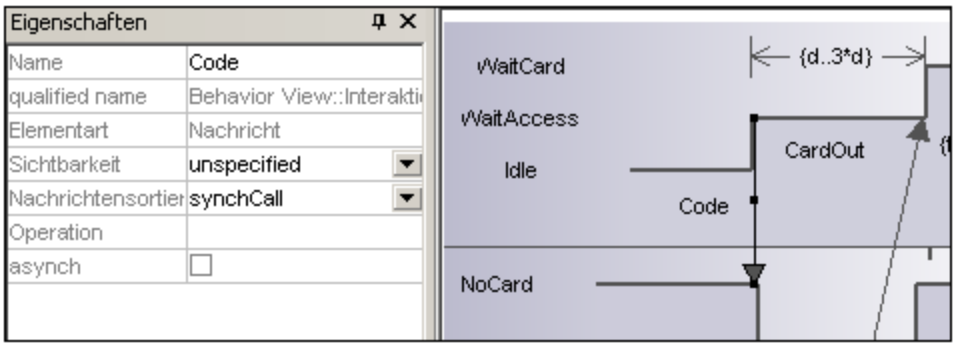
Nachricht (Antwort)

Asynchrone Nachricht (Aufruf)

Nachrichten werden zwischen sendenden und empfangenden Zeitlinien gesendet und werden als beschriftete Pfeile angezeigt.

So fügen Sie eine Nachricht ein:

1. Klicken Sie auf das entsprechende Nachrichtensymbol in der Symbolleiste.
2. Klicken Sie an eine beliebige Stelle auf der Zeitlinie für das sendende Objekt, z.B. auf Idle.
3. Ziehen Sie die Nachrichtenlinie auf die Zeitlinie der empfangenden Objekte, z.B. NoCard. Die Lebenslinie erscheint markiert, wenn die Nachricht an diese bestimmte Stelle gezogen werden kann.



- Die Richtung, in die Sie den Pfeil ziehen, bestimmt die Richtung der Nachricht. Antwortnachrichten können in jede der beiden Richtungen weisen.
- Wenn Sie auf ein Nachrichtensymbol klicken und dabei die Strg-Taste gedrückt halten, können Sie mehrere Nachrichten einfügen, indem Sie mehrmals ins Diagramm klicken und die Maus ziehen.

So löschen Sie eine Nachricht:

1. Klicken Sie auf die gewünschte Nachricht.
2. Drücken Sie die Entf-Taste, um die Nachricht aus dem Modell zu löschen oder rechtsklicken Sie darauf und wählen Sie den Befehl "Aus Diagramm löschen".

8.2 Strukturdiagramme

Diese Diagramme dienen zur Darstellung der Strukturelemente, aus denen ein System oder eine Funktion besteht. Es werden sowohl die statischen, z.B. Klassendiagramm, als auch die dynamischen Beziehungen, z.B. Objektdiagramm, dargestellt.

Klassendiagramm
Komponentendiagramm
Kompositionsstrukturdiagramm
Deployment-Diagramm
Objektdiagramm
Paketdiagramm
Profildiagramm 421

8.2.1 Klassendiagramm

Dieser Abschnitt enthält die folgenden Kapitel zu Aufgaben und Begriffen im Zusammenhang mit Klassendiagrammen:

- Anpassen von Klassendiagrammen 394
- Außerkraftsetzen von Basisklassenoperationen und Implementieren von Schnittstellenoperationen 401
- Erstellen von Getter / Setter-Methoden 402
- Ball-and-socket Notation 404
- Hinzufügen von Ausnahmeereignissen zu Methoden einer Klasse 405
- Hinzufügen von Signalempfängern zu einer Klasse 406
- Generieren von Klassendiagrammen 407

Eine grundlegende Einführung in Klassendiagramme finden Sie unter Klassendiagramme 25 im Abschnitt "Tutorial" diese Dokumentation.

8.2.1.1 Anpassen von Klassendiagrammen

Erweitern / Ausblenden von Klassenbereichen in einem UML-Diagramm

Es gibt verschiedene Methoden, um die verschiedenen Bereiche von Klassendiagrammen zu erweitern.

 © 2016-2022 Altova GmbH

- Klicken Sie auf das **+** bzw. das **-** Symbol der gerade aktiven Klasse, um den jeweiligen Bereich zu erweitern/auszublenden.
- Ziehen Sie ein Rechteck (über dem Diagrammhintergrund) auf, um **mehrere** Klassen zu markieren und klicken Sie anschließend die Schaltfläche "Erweitern/Ausblenden". Sie können mehrere Klassen auch mit Hilfe von **Strg+Klick** auswählen.
- Drücken Sie **Strg + A**, um **alle Klassen** auszuwählen. Klicken Sie anschließend in einer der Klassen auf die Schaltfläche "Erweitern/Reduzieren", um die entsprechenden Bereiche zu erweitern bzw. zu reduzieren.

Erweitern/Reduzieren von Klassenbereichen in der Modellstruktur

In der Modellstruktur sind Klassen Unterelemente von Paketen und Sie können entweder die Pakete oder die Klassen erweitern bzw. reduzieren.

- Klicken Sie auf das Paket / die Klasse um es/sie zu **erweitern** und:
  - Drücken Sie die *****-Taste, um das aktuelle Paket/die aktuelle Klasse und alle Subelemente zu erweitern
  - Drücken Sie die **+**-Taste um das aktuelle Paket/die aktuelle Klasse zu öffnen.

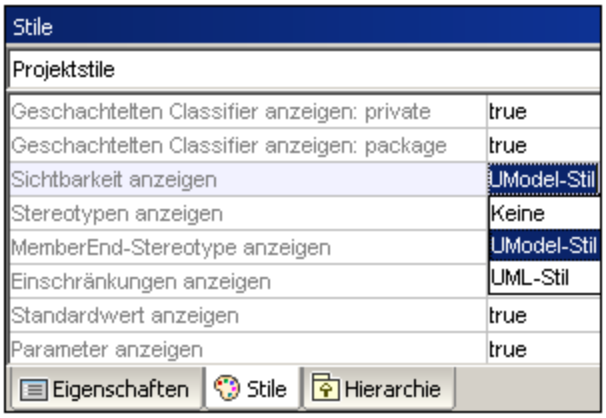
Um das Paket/die Klassen zu **reduzieren**, drücken Sie die **-** Taste.

Sie können dazu die Standardtasten der Tastatur oder die Tasten des Ziffernblocks verwenden.

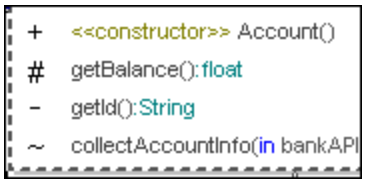
Ändern der Symbole für die Sichtbarkeit

Wenn Sie auf das **Sichtbarkeitssymbol** links von einer Operation oder Eigenschaft klicken, wird eine Dropdown-Liste geöffnet, in der Sie den Sichtbarkeitsstatus ändern können. Sie können auch ändern, welche Art von Sichtbarkeitssymbol angezeigt werden soll.

- Klicken Sie im Diagrammfenster auf eine Klasse, anschließend auf das Register **Stile** und scrollen Sie in der Liste hinunter bis zum Eintrag **Sichtbarkeit anzeigen**.



Sie können wählen zwischen dem oben gezeigten UModel Typ und den unten gezeigten UML-konformen Symbolen.

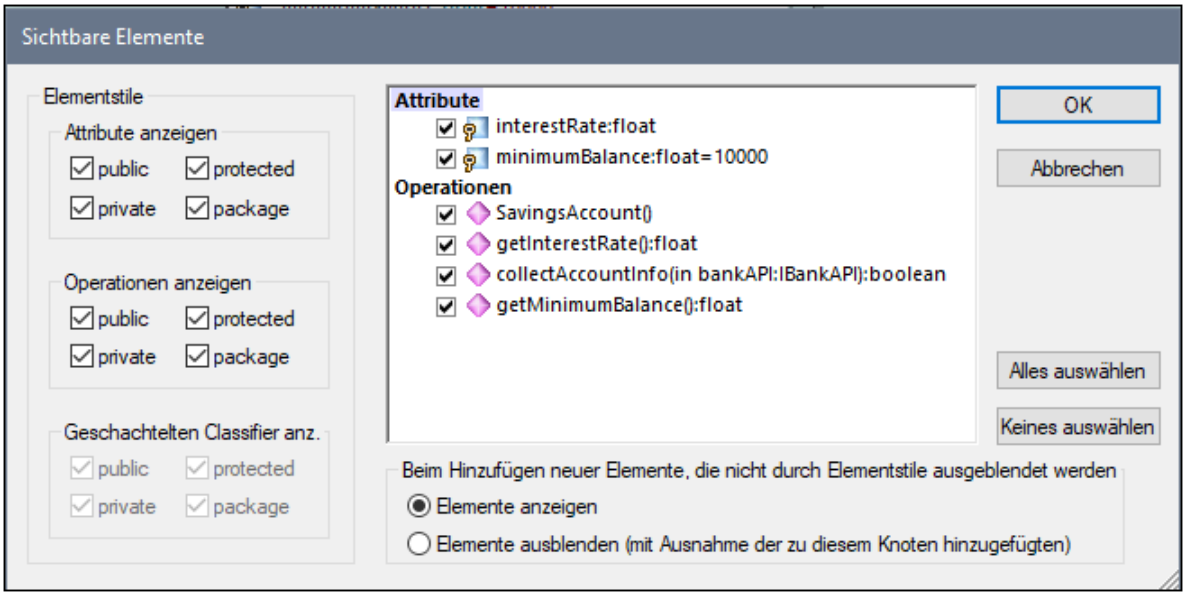


Ein-/Ausblenden von Knoteninhalt (Klassenattribute, Operationen, Slots)

Sie können bestimmte Mitglieder einer Klasse wie Attribute oder Operationen in Klassendiagrammen ein- oder ausblenden. Sie können nicht nur einzelne Mitglieder, sondern auch mehrere Mitglieder desselben Typs nach ihrer Sichtbarkeit ein- oder ausblenden. So können Sie z.B. nur diejenigen Klassenattribute ausblenden, die eine private Sichtbarkeit haben. Das Ein- oder Ausblenden wird auch für Objekt-Slots (`Instanzspezifikationen`) in Objektdiagrammen unterstützt.

So blenden Sie Klassenmitglieder oder Objekt-Slots ein oder aus:

1. Klicken Sie mit der rechten Maustaste auf eine Klasse (z.B. `SavingsAccount` aus dem Beispielprojekt **Bank_MultiLanguage.ump**) und wählen Sie im Kontextmenü den Befehl **Knoteninhalt ein-/ausblenden**.
2. Aktivieren oder deaktivieren Sie das Kontrollkästchen neben den Mitgliedern, die ein- bzw. ausgeblendet werden sollen.

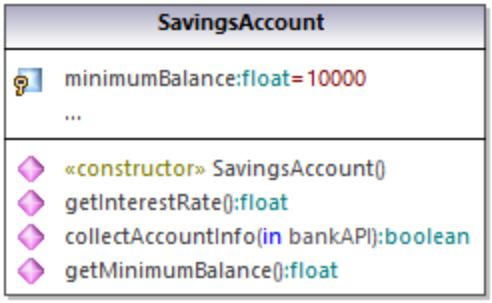


Um mehrere Mitglieder auf Basis ihrer Sichtbarkeit anzuzeigen, verwenden Sie die Kontrollkästchen in der Gruppe **Elementstile**. Wenn Sie z.B. in der Gruppe **Attribute anzeigen** das Kontrollkästchen **protected** deaktivieren, werden alle geschützten Attribute der Klasse ausgeblendet.

Anmerkung: Eigenschaftswerte von ausgeblendeten Elementen werden ebenfalls ausgeblendet, wenn Sie die Option "Ausblenden" auswählen.

Nachdem Sie die Einstellungen mit **OK** bestätigt und das Dialogfeld geschlossen haben, werden alle ausgeblendeten Mitglieder im Diagramm durch das Auslassungssymbol `...` ersetzt. Um das Dialogfeld wieder zu öffnen, doppelklicken Sie auf das Auslassungszeichen.

 © 2016-2022 Altova GmbH



Mit Hilfe der Option **Beim Hinzufügen neuer Elemente, die nicht durch Elementstile ausgeblendet werden** können Sie festlegen, was sichtbar gemacht werden soll, wenn neue Elemente zur Klasse hinzugefügt werden. Dies gilt nicht nur für Elemente, die im Diagramm oder der Modell-Struktur manuell hinzugefügt wurden, sondern auch für solche, die während des Code Engineering automatisch hinzugefügt wurden. Für diese Option gibt es die folgenden gültigen Werte:

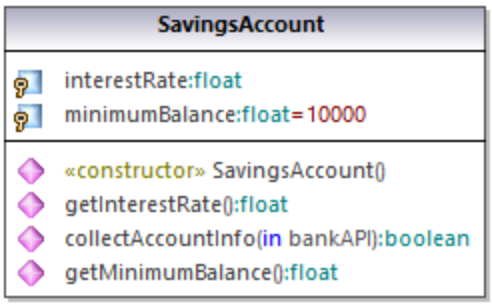
Elemente anzeigen	Wenn ein neues Mitglied zur Klasse hinzugefügt wird, wird dieses im Diagramm angezeigt. Wenn das Element jedoch aufgrund einer der Optionen unter "Elementstile" ausgeblendet werden muss, wird es ausgeblendet.
Elemente ausblenden (mit Ausnahme der zu diesem Knoten hinzugefügten)	Hier bezieht sich "Knoten" auf die aktuelle Instanz der Klasse im Diagramm. (Bedenken Sie, dass dieselbe Klasse mehrmals zum selben Diagramm hinzugefügt werden kann, siehe Umbenennen, Verschieben und Kopieren von Elementen 109.) Wenn im Diagramm zwei oder mehr Instanzen derselben Klasse vorhanden sind und ein neues Mitglied zu *dieser Instanz* der Klasse hinzugefügt wird, wird das Mitglied in allen Instanzen der Klasse ausgeblendet und nur für die aktuelle Instanz angezeigt.

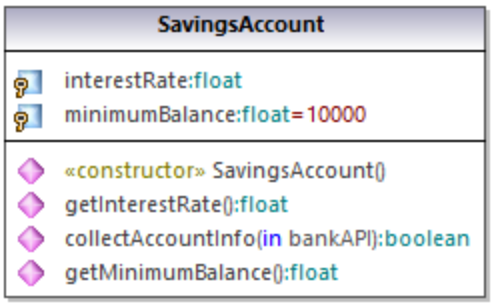
Ein Beispiel für die Verwendung der obigen Optionen finden Sie im Beispielprojekt **Bank_MultiLanguage.ump**. Suchen Sie darin nach dem Klassendiagramm "Hierarchy of Account".

Erstellen Sie als nächstes folgendermaßen eine neue Instanz der Klasse `SavingsAccount`:

1. Rechtsklicken Sie im Diagramm auf die Klasse `SavingsAccount` und wählen Sie **Kopieren**.
2. Klicken Sie mit der rechten Maustaste auf einen leeren Bereich im selben Diagramm und wählen Sie im Kontextmenü den Befehl **Nur in Diagramm einfügen**.

Es gibt nun im Diagramm zwei Instanzen der Klasse `SavingsAccount`.

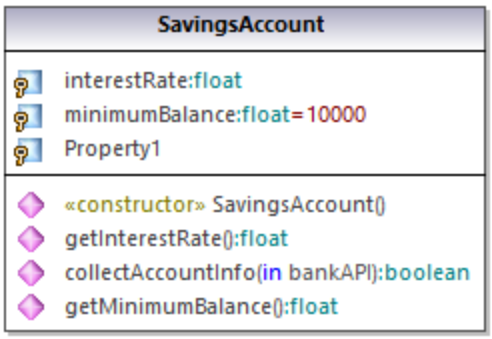


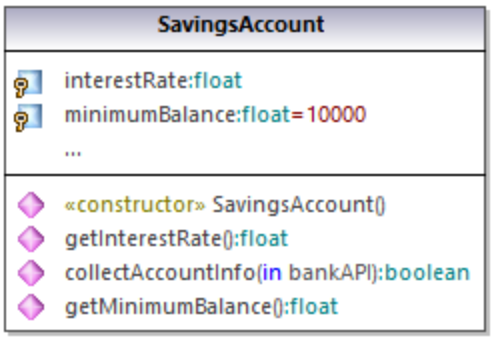


Definieren Sie als nächstes in jeder der Instanzen verschiedene Sichtbarkeitsoptionen:

1. Klicken Sie mit der rechten Maustaste auf die linke Instanz der Klasse, wählen Sie **Knoteninhalt ein-/ausblenden** und aktivieren Sie anschließend die Option **Elemente anzeigen**.
2. Klicken Sie mit der rechten Maustaste auf die rechte Instanz der Klasse, wählen Sie **Knoteninhalt ein-/ausblenden** und aktivieren Sie anschließend die Option **Elemente ausblenden (mit Ausnahme der zu diesem Knoten hinzugefügten)**.

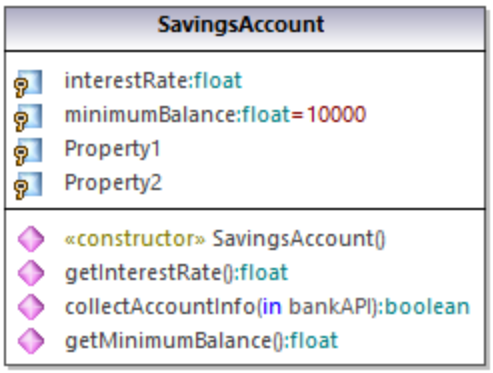
Fügen Sie als nächstes eine neue Eigenschaft zur linken Instanz hinzu (wählen Sie die Klasse aus und drücken Sie **F7**). Wie unten gezeigt, ist die neue Eigenschaft (`Eigenschaft1`) in der linken Instanz sichtbar, nicht aber in der rechten Instanz. Der Grund dafür ist, dass für die rechte Instanz der Klasse die Option **Elemente ausblenden (mit Ausnahme der zu diesem Knoten hinzugefügten)** aktiviert wurde.

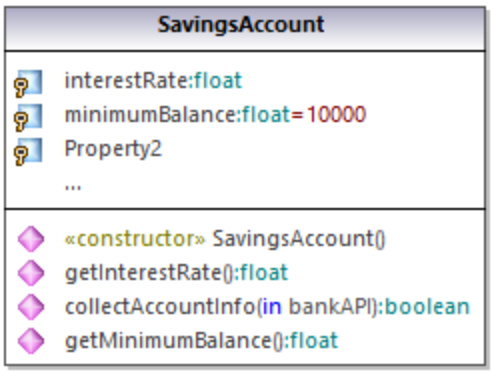




Fügen Sie abschließend eine neue Eigenschaft zur rechten Instanz der Klasse hinzu. Wie unten gezeigt, ist die neue Eigenschaft (`Eigenschaft2`) in beiden Instanzen sichtbar. Der Grund dafür ist, dass neue Elemente laut Konfiguration für die linke Instanz angezeigt werden sollen, während die rechte Instanz die *aktuelle Instanz* ist, zu der die Eigenschaft hinzugefügt wird, weswegen die neue Eigenschaft auf jeden Fall angezeigt wird.

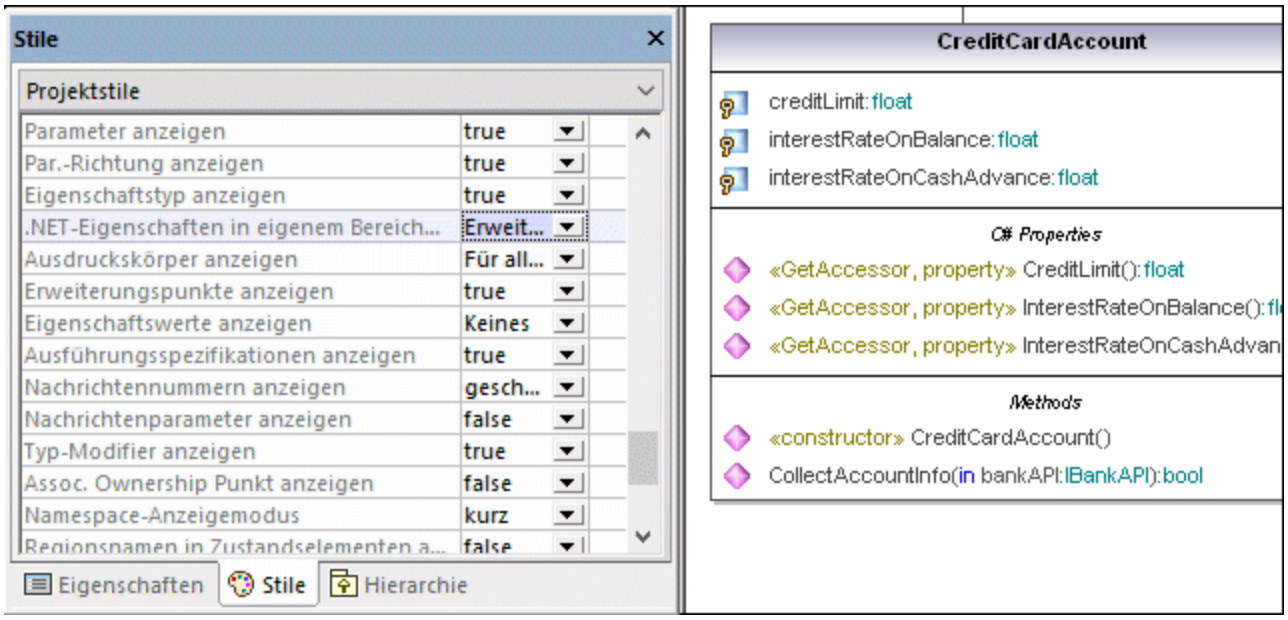
 © 2016-2022 Altova GmbH





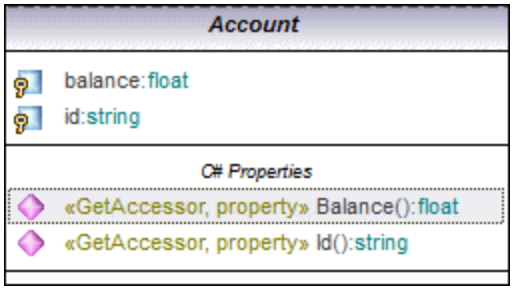
Ein- /Ausblenden von .NET-Bereichen

Um .NET-Eigenschaften in ihrem eigenen Bereich anzuzeigen, aktivieren Sie auf dem Register **Stile** die Option ".NET-Eigenschaften in eigenem Bereich anzeigen".



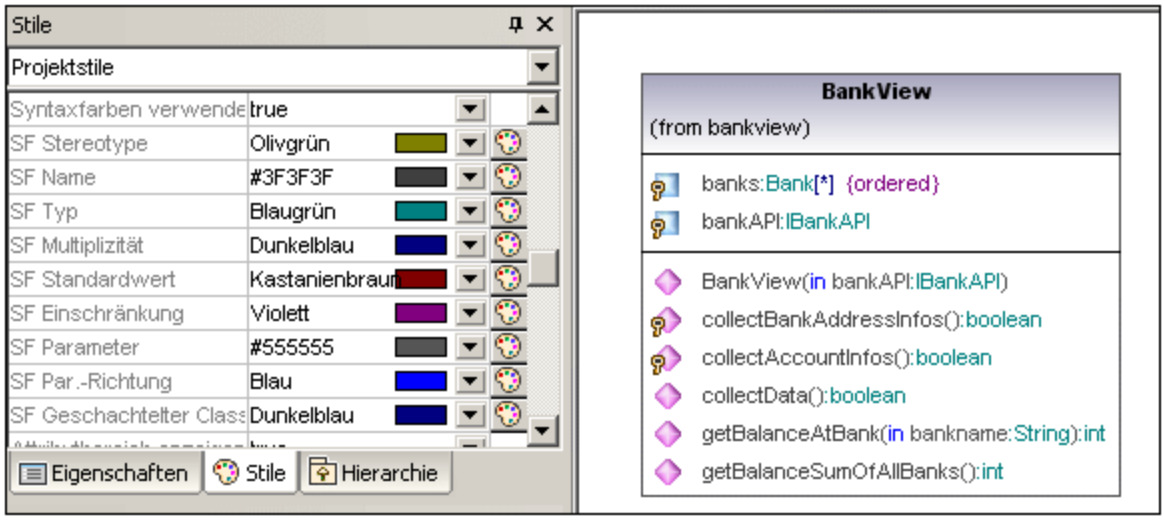
Anzeige von .NET-Eigenschaften als Assoziationen

Um .NET-Eigenschaften als Assoziationen anzuzeigen, klicken Sie mit der rechten Maustaste, wie unten gezeigt, auf eine C#-Eigenschaft und wählen Sie den Befehl **Anzeigen | Alle .NET-Eigenschaften als Assoziationen**.



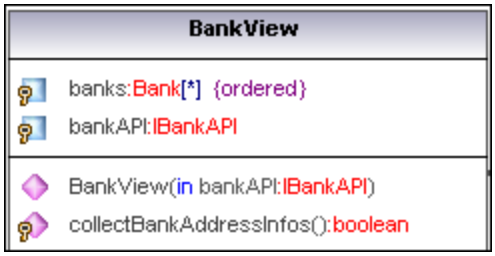
Ändern der Syntaxfarbe von Operationen/Eigenschaften

Die Syntaxfärbung ist in UModel automatisch aktiviert. Sie können die Syntaxfarben allerdings nach Ihren eigenen Wünschen anpassen. In der Abbildung unten sehen Sie die Standardeinstellungen.



So ändern Sie die Standardeinstellung für die Syntaxfarben (siehe unten):

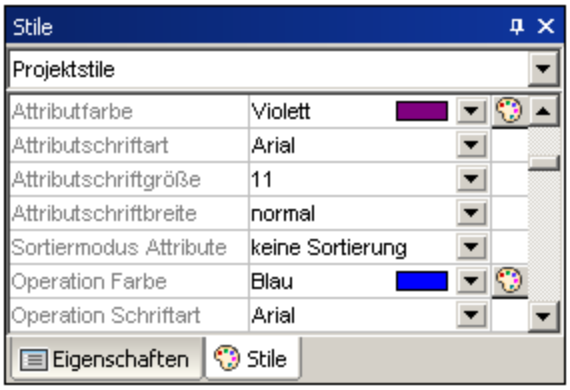
1. Wechseln Sie zum Register **Stile** und scrollen Sie zu den vordefinierten **SF**-Einträgen.
2. Ändern Sie einen der SF-Farbeinträge, z.B. "SF-Typ" zu "rot".



 © 2016-2022 Altova GmbH

So deaktivieren Sie die Syntaxfärbung:

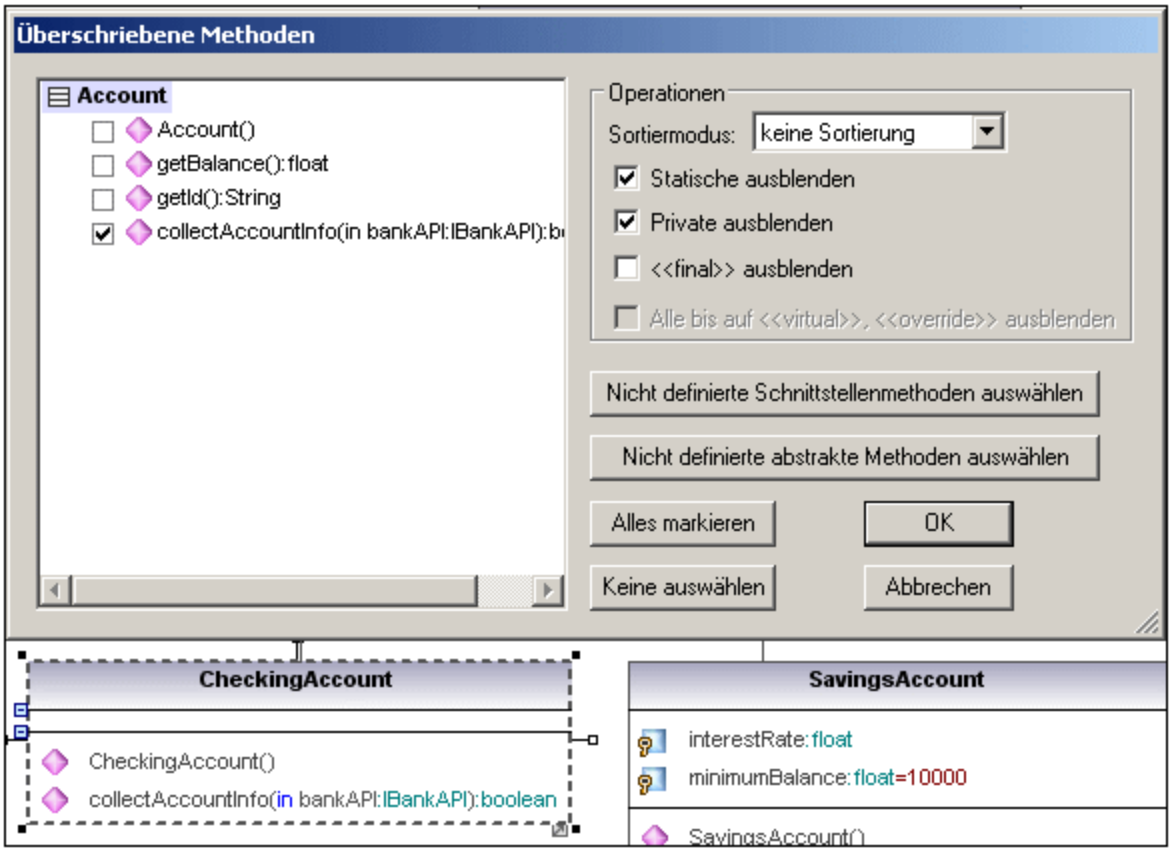
1. Wechseln Sie zum Register **Stile** und ändern Sie den Eintrag **Syntaxfarben verwenden** in **false**.
2. Über die Einträge **Attributfarbe** oder **Operation Farbe** auf dem Register **Stile** können Sie diese Einstellungen in der Klasse ändern.



8.2.1.2 Außerkraftsetzen von Basisklassenoperationen und Implementieren von Schnittstellenoperationen

Sie haben in UModel die Möglichkeit, die Basisklassenoperationen/Schnittstellen vor der Code Engineering-Phase außer Kraft zu setzen oder Schnittstellenoperationen einer Klasse zu implementieren. Die kann über die Modell-Struktur, das Register "Favoriten" oder in Klassendiagrammen durchgeführt werden.

1. Rechtsklicken Sie auf eine der abgeleiteten Klassen im Klassendiagramm, z.B. auf CheckingAccount und wählen Sie den Befehl **Operationen überschreiben/implementieren**.
   Daraufhin wird das unten gezeigte Dialogfeld geöffnet.



2. Wählen sie die Operationen aus, die Sie außer Kraft setzen möchten und bestätigen Sie mit OK. Mit Hilfe der Schaltflächen "Nicht definierte...auswählen" werden die jeweiligen Methodenarten im Fenster auf der linken Seite ausgewählt.

Anmerkung:
Beim Öffnen des Dialogfelds werden Operationen von Basisklassen und implementierten Schnittstellen, die dieselbe Signatur wie bestehende Operationen haben, automatisch ausgewählt, d.h. aktiviert.

8.2.1.3 Erstellen von Getter / Setter-Methoden

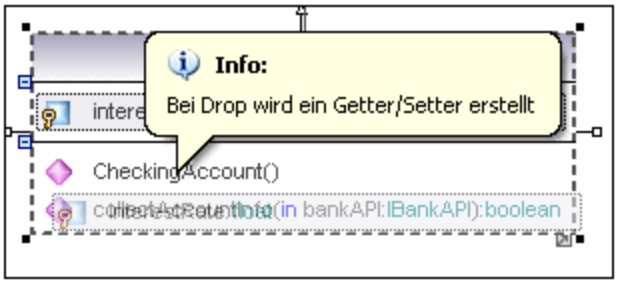
Während des Modellierens müssen oft get/set-Methoden für bestehende Attribute erstellt werden. UModel bietet für diesen Zweck zwei separate Methoden:

- Verschieben eines Attributs mit Hilfe von Drag and Drop in den Operation-Bereich
- Verwendung des Kontextmenüs zum Öffnen eines Dialogfelds, in dem Sie die get/set-Methoden verwalten können

So erstellen Sie Getter/Setter-Methoden mittels Drag and Drop:

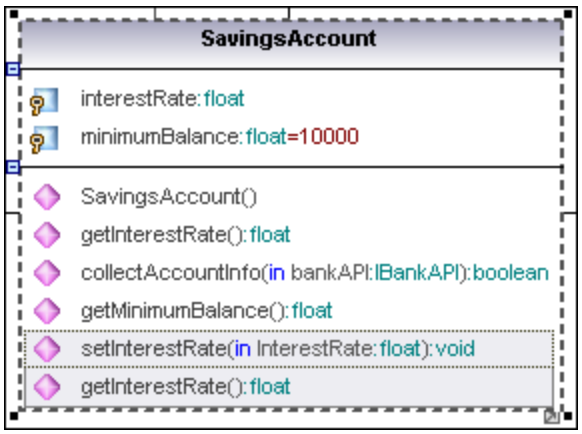
1. Ziehen Sie ein Attribut aus dem Attribut-Bereich in den Operations-Bereich.

 © 2016-2022 Altova GmbH



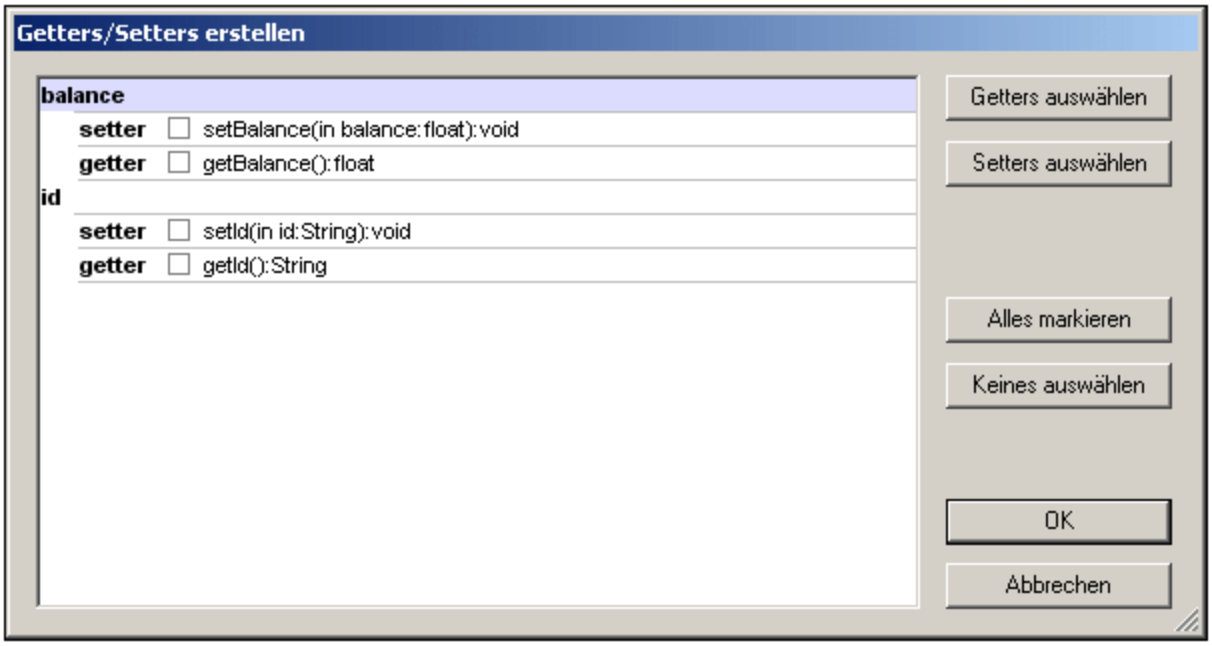
Daraufhin wird ein Popup-Fenster angezeigt, in dem Sie auswählen können, welche Art von get/set-Methode erstellt werden soll.

Wenn Sie den ersten Eintrag auswählen, wird eine get- und set-Methode für interestRate:float erstellt.



So erstellen Sie Getter/Setter-Methoden über das Kontextmenü:

1. Rechtsklicken Sie auf den Namen einer Klasse, z.B. SavingsAccount und wählen Sie im Kontextmenü die Option **Getter/Setter-Operationen erstellen**.



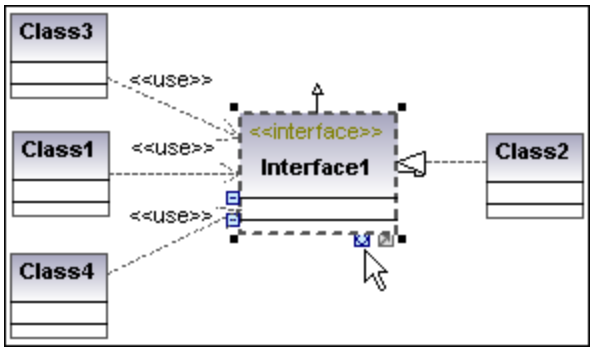
Daraufhin erscheint das Dialogfeld "Getters/Setters erstellen", in dem alle in der derzeit aktiven Klasse vorhandenen Attribute angezeigt werden.

2. Verwenden Sie die Schaltflächen, um die Einträge als Gruppe auszuwählen oder klicken Sie auf die Kontrollkästchen der einzelnen Methoden.

Anmerkung:
Sie können auch auf ein einzelnes Attribut rechtsklicken und auf dieselbe Art eine Operation dafür erstellen.

8.2.1.4 Ball-and-socket Notation

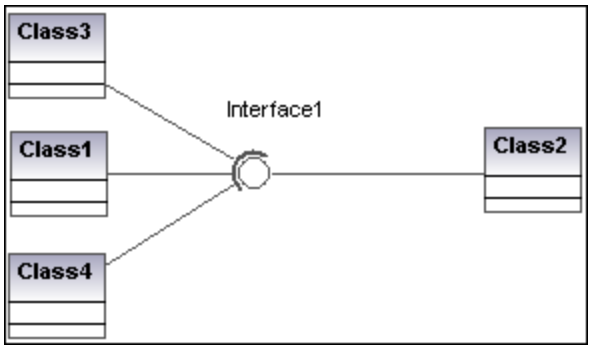
UModel unterstützt die Ball-and-socket Notation von UML-Klassen, die eine Schnittstelle benötigen, ein "Socket" (Buchse) und den Schnittstellennamen anzeigen, während Klassen, die eine Schnittstelle implementieren den "ball" (Stecker) anzeigen.



In den Abbildungen oben realisiert Class2 Interface1, welche von den Klassen 1, 3 und 4 verwendet wird. Die Verwendungssymbole dienten zum Erstellen der Verwendungsbeziehung zwischen den Klassen und der Schnittstelle.

So wechseln Sie zwischen der Standard- und der Ball-and-socket-Ansicht:

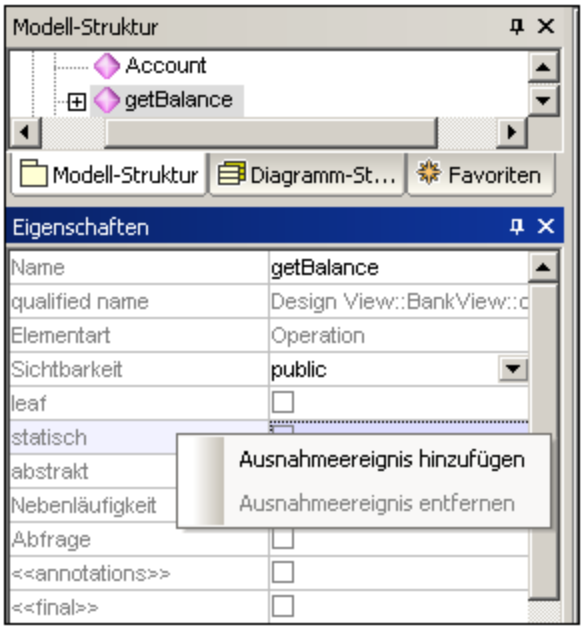
- Klicken Sie auf das Symbol "Darstellung wechseln" am unteren Rand des Schnittstellenelements.



8.2.1.5 Hinzufügen von Ausnahmeereignissen zu Methoden einer Klasse

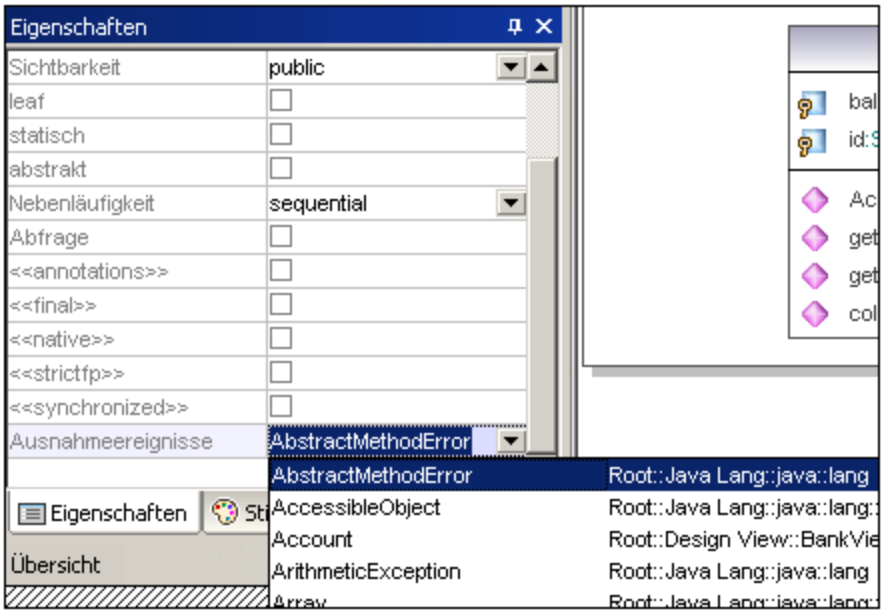
So fügen Sie Ausnahmeereignisse zu Methoden einer Klasse hinzu:

1. Klicken Sie im Fenster "Modellstruktur" auf die Methode der Klasse, zu der Sie das Ausnahmeereignis hinzufügen möchten, z.B. in der Klasse "Account" auf getBalance.
2. Rechtsklicken Sie in das Fenster "Eigenschaften" und wählen Sie im Popup-Menü den Befehl **Ausnahmeereignis hinzufügen**.



Daraufhin wird das Feld "Ausnahmeereignis" zum Fenster "Eigenschaften" hinzugefügt und der erste Eintrag im Popup-Menü wird automatisch markiert.

3. Wählen Sie einen Eintrag im Popup-Menü aus oder geben Sie Ihren eigenen in das Feld ein.



8.2.1.6 Hinzufügen von Signalempfängern zu einer Klasse

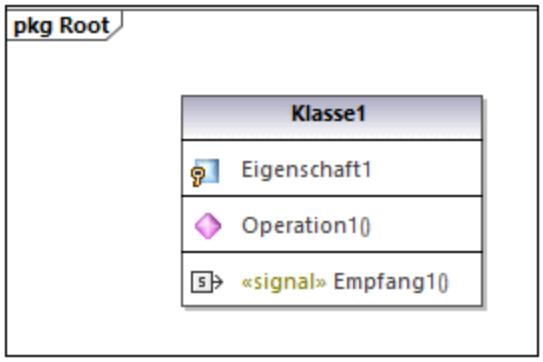
Neben Operationen und Eigenschaften können Sie auch Signalempfängerelemente zu einer Klasse hinzufügen.

So fügen Sie einen Signalempfänger zu einer Klasse hinzu:

- Klicken Sie mit der rechten Maustaste in das Diagramm und wählen Sie im Kontextmenü den Befehl **Neu | Empfang**.

Signalempfänger werden ähnlich Eigenschaften und Operationen in einem eigenen Bereich im Klassendiagramm angezeigt, z.B:

 © 2016-2022 Altova GmbH

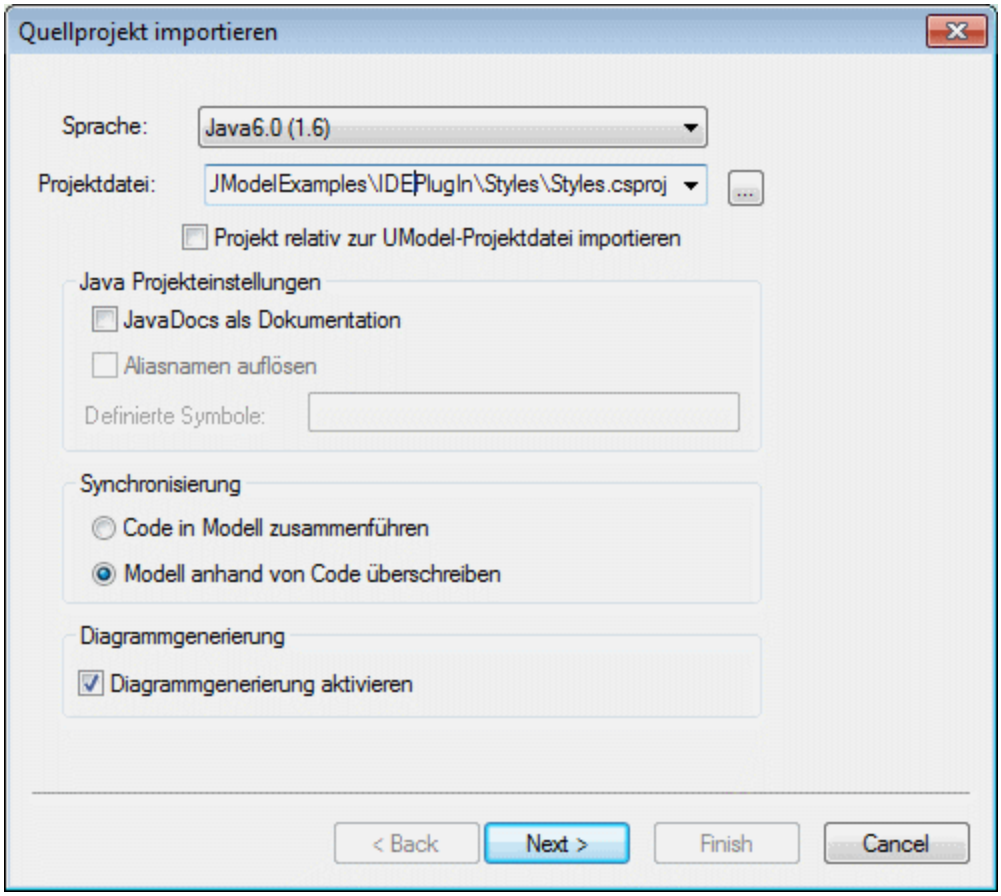


Signalempfänger haben dieselben Stile wie Operationen. D.h., immer wenn Sie den Stil von Operationen ändern, wirken sich diese Änderungen auch auf Signalempfänger aus. Nähere Informationen dazu finden Sie unter Ändern des Stils von Elementen 119.

8.2.1.7 Generieren von Klassendiagrammen

Anstatt Klassendiagramme direkt in UModel zu erstellen, können Sie diese beim Import von Quellcode oder Binärdateien in UModel-Projekte automatisch generieren (siehe Importieren von Quellcode in Projekte 194 und Importieren von Java-, C#- und VB.NET-Binärdateien 206). Wenn Sie den Assistenten verwenden, achten Sie auf folgende Dinge:

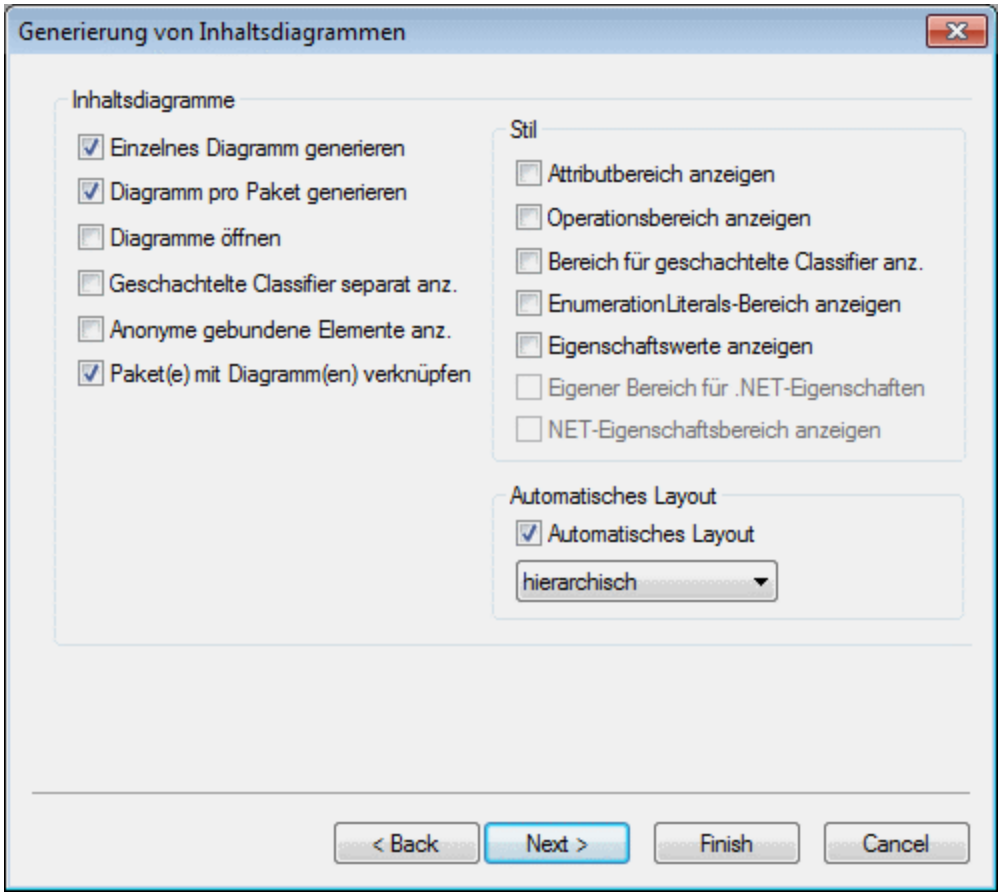
1) Im Dialogfeld "Quellprojekt importieren", "Binärtypen importieren" bzw. "Quellverzeichnis importieren" muss das Kontrollkästchen **Diagrammgenerierung aktivieren** aktiviert sein.



Dialogfeld "Quellprojekt importieren"

2) Im Dialogfeld "Generierung von Inhaltsdiagrammen" müssen die Optionen **Einzelnes Diagramm generieren** und/oder **Diagramm pro Paket generieren** aktiviert sein.

 © 2016-2022 Altova GmbH



Dialogfeld "Generierung von Inhaltsdiagrammen"

Nach Fertigstellung des Imports stehen die generierten Klassendiagramme in der Diagramm-Struktur unter "Klassendiagramme" zur Verfügung.

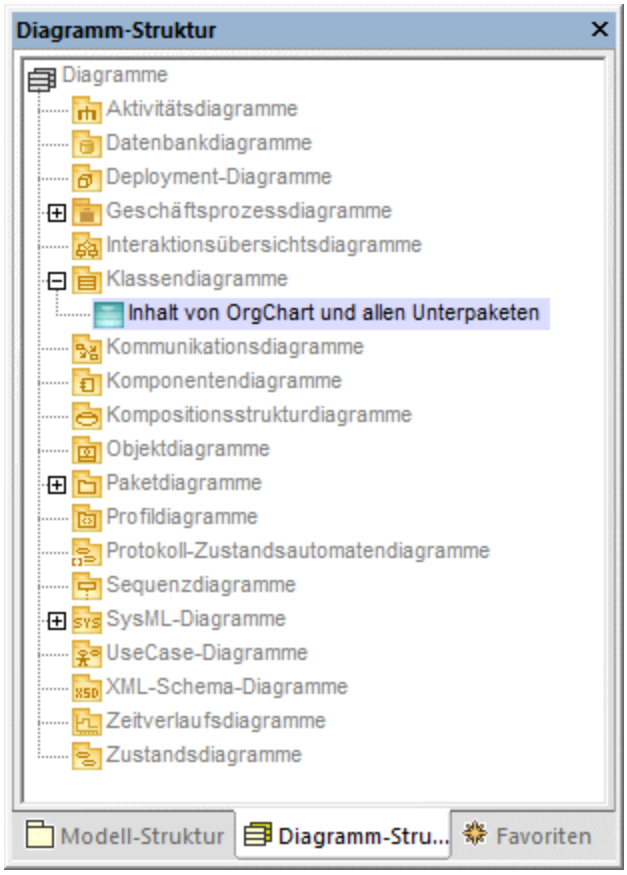


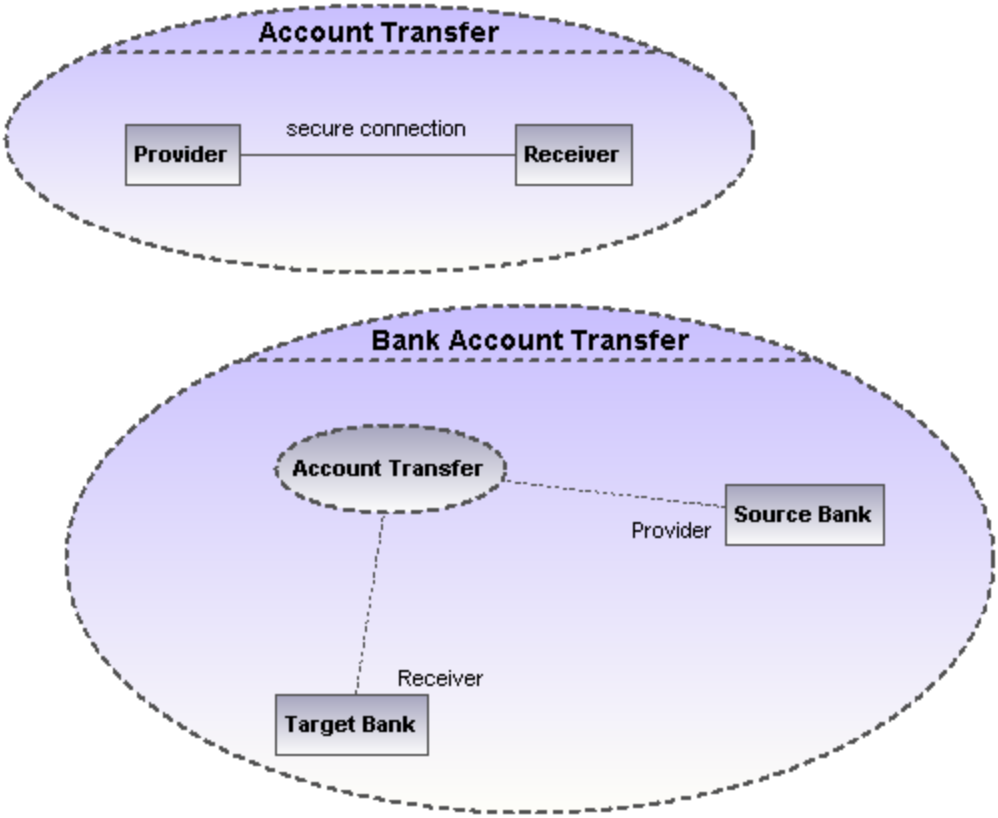
Diagramm-Struktur

8.2.2 Kompositionsstrukturdiagramm

Altova Website: UML-Kompositionsstrukturdiagramme

In UML 2.0 wurde das Kompositionsstrukturdiagramm hinzugefügt. Es dient dazu, die interne Struktur einschließlich Bereichen, Ports und Konnektoren eines strukturierten Classifier oder einer Kollaboration anzuzeigen.

 © 2016-2022 Altova GmbH



8.2.2.1 Einfügen von Elementen eines Kompositionsstrukturdiagramms

Über die Schaltflächen der Symbolleiste:

1. Klicken Sie in der Symbolleiste auf das jeweilige Kompositionsstrukturdiagramm-Symbol.

2. Klicken Sie in das Kompositionsstrukturdiagramm, um das Element einzufügen. Um mehrere Elemente des ausgewählten Typs einzufügen, halten Sie die **Strg-Taste** gedrückt und klicken Sie in das Diagrammfenster.

Ziehen bestehender Elemente in das Kompositionsstrukturdiagramm

Die meisten Elemente, die in anderen Kompositionsstrukturdiagrammen vorkommen, können in ein bestehendes Kompositionsstrukturdiagramm eingefügt werden.

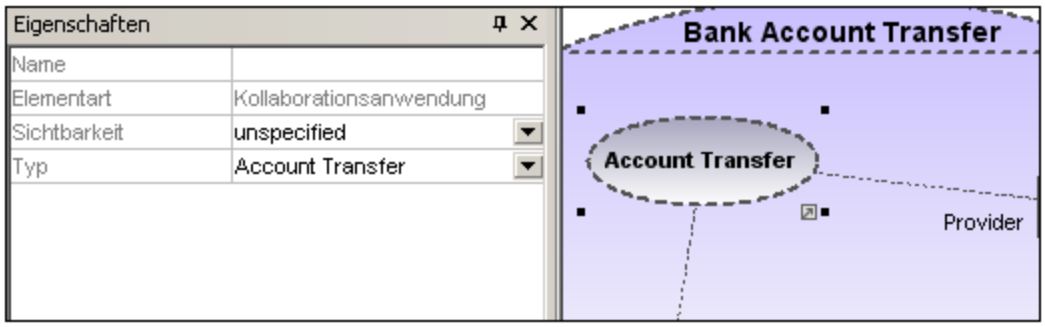
1. Suchen Sie das gewünschte Element auf dem Register "Modell-Struktur" (Sie können das Element über das Suchfunktionstextfeld oder mit Hilfe von Strg + F suchen).
2. Ziehen Sie das/die Element(e) in das Kompositionsstrukturdiagramm.

Kollaboration

Fügt ein Kollaborationselement ein. Dabei handelt es sich um eine Art von Classifier/Instanz, der/die mit anderen Instanzen kommuniziert, um das Verhalten des Systems zu erzeugen.

Kollaborationsanwendung

Fügt ein Kollaborationsanwendungselement ein. Dabei handelt es sich um eine bestimmte Kollaborationsanwendung, wobei bestimmte Klassen oder Instanzen die Rolle der Kollaboration spielen. Eine Kollaborationsanwendung wird als strichlierte Ellipse angezeigt, die den Namen der Instanz, einen Doppelpunkt und den Namen des Kollaborationstyps enthält.



Beim Erstellen von Abhängigkeiten zwischen Kollaborationsanwendungselementen muss das Feld "Typ" ausgefüllt sein, damit eine Rollenbindung erstellt werden kann und die Zielkollaboration muss mindestens einen Part/eine Rolle haben.

Part (Eigenschaft)

Fügt ein Part-Element ein, das eine Gruppe von einer oder mehreren Instanzen darstellt, die ein Classifier besitzt, der diese enthält. Ein Part kann zu Kollaborationen und Klassen hinzugefügt werden.

Port

Fügt ein Port-Element ein, das den Interaktionspunkt zwischen einem Classifier und seiner Umgebung definiert. Ein Port-Element kann zu einem Part-Elementen mit einem definierte Typ hinzugefügt werden.

Klasse

Fügt ein Klassenelement ein. Dabei handelt es sich um den eigentlichen Classifier, der in dieser bestimmten Verwendung der Kollaboration vorkommt.

Konnektor

Fügt ein Konnektorelement hinzu, das verwendet werden kann, um zwei oder mehr Instanzen eines Teils oder Ports zu verbinden. Der Konnektor definiert die Beziehung zwischen den Objekten und identifiziert die Kommunikation zwischen den Rollen.

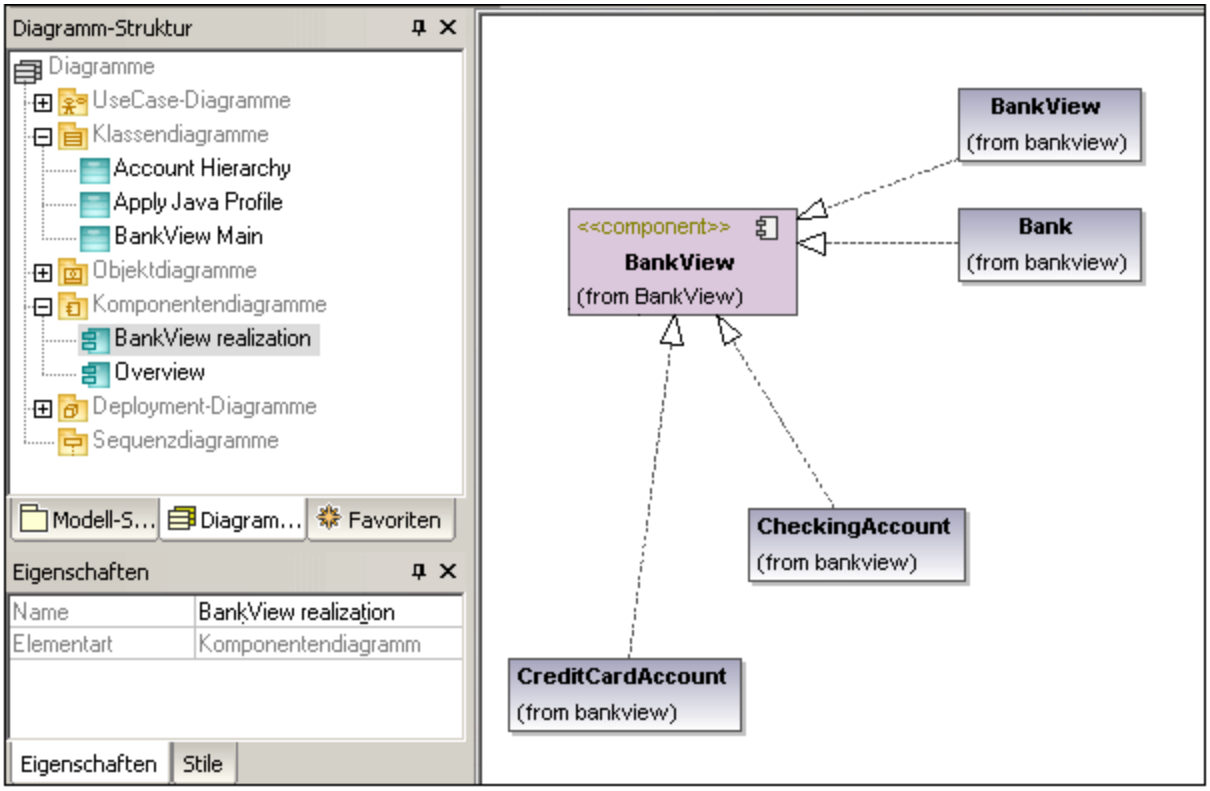
© 2016-2022 Altova GmbH

Abhängigkeit (Rollenbindung)

Fügt das Abhängigkeitselement ein. Dieses Element zeigt an, welches verbindbare Element des Classifiers oder der Operation welche Rolle in der Kollaboration spielt.

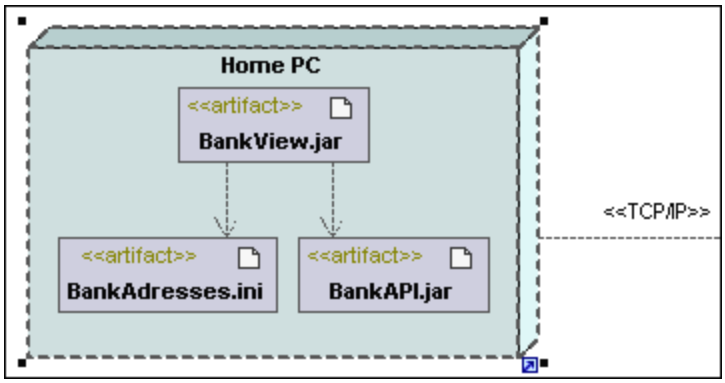
8.2.3 Komponentendiagramm

Eine Anleitung zum Hinzufügen von Komponentenelementen zum Diagramm finden Sie im Tutorial im Abschnitt Komponentendiagramme 48.



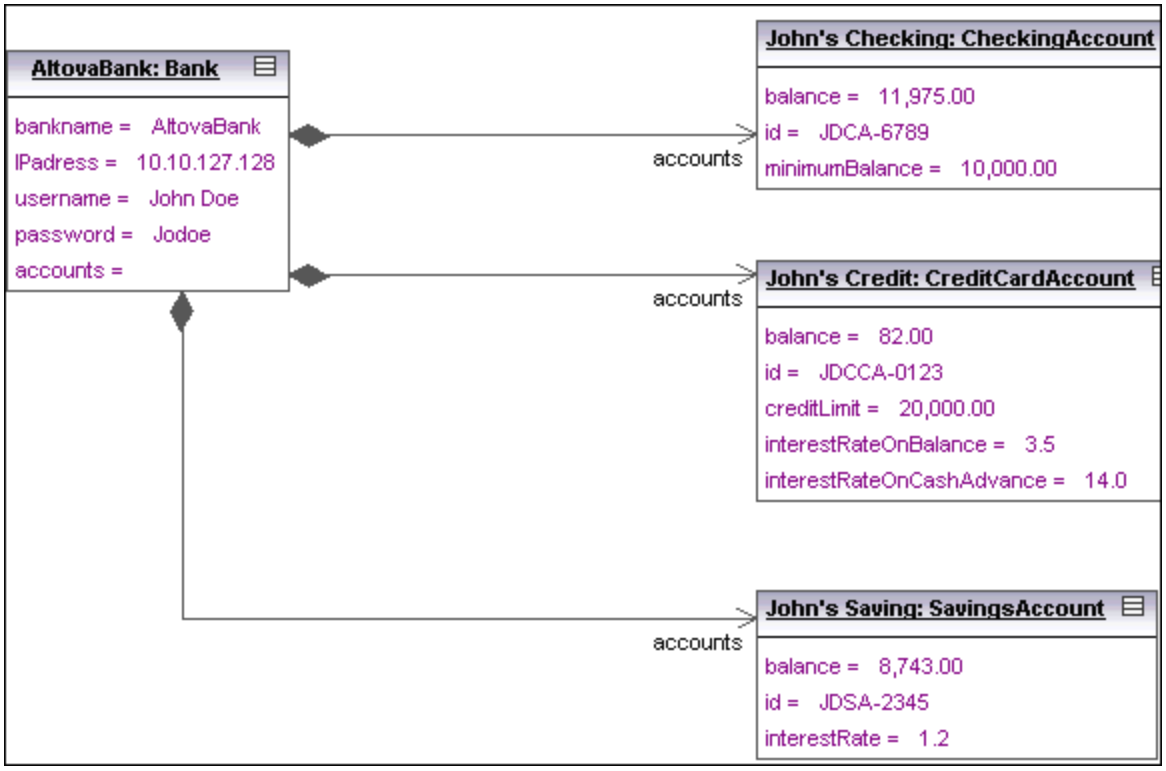
8.2.4 Deployment-Diagramm

Eine Anleitung zum Hinzufügen von Knoten und Artefakten zum Diagramm finden Sie im Tutorial im Abschnitt Deployment-Diagramme 54.



8.2.5 Objektdiagramm

Eine Anleitung zum Hinzufügen neuer Objekte/Instanzen zum Diagramm finden Sie im Tutorial im Abschnitt Objektdiagramme 40.

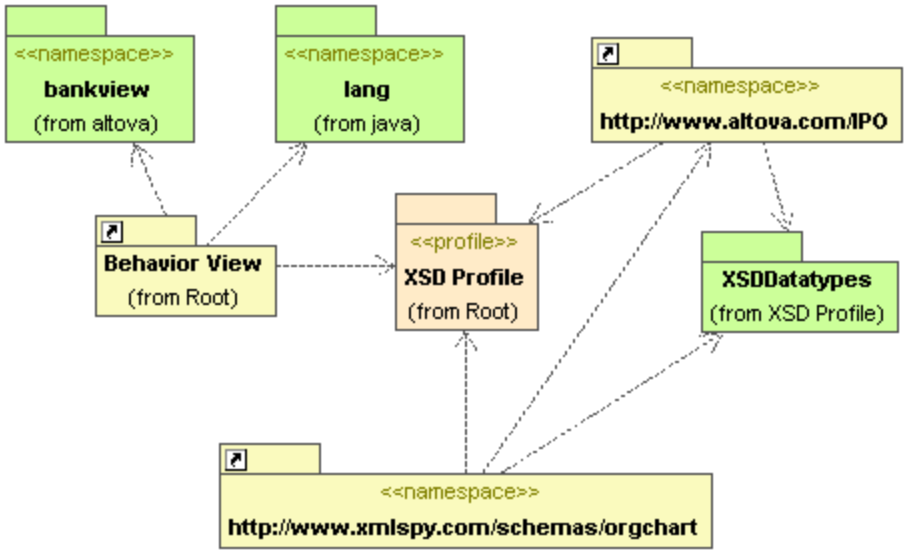


8.2.6 Paketdiagramm

In Paketdiagrammen werden die Struktur von Paketen und ihren Elementen sowie die entsprechenden Namespaces angezeigt. Zusätzlich dazu können Sie in UModel einen Hyperlink erstellen und zum jeweiligen Inhalt des Pakets navigieren.

 © 2016-2022 Altova GmbH

Pakete werden als Ordner dargestellt und können in jedem der UML-Diagramme verwendet werden. Sie werden jedoch hauptsächlich in Use Case- und Klassendiagrammen verwendet.



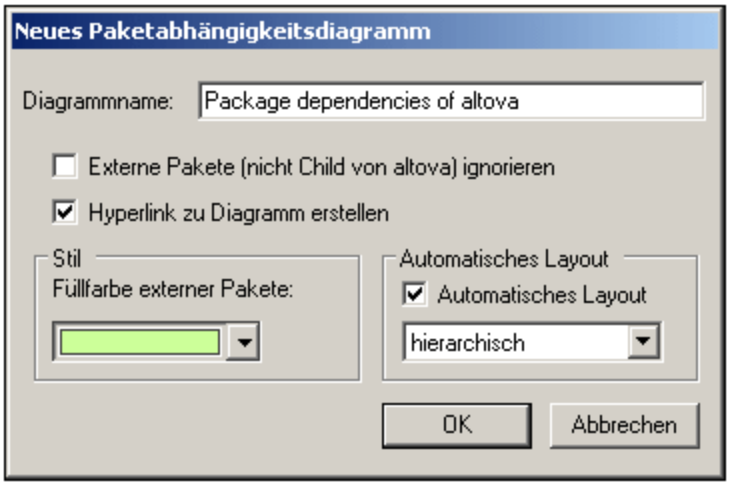
Automatische Generierung von Paketabhängigkeiten

UModel bietet die Möglichkeit, ein Paketabhängigkeitsdiagramm für jedes beliebige in der Modell-Struktur bereits existierende Paket zu generieren.

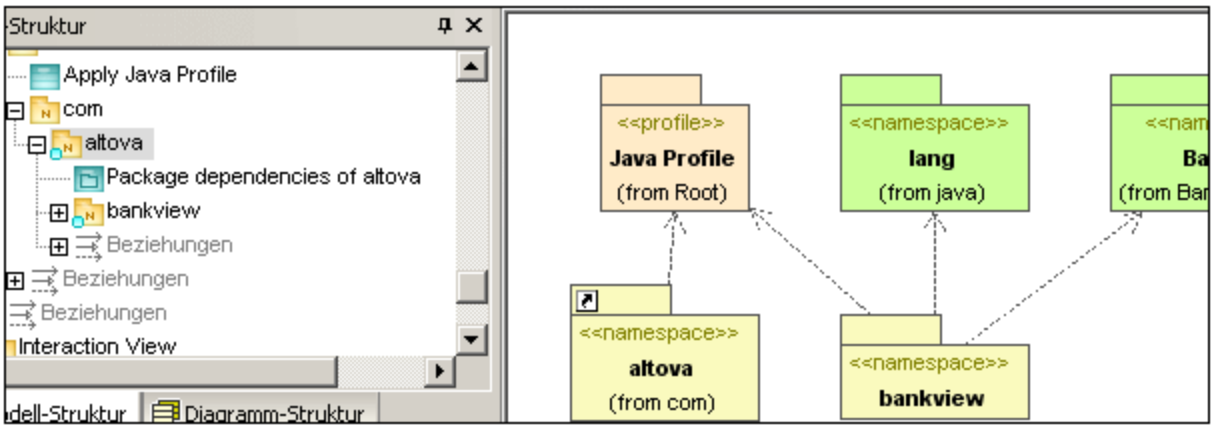
Abhängigkeitslinks zwischen Paketen werden erstellt, wenn Referenzen zwischen den Modellierungselementen dieser Pakete vorhanden sind, d.h. wenn es Abhängigkeiten zwischen Klassen, abgeleitete Klassen oder Attributen gibt, die Typen haben, die in einem anderen Paket definiert sind.

So generieren Sie ein Paketabhängigkeitsdiagramm:

1. Rechtsklicken Sie auf ein Paket in der Modell-Struktur, z.B. altova und wählen Sie den Befehl **In neuem Diagramm anzeigen | Paketabhängigkeiten...** .
   Daraufhin wird das Dialogfeld "Neues Paketabhängigkeitsdiagramm" geöffnet.



2. Wählen Sie die gewünschten Optionen aus und klicken Sie zur Bestätigung auf OK.



Es wird ein neues Diagramm generiert, in dem die Paketabhängigkeiten des Pakets "altova" angezeigt werden.

8.2.6.1 Einfügen von Paketdiagrammelementen

Verwendung der Symbolleisten-Schaltflächen

1. Klicken Sie in der Paketdiagramm-Symbolleiste auf das entsprechende Symbol.

2. Klicken Sie in das Diagramm, um das Element einzufügen. Um mehrere Elemente des ausgewählten Typs einzufügen, halten Sie die **Strg-Taste** gedrückt und klicken Sie in das Diagrammfenster.

 © 2016-2022 Altova GmbH

Ziehen vorhandener Elemente in das Paketdiagramm

Sie können Elemente aus anderen Diagrammen, z.B. Pakete, in ein Paketdiagramm einfügen.

1. Suchen Sie das gewünschte Element auf dem Register "Modell-Struktur" (über das Suchfunktionstextfeld oder durch Drücken der Tasten Strg + F).
2. Ziehen Sie das Element/die Elemente in das Diagramm.

Paket

Fügt das Paketelement in das Diagramm ein. Pakete dienen zum Gruppieren von Elementen und zur Bereitstellung eines Namespace für die gruppierten Elemente. Da es sich bei einem Paket um einen Namespace handelt, kann ein Paket einzelne Elemente oder alle Elemente anderer Pakete importieren. Pakete können auch mit anderen Paketen zusammengeführt werden.

Profil

Fügt das Profil-Element ein, einen bestimmten Pakettyp, der auf andere Pakete angewendet werden kann.

Das Profile-Paket dient zum Erweitern des UML-Metamodells. Das primäre Erweiterungskonstrukt ist das Stereotyp, welches selbst Teil des Profils ist. Profile müssen immer in Beziehung zu einem Referenz-Metamodell wie z.B. UML stehen und können nicht alleine stehen.

Abhängigkeit

Fügt das Abhängigkeits-Element ein, welches eine Bereitsteller/Client-Beziehung zwischen Modellierungselementen - in diesem Fall Paketen oder Profilen anzeigt.

Paketimport

Fügt eine <<import>> Beziehung ein, welche anzeigt, dass die Elemente des inkludierten Pakets in das inkludierende Paket importiert werden. Der Namespace des inkludierenden Pakets erhält Zugriff auf den inkludierten Namespace; der Namespace des inkludierten Pakets ist nicht betroffen.

Anmerkung: Elemente, die in einem Paket als "privat" definiert sind, können nicht zusammengeführt oder importiert werden.

Paketmerge

Fügt eine <<merge>> Beziehung ein, welche anzeigt, dass die Elemente des zusammengeführten (Quell)-Pakets einschließlich aller in das zusammengeführte (Quell)-Paket importierten Inhalte in das zusammenführende (Ziel)-Paket importiert werden.

Wenn dasselbe Element im Zielpaket bereits vorhanden ist, werden die Definitionen dieser Elemente um die Definitionen aus dem Zielpaket erweitert. Aktualisierte oder hinzugefügte Elemente werden durch eine Generalisierungsbeziehung zurück zum Quellpaket gekennzeichnet.
Anmerkung: Elemente, die in einem Paket als "privat" definiert sind, können nicht zusammengeführt oder importiert werden.

Profilzuweisung

Fügt eine Profilzuweisung ein, die anzeigt, welche Profile einem Paket zugewiesen wurden. Hierbei handelt es sich um eine Art des Paketimports, der festlegt, dass ein Profil auf ein Paket angewendet wird.

Das Profil erweitert das Paket, dem es zugewiesen wurde. Bei Zuweisung eines Profils mit Hilfe des Symbols "Profilzuweisung" stehen alle Stereotype, die Teil dieses Profils sind, auch dem Paket zur Verfügung.

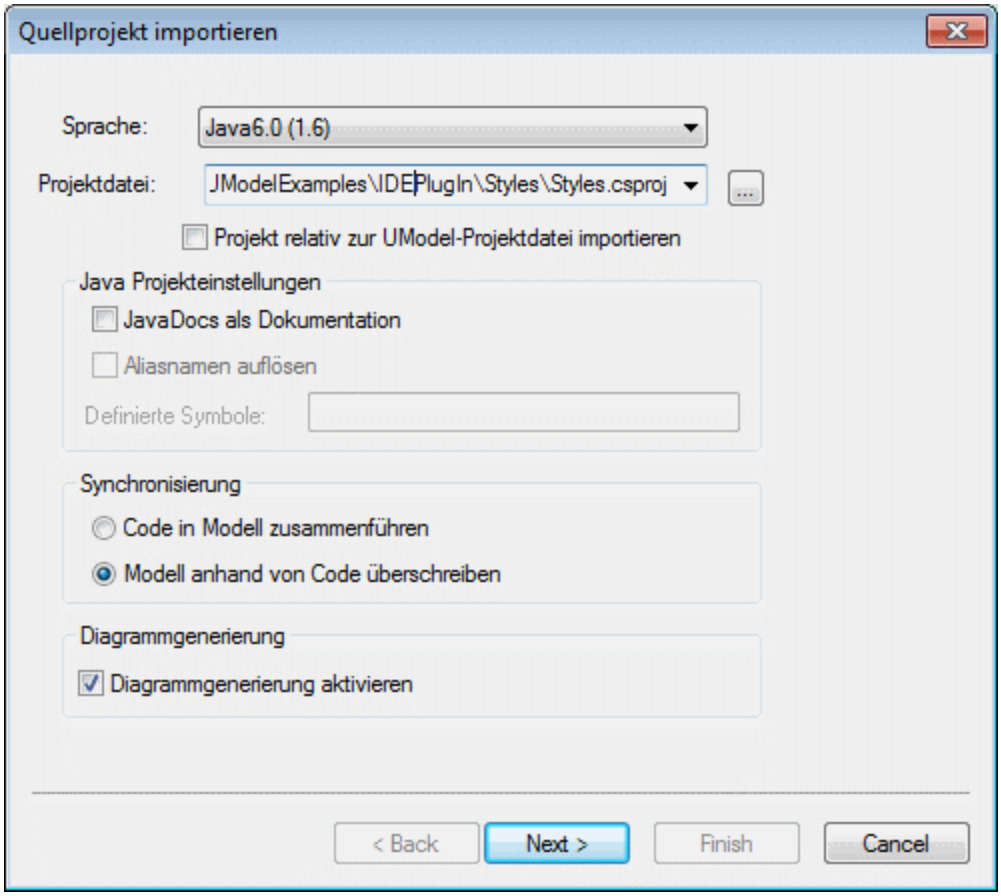
Profilnamen werden als strichlierte Pfeile vom Paket zum zugewiesenen Profil mit dem Schlüsselwort <<apply>> angezeigt.

8.2.6.2 Generieren von Paketdiagrammen

Sie können bei Import von Quellcode oder Binärdateien in ein UModel-Projekt (siehe Importieren von Quellcode in Projekte 194 und Importieren von Java-, C#- und VB-Binärdateien 206) Paketdiagramme generieren. Nehmen Sie im Importassistenten folgende Einstellungen vor:

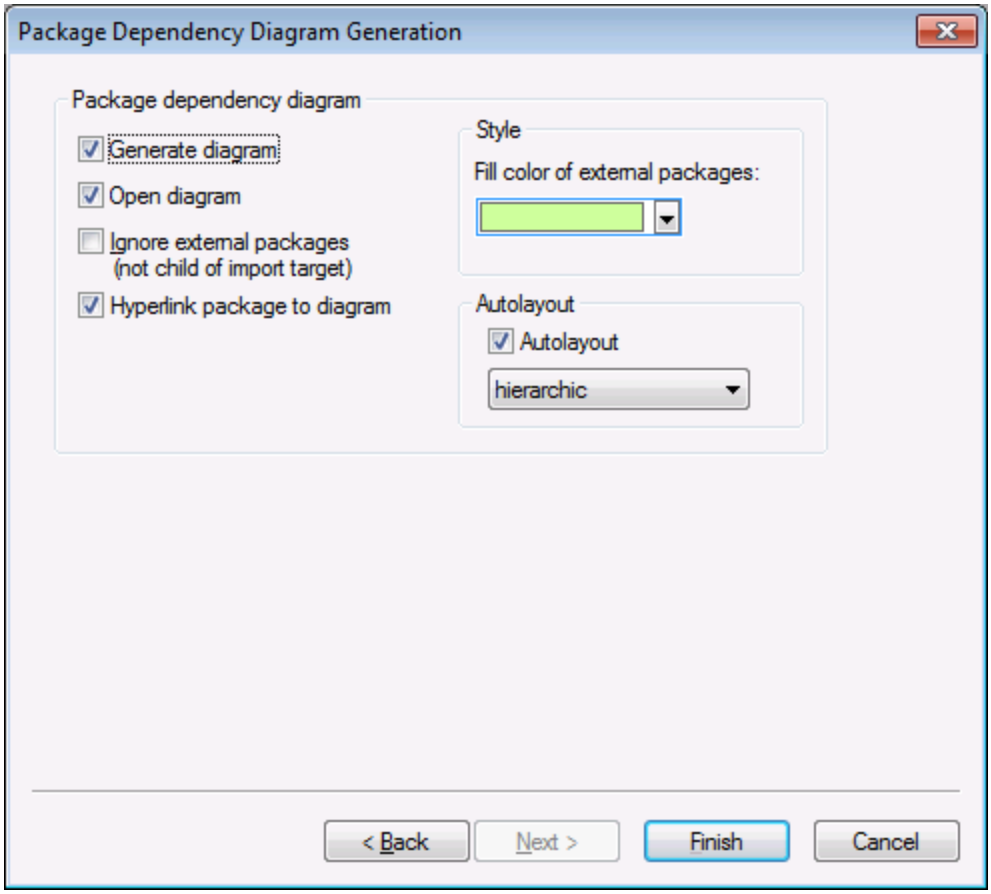
1) Im Dialogfeld "Quellprojekt importieren", "Binärtypen importieren" bzw. "Quellverzeichnis importieren" muss das Kontrollkästchen **Diagrammgenerierung aktivieren** aktiviert sein.

 © 2016-2022 Altova GmbH



Dialogfeld "Quellprojekt importieren"

2) Im Dialogfeld "Generierung von Paketabhängigkeitsdiagrammen" muss die Option **Diagramm generieren** aktiviert sein.



Dialogfeld "Generierung von Paketabhängigkeitsdiagrammen"

Nach Fertigstellung des Imports stehen die generierten Klassendiagramme in der Diagramm-Struktur unter "Klassendiagramme" zur Verfügung.

© 2016-2022 Altova GmbH

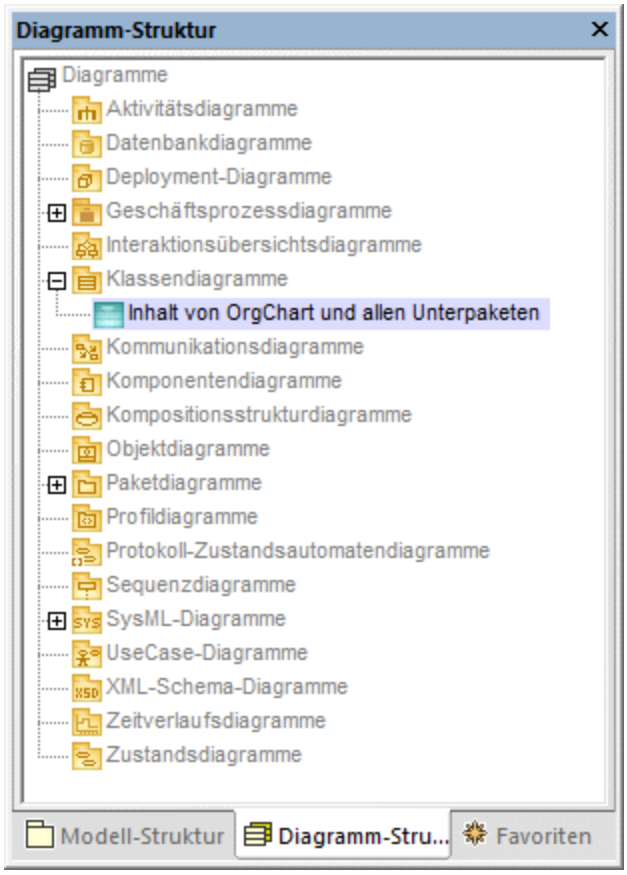


Diagramm-Struktur

8.2.7 Profildiagramm

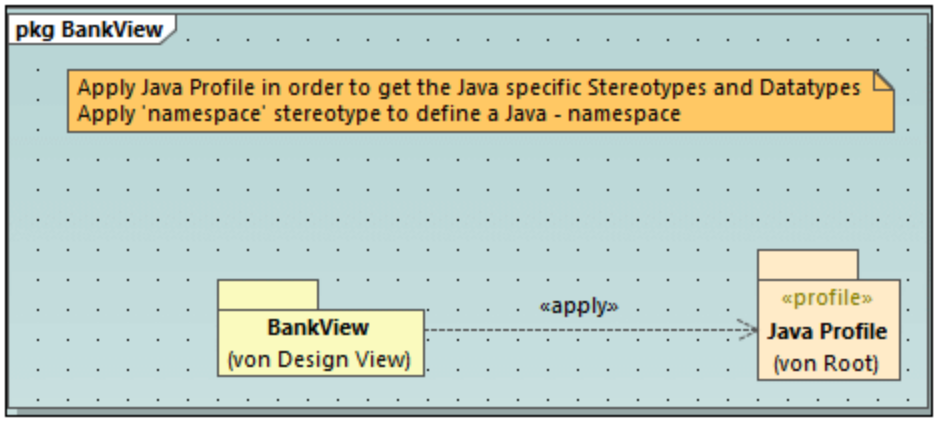
Altova Website: UML-Profildiagramme

Mit Hilfe von Profildiagrammen kann UML für eine bestimmte Plattform oder Domain erweitert werden. Im Gegensatz zu einem Paket befindet sich ein Profil im Metamodell und besteht aus Metabausteinen, die etwas erweitern oder einschränken. Dies lässt sich mit Hilfe der folgenden in ein Profil inkludierten Erweiterungsmechanismen bewerkstelligen: Stereotype, Eigenschaftswerte und Constraints.

Im Profildiagramm können Sie in UModel Ihre eigenen in einem benutzerdefinierten Profil gebündelten Stereotype, Eigenschaftswerte und Constraints erstellen. Mit Hilfe von Profilen können Sie UML erweitern oder an Ihre spezifische Domain adaptieren oder das Aussehen von Elementen in Ihren Modellierungsprojekten anpassen. So können Sie z.B. benutzerdefinierte Stile definieren oder benutzerdefinierte Symbole für UML-Elemente wie Klassen, Schnittstellen, usw. hinzufügen.

Über das Profildiagramm können Sie ein *Profil auf ein Paket anwenden*. Im unten gezeigten Profildiagramm sehen Sie z.B. eine **Profilzuweisungsbeziehung** zwischen dem Paket **BankView** und dem vordefinierten Java-Profil in UModel. Sie finden dieses Diagramm im folgenden Beispielprojekt: **C:**

\Benutzer\<Benutzername>\Dokumente\Altova\UModel2023\UModelExamples\BankView_Java.ump unter dem Namen "Apply Java Profile".



Profildiagramm

Gemäß dem angewendeten Java-Profil muss jede Klasse oder Schnittstelle, die Teil des **BankView**-Pakets ist (oder in Zukunft zu diesem Paket hinzugefügt wird) wie eine Java-Klasse oder -Schnittstelle aussehen und alle ihre Mitglieder müssen das sprachspezifische Verhalten aufweisen. Beispiel:

- Alle im Profil vorhandenen Java-Datentypen stehen bei der Erstellung einer Klasse in einem Klassendiagramm über eine Dropdown-Liste zur Auswahl zu Verfügung, siehe auch Klassendiagramme 25.
- Alle im Profil definierten Java-spezifischen Stereotype wie z.B. `«annotations»`, `«final»`, `«static»`, `«strictfp»`, usw., werden als Eigenschaften im Fenster "Eigenschaften" angezeigt, wenn Sie ein Element auswählen.

In diesem Kapitel wird beschrieben, wie Sie UModel-Projekte mit Hilfe von benutzerdefinierten Profilen und Stereotypen erweitern können. Informationen zur Verwendung der vordefinierten UModel-Profile finden Sie unter Anwenden von UModel-Profilen 160 und Stereotype und Eigenschaftswerte 145.

8.2.7.1 Erstellen und Anwenden von benutzerdefinierten Profilen

In der Anleitung unten wird beschrieben, wie Sie ein benutzerdefiniertes UModel-Profil erstellen und auf ein Paket anwenden. Dies ist normalerweise erforderlich, wenn Sie Stereotype erstellen und anwenden müssen, die nicht wie die UModel-Standardprofile bereits vordefiniert sind. Informationen zur Anwendung der UModel-Standardprofile finden Sie unter Anwenden von UModel-Profilen 160.

So erstellen Sie ein benutzerdefiniertes Profil:

1. Klicken Sie mit der rechten Maustaste auf das Paket, für das Sie das neue Profil erstellen möchten (z.B. "Root") und wählen Sie im Kontextmenü den Befehl **Neues Element | Profil**.
2. Erstellen Sie alle Elemente, die Teil dieses Profils sein sollen, wie z.B. Stereotype, Datentypen, usw. Sie können dies entweder im Fenster "Modell-Struktur" oder über ein Profildiagramm tun. Um z.B. ein neues Stereotyp im Modell zu erstellen, klicken Sie mit der rechten Maustaste auf das Profil und wählen Sie im Kontextmenü den Befehl **Neues Element | Stereotyp**. Siehe auch Erstellen von Stereotypen 424.

 © 2016-2022 Altova GmbH

3. Erstellen Sie optional ein Profildiagramm (Klicken Sie mit der rechten Maustaste auf das Profil und wählen Sie im Kontextmenü den Befehl **Neues Diagramm | Profildiagramm**). Um die gewünschten Elemente zum Diagramm hinzuzufügen, verwenden Sie die UModel-Standardmenübefehle und Symbolleisten, siehe Anleitung zur Modellierung von... 104.

> Wenn Sie das Profil anhand eines Profildiagramms erstellen möchten, stellen Sie sicher, dass das Diagramm im Eigentum eines Profils (unter einem Profil erstellt wurde) oder eines Pakets innerhalb eines Profils ist.

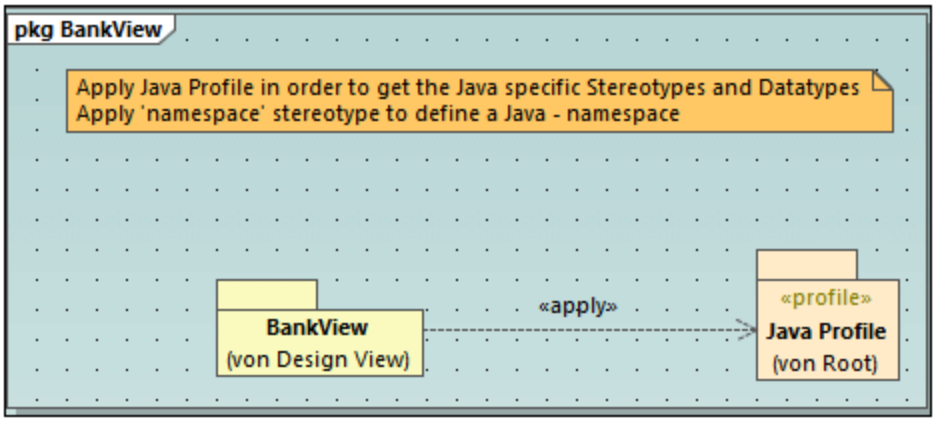
Wenn Sie das Profil in mehreren UModel-Projekten verwenden möchten, gehen Sie folgendermaßen vor:

1. Geben Sie alle Pakete, die Sie wiederverwenden möchten, frei. (Rechtsklick auf das Paket oder Profil selbst und Auswahl des Kontextmenübefehls **Unterprojekt | Paket freigeben**.)
2. Speichern Sie das Projekt in einem Verzeichnis, über das Sie es später als Unterprojekt inkludieren können, siehe Inkludieren von Unterprojekten 164.

Sie haben bisher ein Profil erstellt, es aber noch nicht zu einem Paket hinzugefügt (oder darauf angewendet). Durch Anwendung eines Profils auf ein Paket stellen Sie alle Erweiterungsmechanismen dieses Profils (wie z.B. Stereotype, Datentypen, usw.) für Elemente des Pakets zur Verfügung.

So wenden Sie ein benutzerdefiniertes Profil auf ein Paket an:

1. Erstellen Sie ein neues UModel-Projekt oder öffnen Sie ein bestehendes.
2. Wählen Sie eine der folgenden Methoden:
   a. Erstellen Sie Ihr benutzerdefiniertes Profil im vorhandenen Projekt, wie oben gezeigt.
   b. Inkludieren Sie mit dem Menübefehl **Projekt | Unterprojekt inkludieren** ein benutzerdefiniertes Profil aus einem vorhanden Projekt. Beachten Sie, dass entweder das gesamte Profil oder seine Pakete darunter freigegeben werden müssen, damit sie wiederverwendet werden können, siehe Freigeben von Paketen und Diagrammen 166 .
3. Klicken Sie mit der rechten Maustaste auf das Profil und wählen Sie im Kontextmenü den Befehl **Neues Diagramm | Profildiagramm**.
4. Fügen Sie ein oder mehrere Pakete und das benutzerdefinierte Profil zum Diagramm hinzu.
5. Ziehen Sie eine **Profilzuweisungsbeziehung** vom Paket zum Profil. So sehen Sie etwa im Profildiagramm unten eine **Profilzuweisungsbeziehung** zwischen dem Paket **BankView** und dem vordefinierten Java-Profil in UModel. Profilzuweisungen werden, wie unten gezeigt, zusammen mit dem Schlüsselwort `<<apply>>` als gestrichelte Pfeile vom Paket zum angewendeten Profil dargestellt.



8.2.7.2 Erstellen von Stereotypen

Wenn Sie Projekte mit Hilfe eines der vordefinierten UModel-Profile (wie z.B. C#, Java, VB.NET, XML Schema, usw.) modellieren, ist es normalerweise nicht notwendig, benutzerdefinierte Stereotype zu erstellen. Sie können stattdessen einfach die vorhandenen Stereotype auf die Elemente Ihres Modells anwenden, wie unter Anwenden von Stereotypen 147 beschrieben.

Wenn Sie jedoch benutzerdefinierte Symbole zu Elementen hinzufügen möchten oder deren Aussehen auf Basis des angewendeten Stereotyps anpassen möchten, können Sie dies mit Hilfe von benutzerdefinierten Stereotypen tun. Beachten Sie dabei die folgenden Voraussetzungen:

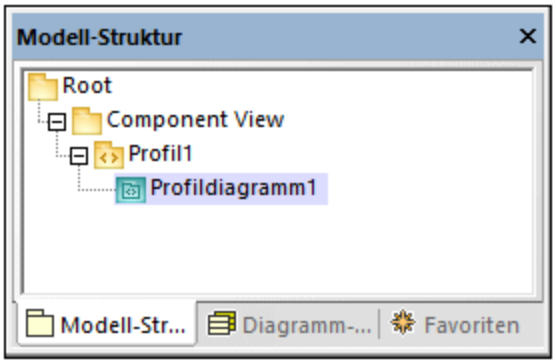
- Stereotype müssen als Owner ein Profil oder ein Paket innerhalb eines Profils haben. Um daher ein Stereotyp erstellen zu können, müssen Sie ein Profil (oder ein Paket innerhalb eines vorhandenen Profils) erstellen.
- Nachdem Sie das Profil erstellt haben, müssen Sie es auf das Paket, in dem Sie die benutzerdefinierten Stereotype verwenden möchten, anwenden, wie unter Erstellen und Anwenden von benutzerdefinierten Profilen 422 beschrieben.

Nachdem Sie ein Profil erstellt haben, können Sie Stereotype dazu hinzufügen. Sie können dies entweder direkt im Fenster "Modell-Struktur" oder über ein Profildiagramm tun. Wenn Sie Stereotype über ein Profildiagramm erstellen möchten, stellen Sie sicher, dass das Diagramm als Owner ein Profil oder ein Paket innerhalb eines Profils hat, wie unten gezeigt.

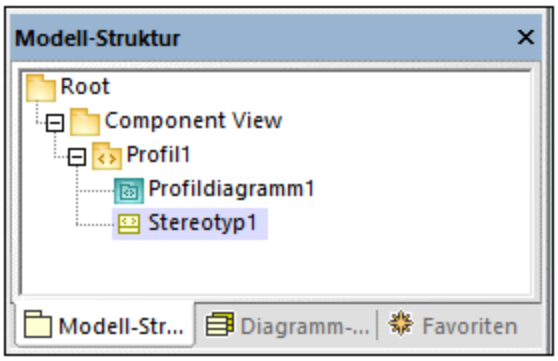
So erstellen Sie ein Stereotyp:

1. Erstellen Sie zuerst ein Profil, falls Sie dies noch nicht getan haben, siehe Erstellen und Anwenden von benutzerdefinierten Profilen 422.
2. Klicken Sie optional mit der rechten Maustaste auf das Profil und wählen Sie im Kontextmenü den Befehl **Neues Diagramm | Profildiagramm**. Dadurch wird unter dem aktuellen Profil ein neues Profil erstellt - so sehen sie alle Stereotype, Datentypen und anderen Elemente, die sie später zu diesem Profil hinzufügen, an einer Stelle.

 © 2016-2022 Altova GmbH



3. Klicken Sie mit der rechten Maustaste im Fenster "Modell-Struktur" auf das Profil und wählen Sie im Kontextmenü den Befehl **Neues Element | Stereotyp**.



4. Definieren Sie optional im Fenster "Eigenschaften" die Eigenschaften des Stereotyps. Wenn Sie z.B. die **Metaklasse** des Stereotyps auf "Klasse" setzen, gilt das Stereotyp nur für Klassen. Ebenso können Sie ein benutzerdefiniertes Symbol für das Stereotyp definieren, indem Sie neben **Symboldateiname** auf die **Auslassungspunkte** ... klicken.

Anmerkungen

- Wenn der Pfad des Bilds relativ ist, muss er relativ zum UModel-Projektordner sein.

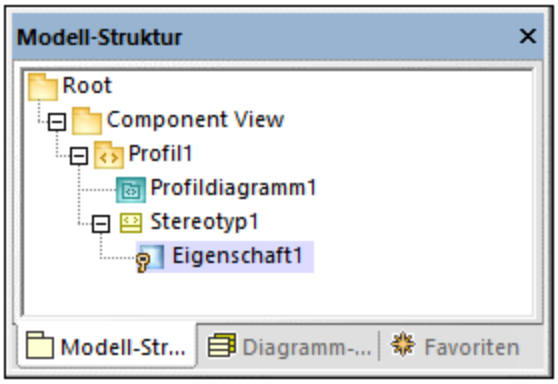
- Um benutzerdefinierte Symbole mit transparentem Hintergrund zu verwenden, setzen Sie die Hintergrundfarbe auf den RGB-Wert 82,82,82.
- Um Stereotype für Assoziationsbeziehungen anzuzeigen, setzen Sie im Fenster "Stile" die Eigenschaft **MemberEnd-Stereotype anzeigen** auf "true".

Hinzufügen von Stereotyp-Attributen (Eigenschaften)

Das oben erstellte Stereotyp ist sehr einfach gehalten und es sind damit keine Attribute (Eigenschaften) verknüpft. Es können jedoch auch Eigenschaften zu einem Stereotyp hinzugefügt werden. Solche Eigenschaften werden zu Eigenschaftswerten, wenn dieses Stereotyp in Zukunft auf ein Element angewendet wird.

So fügen Sie Attribute (Eigenschaften) zu einem Stereotyp hinzu:

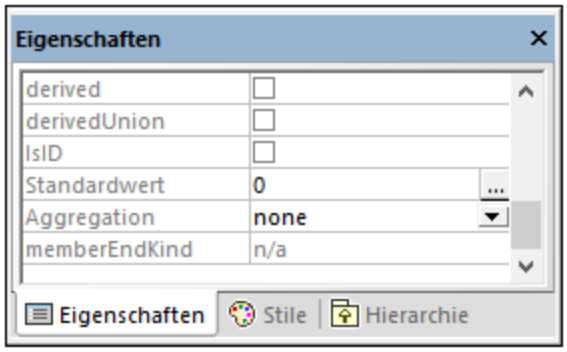
1. Klicken Sie im Fenster "Modell-Struktur" oder im Diagramm auf das Stereotyp.
2. Wählen Sie eine der folgenden Methoden
   a. Klicken Sie mit der rechten Maustaste auf das Stereotyp und wählen Sie im Kontextmenü den Befehl **Neu | Eigenschaft**.
   b. Drücken Sie **F7**.



Sie können über das Fenster "Eigenschaften" den Datentyp jeder Eigenschaft definieren, indem Sie einen Wert aus der Liste **Typ** auswählen. Zur Auswahl steht jeder Datentyp, der zuvor im selben Profil wie das Stereotyp definiert wurde. Wenn das Profil noch keine Datentypen enthält, können Sie einen definieren, indem Sie mit der rechten Maustaste auf das Profildiagramm klicken und im Kontextmenü den Befehl **Neu | Datentyp** auswählen.

Um den Standardwert einer Eigenschaft zu definieren, geben Sie diesen Wert im Fenster "Eigenschaften" in das Feld **Standardwert** ein. Die unten gezeigte Stereotypeigenschaft hat z.B. den Standardwert "0".

 © 2016-2022 Altova GmbH



Der Datentyp eines Stereotypattributs (Eigenschaft) kann auch eine Enumeration sein, siehe Beispiel: Erstellen und Anwenden von Stereotypen 427.

8.2.7.3 Beispiel: Erstellen und Anwenden von Stereotypen

In diesem Beispiel wird Schritt für Schritt beschrieben, wie ein Stereotyp erstellt wird. Es wird gezeigt, wie Sie die folgenden Zielsetzungen erreichen:

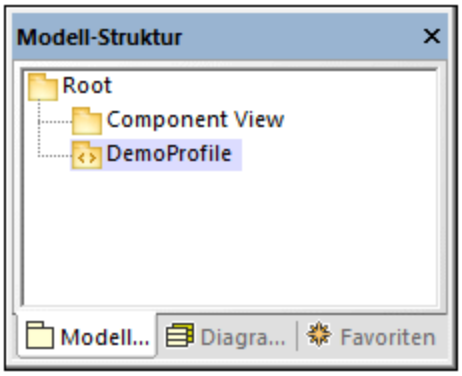
- Erstellung eines Stereotyps
- Erstellung von Stereotypattributen (Eigenschaften), die bei Anwendung auf ein Element zu Eigenschaftswerten werden
- Definition eines Stereotypattributs als Enumeration
- Definition eines Standardwerts für ein Stereotypattribut
- Anwendung des Stereotyps auf Elemente im Modell.

Die Beispielprojektdatei dazu finden Sie unter dem Namen **StereotypesDemo.ump** unter dem folgenden Pfad: **C:\Benutzer\<Benutzername>\Dokumente\Altova\UModel2023\UModelExamples\Tutorial**. Mit der nachstehenden Anleitung können Sie ein ähnliches Projekt erstellen.

Erstellung eines neuen Profils

Wie oben erwähnt, muss ein Stereotyp als Owner ein Profil haben. Wir wollen daher zuerst ein Profil erstellen.

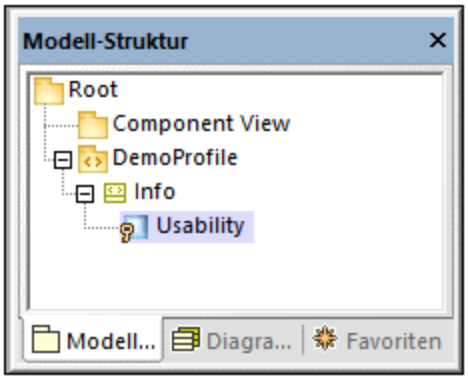
1. Erstellen Sie ein neues UModel-Projekt.
2. Klicken Sie mit der rechten Maustaste auf das Root-Paket und fügen Sie durch Auswahl des Kontextmenüs **Neues Element | Profil** ein neues Profil hinzu.
3. Benennen Sie das neue Profil in "DemoProfile" um.



Erstellen eines Stereotyps

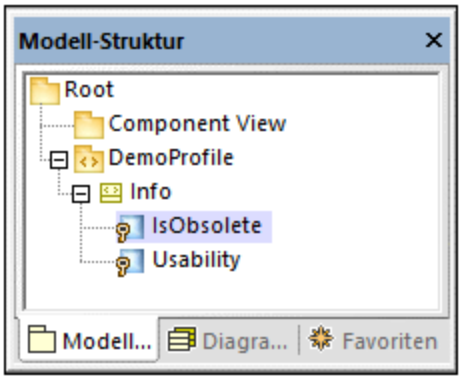
Wir wollen in diesem Tutorial ein Stereotyp mit zwei Attributen erstellen: "Usability" und "IsObsolete". Das Attribut "IsObsolete" wird als Enumeration definiert. Die Enumeration besteht aus den zwei Werten "Yes" und "No", wobei "No" der Standardwert ist.

1. Klicken Sie mit der rechten Maustaste auf das Profil und wählen Sie im Kontextmenü den Befehl **Neues Element | Stereotyp** aus. Daraufhin wird ein neues Stereotyp zum Profil hinzugefügt.
2. Benennen Sie das neue Stereotyp in "Info" um.
3. Klicken Sie mit der rechten Maustaste auf das Stereotyp und wählen Sie im Kontextmenü den Befehl **Neues Element | Eigenschaft**. Daraufhin wird eine neue Eigenschaft hinzugefügt.
4. Benennen Sie die neue Eigenschaft in "Usability" um.

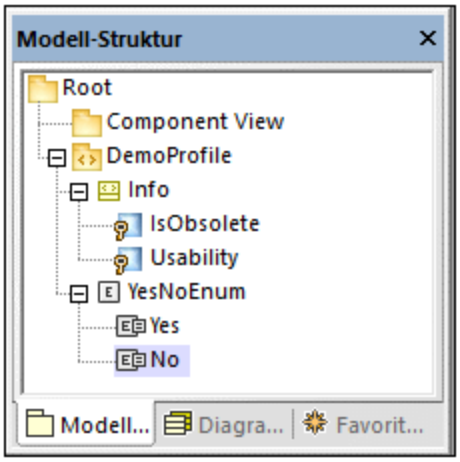


5. Wiederholen Sie die obigen Schritte, um eine neue Eigenschaft namens "IsObsolete" zu erstellen.

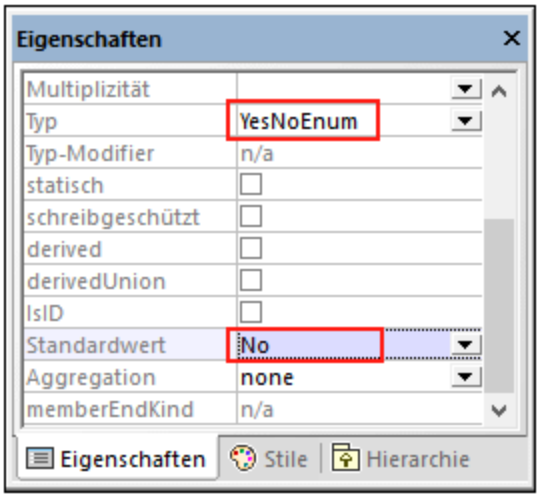
© 2016-2022 Altova GmbH



6. Klicken Sie mit der rechten Maustaste auf das "DemoProfile" und wählen Sie im Kontextmenü **Neues Element | Enumeration.** Benennen Sie die Eumeration in "YesNoEnum" um.
7. Klicken Sie mit der rechten Maustaste auf die Enumeration und wählen Sie im Kontextmenü **Neues Element | EnumerationLiteral.** Benennen Sie die Eumeration in "Yes" um.
8. Wiederholen Sie den obigen Schritt und erstellen Sie ein Enumerationsliteral namens "No".



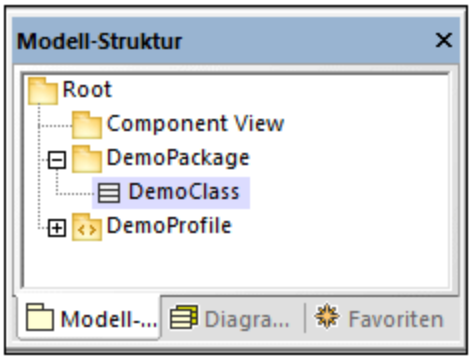
9. Klicken Sie auf die Eigenschaft "IsObsolete" und ändern Sie ihren Typ in `YesNoEnum`. Setzen Sie außerdem die Eigenschaft **Standardwert** auf "No"



Erstellen eines neues Pakets

Um zu zeigen, wie das benutzerdefinierte Stereotyp verwendet werden kann, wollen wir ein einfaches Paket, das nur eine Klasse enthält, erstellen.

1. Klicken Sie mit der rechten Maustaste auf das Root-Paket und fügen Sie durch Auswahl des Kontextmenübefehls **Neues Element | Paket** ein neues Paket hinzu.
2. Benennen Sie das neue Paket in "DemoPackage" um.
3. Fügen Sie eine Klasse zum Paket hinzu (in diesem Beispiel "DemoClass").

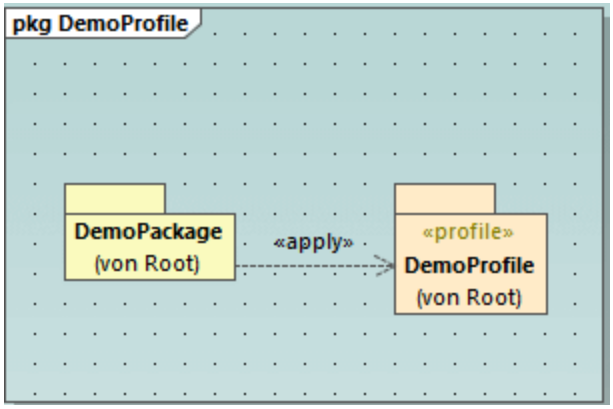


Anwenden des Profils auf ein Paket

Wie in Schritt 1 erwähnt, wurde das Stereotyp innerhalb eines Profils erstellt. In diesem Schritt wenden wir das Profil auf ein Paket an, damit das Stereotyp für das Paket "sichtbar" wird.

1. Klicken Sie mit der rechten Maustaste im Fenster "Modell-Struktur" auf das "DemoProfile" und wählen Sie im Kontextmenü den Befehl **Neues Diagramm | Profildiagramm**.
2. Ziehen Sie sowohl das Paket "DemoPackage" als auch das Profil "DemoProfile" aus dem Fenster "Modell-Struktur" in das Diagramm.
3. Klicken Sie auf die Symbolleistenschaltfläche **Profilzuweisung** und ziehen Sie eine **Profilzuweisungsbeziehung** vom Paket auf das Profil.

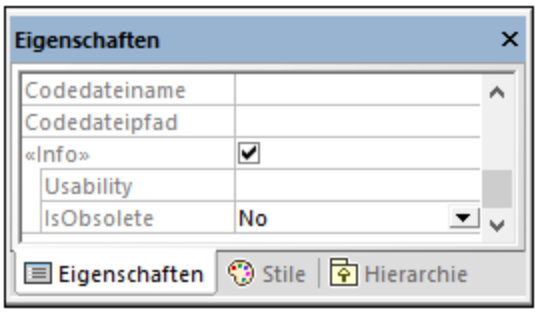
 © 2016-2022 Altova GmbH



Anwenden des Stereotyps auf Klassen

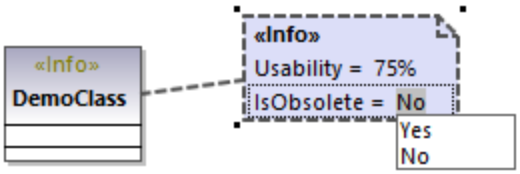
Sie können das Stereotyp nun auf eine Klasse anwenden.

1. Klicken Sie mit der rechten Maustaste auf das "DemoPackage" und wählen Sie im Kontextmenü den Befehl **Neues Diagramm | Klassendiagramm**.
2. Ziehen Sie die Klasse "DemoClass" in das Diagramm.
3. Klicken Sie auf die Klasse und aktivieren Sie im Fenster "Eigenschaften" das Stereotyp `«Info»`. Beachten Sie, dass für die Eigenschaft "IsObsolete" der Standardwert ausgefüllt wurde.



4. Geben Sie einen Wert für die Eigenschaft "Usability" ein (in diesem Beispiel "75%").

Die Klasse im Diagramm hat nun einen Eigenschaftswerte-Abschnitt, in dem die Stereotypattribute und deren Werte angezeigt werden. Sie können diese Werte entweder über das Fenster "Eigenschaften" oder direkt über das Diagramm ändern.



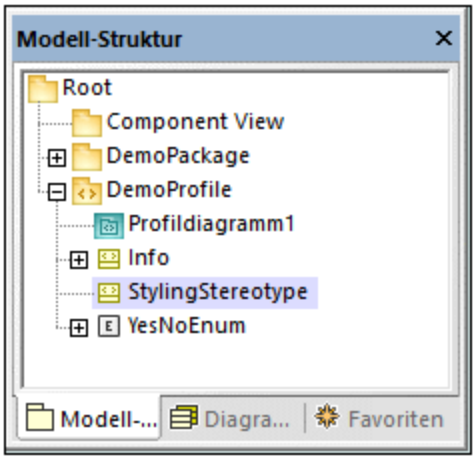
8.2.7.4 Beispiel: Anpassen von Symbolen und Stilen

In diesem Beispiel wird gezeigt, wie Sie das Aussehen einer Klasse in UModel mit Hilfe von Stereotypen anpassen. Nachdem Sie dieses Beispiel fertig gestellt haben, erfahren Sie, wie Sie benutzerdefinierte Elemente zu Symbolen hinzufügen und den Stil aller Elemente, für die dasselbe Stereotyp verwendet wird, ändern.

Die Klasse, die in diesem Beispiel angepasst wird, befindet sich im Projekt **StereotypesDemo.ump,** das unter dem folgenden Pfad zur Verfügung steht: **C:\Benutzer\<Benutzername>\Dokumente\Altova\UModel2023\UModelExamples\Tutorial**. Dies ist ein einfaches Demo-Projekt, das ein benutzerdefiniertes Profil enthält, unter dem wir das Stereotyp erstellen werden. Ein Beispiel dafür, wie Sie Profile und Stereotype von Grund auf neu erstellen, finden Sie unter Beispiel: Erstellen und Anwenden von Stereotypen 427.

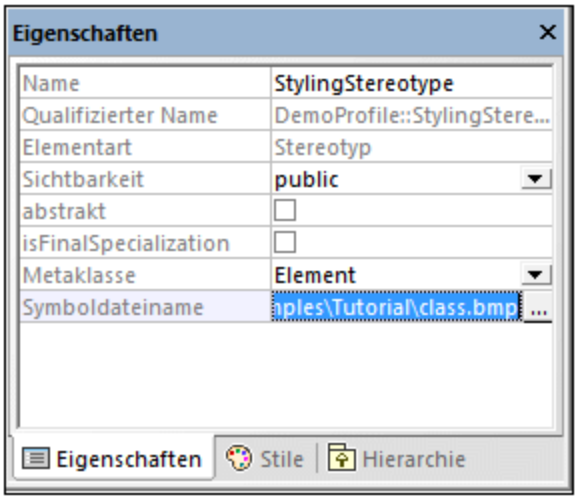
Erstellen wir zuerst das Stereotyp, das wir für die Stile verwenden:

1. Öffnen Sie das Projekt **StereotypesDemo.ump**.
2. Klicken Sie in der Modell-Struktur mit der rechten Maustaste auf das Profil "DemoProfile" und wählen Sie im Kontextmenü den Befehl **Neues Element | Stereotyp**.
3. Benennen Sie das Stereotyp in "StylingStereotype" um.



Um ein benutzerdefiniertes Bild zum Stereotyp hinzuzufügen, klicken Sie auf das Stereotyp und anschließend im Fenster Eigenschaften" auf die **Auslassungspunkte** ... neben der Eigenschaft **Symboldateiname**. Wählen Sie das folgende Beispielbild aus: **C:\Benutzer\<Benutzername>\Dokumente\Altova\UModel2023\UModelExamples\Tutorial\class.bmp**.

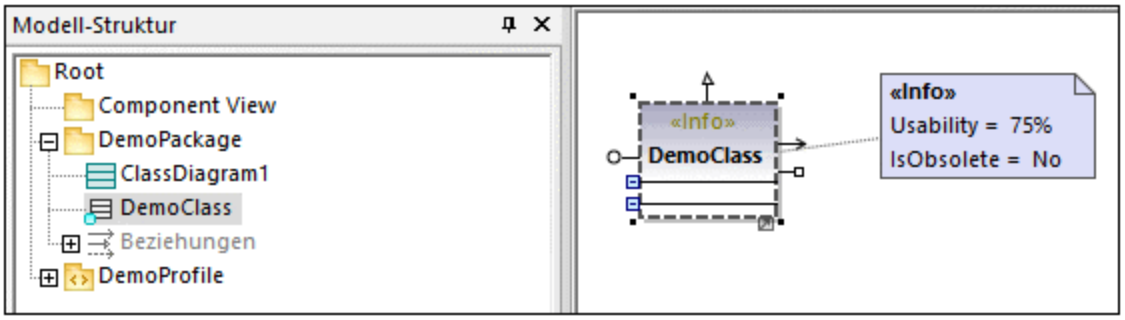
 © 2016-2022 Altova GmbH



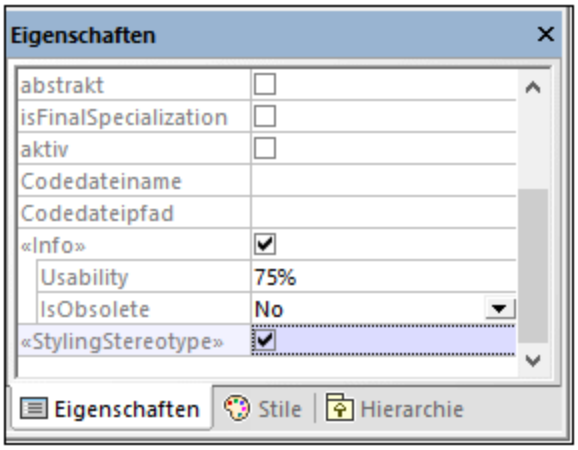
Klicken Sie als nächstes auf das Register **Stile** des Fensters **Eigenschaften**. Wählen Sie in der Liste ganz oben den Eintrag **Stile von Elementen mit diesem Stereotyp** und ändern Sie die Eigenschaft **Schriftgröße Titel** in "16".

Wenden Sie das Stereotyp schließlich auf eine Klasse an.

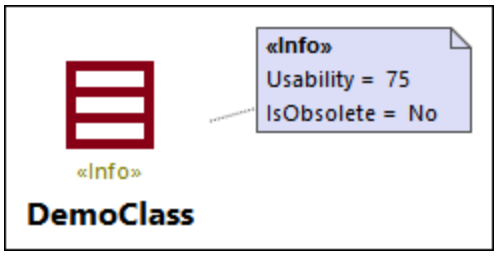
1. Öffnen Sie das Klassendiagramm "ClassDiagram1". Sie finden dieses Diagramm im Fenster "Modell-Struktur" unter dem "DemoPackage".



2. Klicken Sie auf die Klasse "DemoClass" und aktivieren Sie im Fenster "Eigenschaften" das Kontrollkästchen **«StylingStereotype»**.

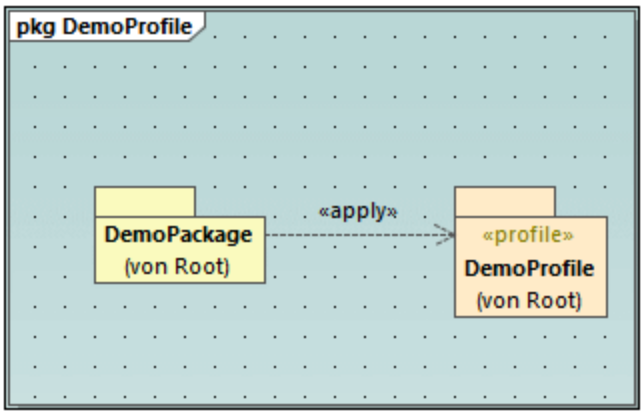


Das Aussehen der Klasse im Diagramm ändert sich nun gemäß dem angewendeten Stereotyp:



Anmerkungen

Das Demo-Projekt enthält das Profildiagramm "ProfileDiagram1". Beachten, Sie dass das "DemoProfile" mit einer Profilzuweisungsbeziehung auf das "DemoPackage" angewendet wurde. Dadurch steht das Stereotyp für das Paket zur Verfügung, siehe auch Erstellen und Anwenden von benutzerdefinierten Profilen 422.



Sie wissen nun, wie man das Aussehen von Elementen mit Hilfe von Stereotypen ändert. Auf dieselbe Art können Sie auch in anderen Projekten arbeiten. Denken Sie allerdings daran, dass das Profil, in dem Sie das Stereotyp erstellen, auf das Zielpaket angewendet werden muss, wie oben gezeigt.

 © 2016-2022 Altova GmbH

8.3 Zusätzliche Diagramme

Die **UModel Basic Edition** unterstützt die folgenden zusätzlichen Diagramme:

XML-Schema-Diagramme 435

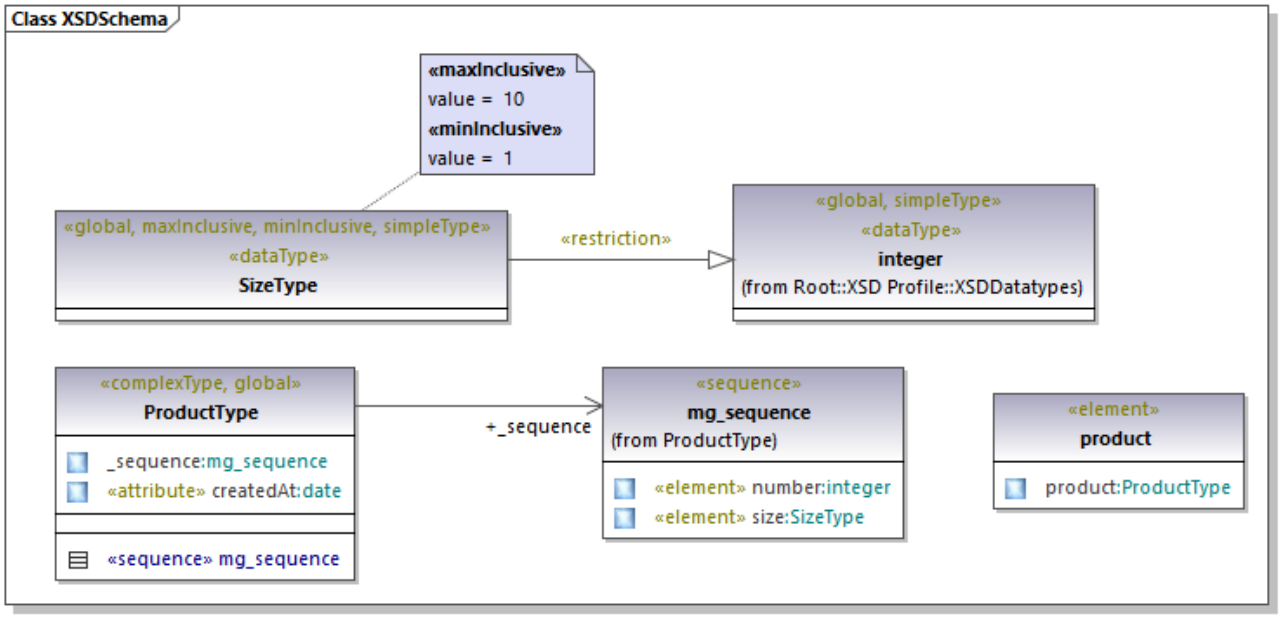
8.3.1 XML-Schema-Diagramme

Altova Website: XML-Schemas in UML

UModel unterstützt den Import und die Generierung von W3C XML Schemas sowie deren Forward und Reverse Engineering. Im Fall von XML-Schemas bedeutet Forward und Reverse Engineering, dass Sie ein Schema (oder mehrere Schemas aus einem Verzeichnis) in UModel importieren, das Modell anzeigen oder ändern und die Änderungen wieder in die Schema-Datei schreiben können. Wenn Sie Daten aus dem Modell in einer Schema-Datei synchronisieren, wird die Schema-Datei immer durch das Modell überschrieben.

Anmerkung: Das XML-Schema muss gültig sein, bevor es in UModel importiert werden kann. XML-Schemas werden nicht validiert, wenn Sie diese in UModel erstellen oder importieren oder wenn Sie eine Syntaxüberprüfung am Projekt durchführen. UModel überprüft jedoch beim Import, ob das XML-Schema wohlgeformt ist.

In XML-Schema-Diagrammen werden Schemakomponenten in UML-Notation dargestellt. So werden etwa simpleTypes in UModel als Datentypen mit dem Stereotyp `«simpleType»` angezeigt. ComplexTypes werden als Klassen mit dem Stereotyp `«complexType»` angezeigt. Verschiedene Schemainformationen werden als Eigenschaftswerte 146 ,dargestellt, während Schema-Annotationen als Kommentare ausgewiesen werden. Unter XML-Schema-Entsprechungen 268 finden Sie eine Tabelle, in der aufgelistet ist, wie alle XML-Schema-Komponenten UModel-Elementen zugeordnet werden.



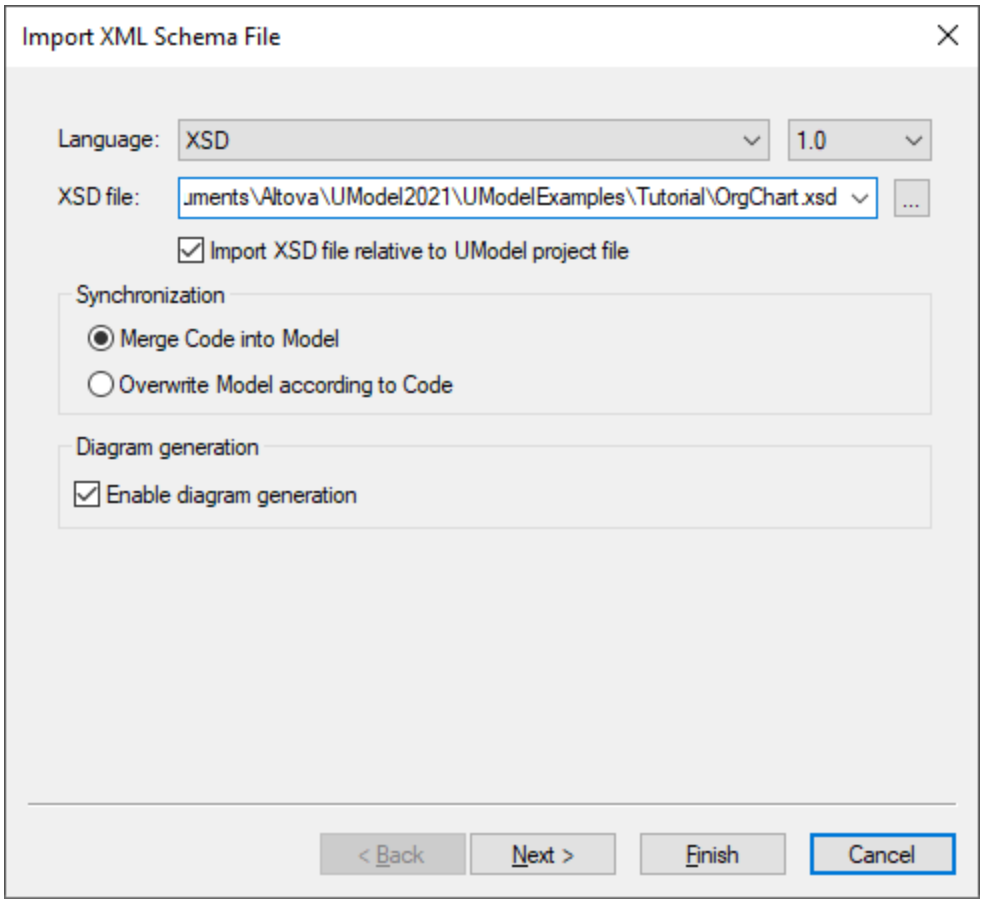
Beispiel für ein XML-Schema-Diagramm

8.3.1.1 Importieren von XML-Schemas

Sie können entweder eine einzelne Schema-Datei oder alle Schemas aus einem Verzeichnis in UModel importieren. Wenn in ein Schema andere Schemas inkludiert oder importiert wurden, so werden diese auch in das Modell importiert.

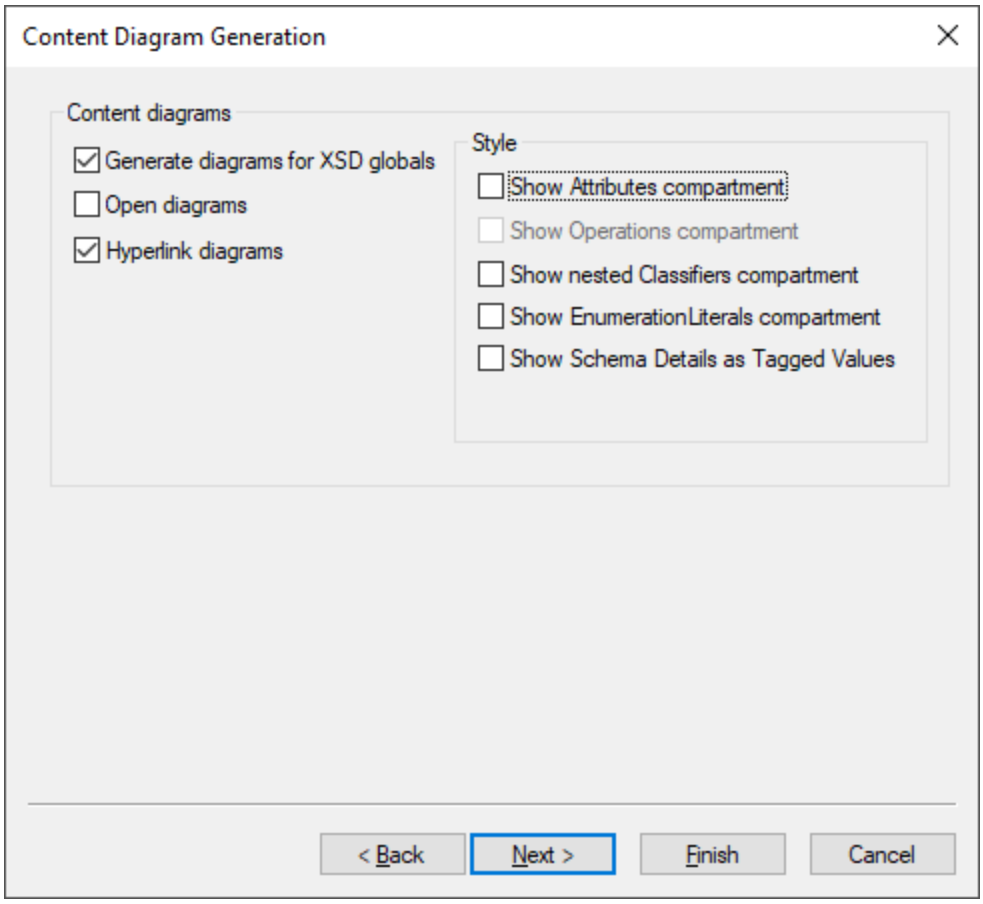
So importieren Sie ein einzelnes XML-Schema:

1. Wählen Sie den Menübefehl **Projekt | XML-Schema-Datei importieren**.
2. Klicken Sie auf **Durchsuchen** und wählen Sie das Quellschema aus. In diesem Beispiel können Sie das folgende Schema verwenden: **C:\Benutzer\<Benutzername>\Dokumente\Altova\UModel2023\UModelExamples\Tutorial\OrgChart.xsd**.

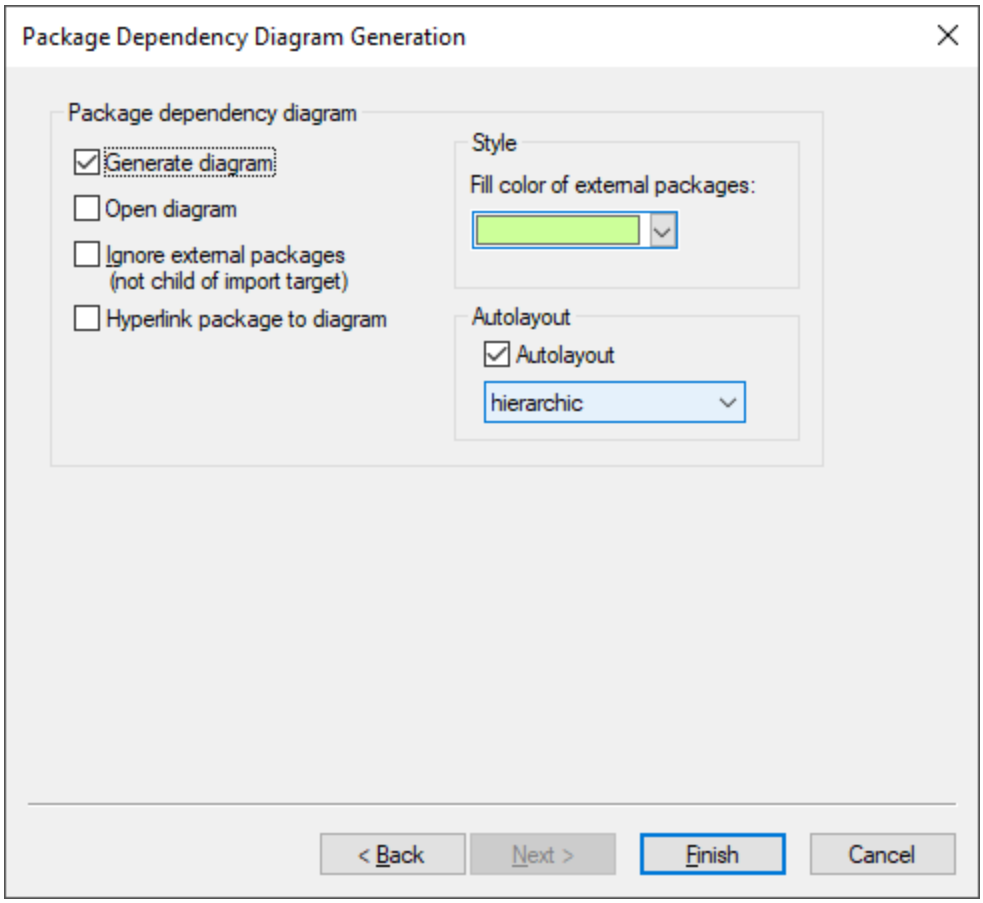


3. Damit anhand des Schemas Diagramme generiert werden, stellen Sie sicher, dass das Kontrollkästchen **Diagrammgenerierung aktivieren** aktiviert ist und klicken Sie auf **Weiter**.

© 2016-2022 Altova GmbH



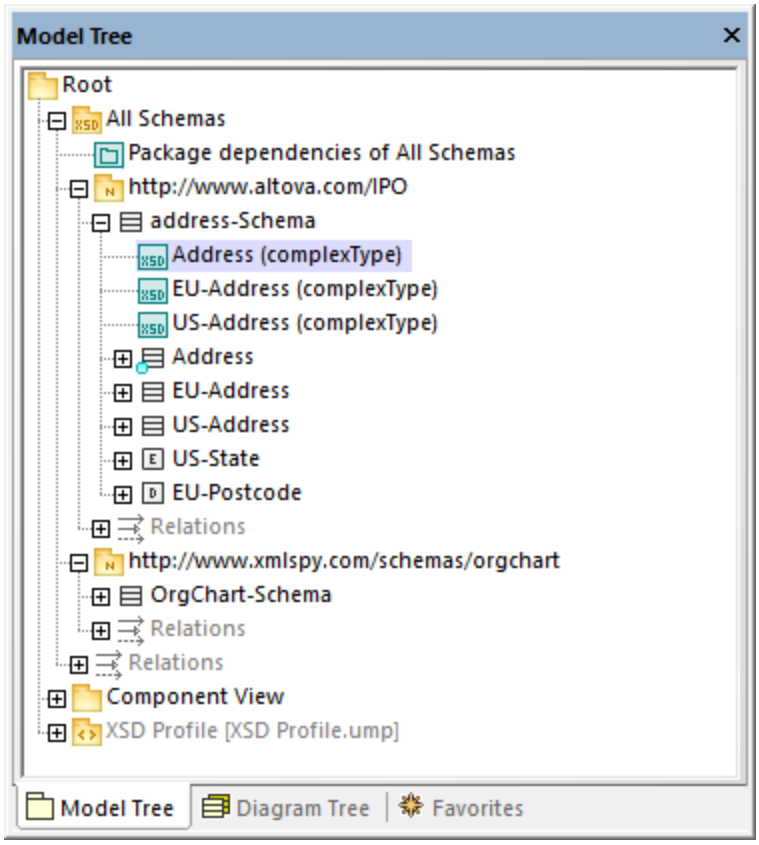
4. Um, wie in diesem Beispiel gezeigt, für jede globale Komponente im Schema ein separates Diagramm zu erstellen, aktivieren Sie die Option **Diagramme für globale XSD-Elemente generieren**. Um alle generierten Diagramme nach dem Import zu öffnen, wählen Sie die Option **Diagramme öffnen**. Über die Optionen aus der Gruppe "Stil" können Sie definieren, welche Bereiche standardmäßig für die einzelnen Schemakomponenten in Diagrammen angezeigt werden sollen. Bei Auswahl der Option **Schemainformationen als Eigenschaftswerte anzeigen** werden die Schemainformationen als Eigenschaftswerte [146] angezeigt.
5. Klicken Sie auf **Weiter**. Um ein Paketabhängigkeitsdiagramm wie dasjenige in diesem Beispiel zu generieren, aktivieren Sie das Kontrollkästchen **Diagramm generieren**.



6. Klicken Sie auf **Fertig stellen**.

Nachdem das Schema fertig in UModel importiert wurde, wird ein neues Paket namens **Alle Schemas** erstellt und automatisch als die XSD Namespace Root definiert. Im Schema **OrgChart.xsd** aus diesem Beispiel werden Typen aus einem anderen Namespace, nämlich aus dem Schema **ipo.xsd** importiert. Folglich werden beide Schemas nach dem Import im Fenster "Modell-Struktur" unter ihrem jeweiligen Namespace angezeigt:

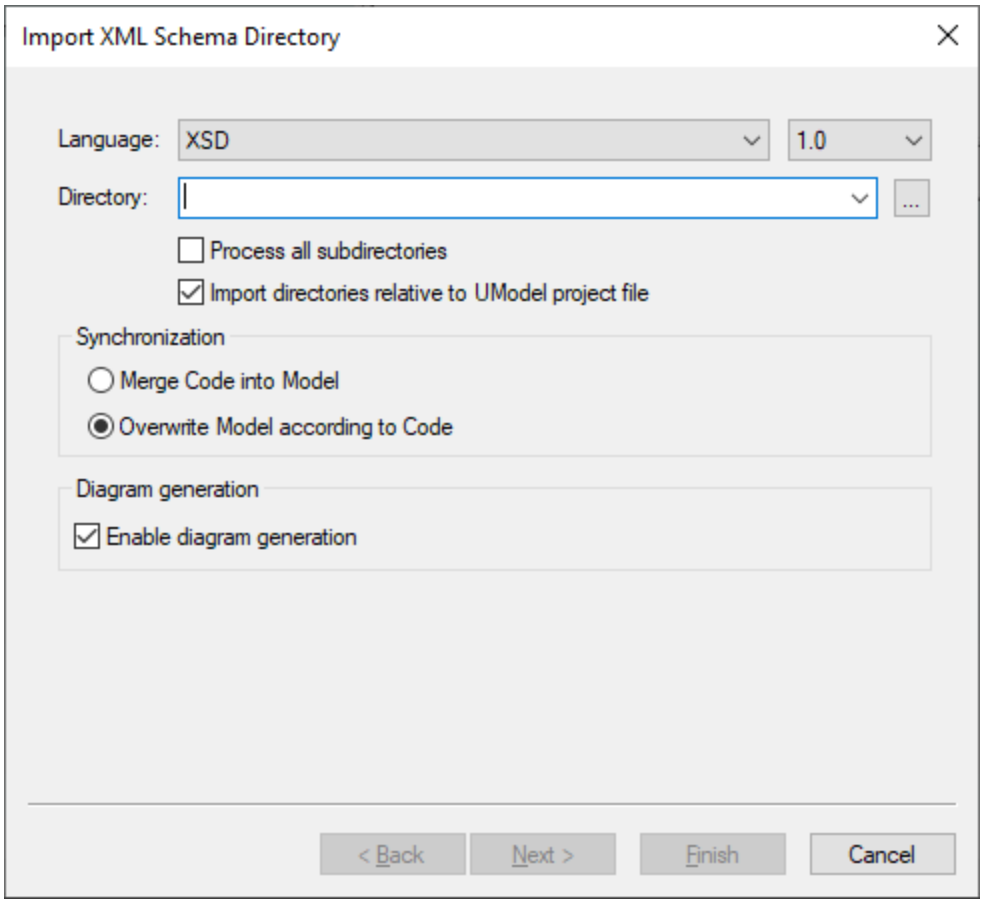
 © 2016-2022 Altova GmbH



Wenn Sie das Kontrollkästchen **Diagramme für globale XSD-Elemente generieren** aktiviert haben, wird anhand aller globalen XSD-Komponenten ein XML-Schema-Diagramm generiert, wobei die Diagramme, wie etwa das Diagramm "Address (complexType)" in der Abbildung oben, unter den jeweiligen Namespace-Paketen angelegt werden.

So importieren Sie mehrere XML-Schemas:

1. Wählen Sie den Menübefehl **Projekt | XML-Schemaverzeichnis importieren**.

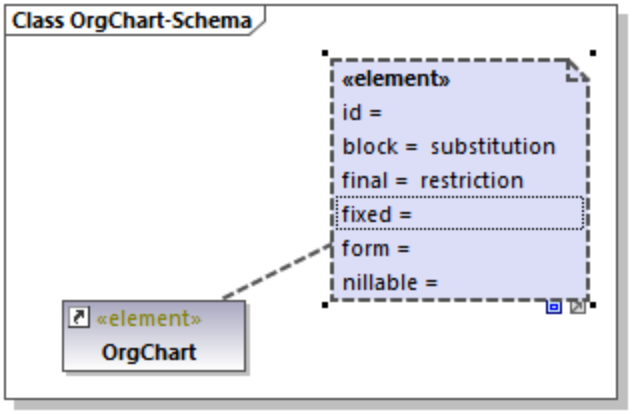


2. Um Schemas aus allen Unterverzeichnissen des ausgewählten Verzeichnisses zu importieren, aktivieren Sie das Kontrollkästchen **Alle Unterverzeichnisse verarbeiten**. Der restliche Importvorgang läuft ab, wie oben für ein einzelnes XML-Schema beschrieben.

Änderung der Anzeige von Eigenschaftswerten

Nach dem Import eines XML-Schemas werden bestimmte Schemainformationen im Diagramm eventuell als Eigenschaftswerte angezeigt, wenn Sie beim Import die Option **Schemainformationen als Eigenschaftswerte anzeigen** aktiviert haben.

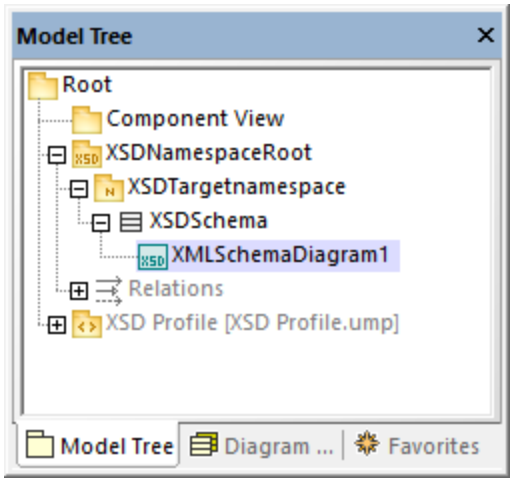
 © 2016-2022 Altova GmbH



Sie können konfigurieren, ob diese Informationen im Diagramm ein- oder ausgeblendet werden sollen. Klicken Sie dazu mit der rechten Maustaste auf das Element und wählen Sie im Kontextmenü den Befehl **Eigenschaftswerte | <Option>**. Sie können die Anzeige von Eigenschaftswerten nicht nur für einzelne Elemente, sondern auch global auf Projektebene konfigurieren. Nähere Informationen dazu finden Sie unter Anzeigen oder Ausblenden von Eigenschaftswerten 149.

8.3.1.2 Modellieren von XML-Schemas

Neue XML-Schema-Projekte haben in UModel die unten gezeigte Struktur. Diese Struktur wird beim ersten Mal, wenn Sie ein XML-Schema-Diagramm zu einem neuen UModel-Projekt hinzufügen, automatisch erstellt.



Die Pakete "Root" und "Komponentenansicht" kommen in allen UModel-Projekten vor und können nicht gelöscht werden. "Root" ist die oberste Ebene, unterhalb welcher weitere Pakete hinzugefügt werden und "Component View" wird für das Code Engineering (in diesem Fall für den Import und die Generierung von Schema-Dateien) verwendet.

Das Paket "XSDNamespaceRoot" enthält alle in Ihrem bzw. Ihren Schemas verwendeten Namespaces. Um ein Paket in eine XSD Namespace Root umzuwandeln, klicken Sie mit der rechten Maustaste darauf und wählen Sie im Kontextmenü den Befehl **Code Engineering | Als XSD Namespace Root definieren**. Wenn Sie ein

vorhandenes XML-Schema in das Projekt importieren, erhält das Paket standardmäßig den Namen "Alle Schemas".

Das Paket "XSDTargetNamespace" ist ein XML-Schema-Namespace. Unter ein und derselben XSD Namespace Root können sich mehrere solcher Namespaces befinden. Um ein Paket in einen Namespace umzuwandeln, wählen Sie es zuerst aus und aktivieren Sie anschließend im Fenster "Eigenschaften" die Eigenschaft (Stereotyp) «namespace».

"XSDSchema" ist ein Schema oder in UML eine Klasse, für die im Fenster "Eigenschaften" die Eigenschaft (Stereotyp) «schema» ausgewählt ist.

XMLSchemaDiagram1 ist das eigentliche Diagramm, in dem das Modell des Schemas beschrieben ist. Sie können XML-Schema-Diagramme unter einer XSD Namespace Root, einem XML Schema Namespace oder einem XML-Schema erstellen. Im oben gezeigten Beispielprojekt wurde das Diagramm unter dem XML-Schema erstellt.

Mit Hilfe des **XSD-Profils** werden alle Typen und Strukturen, die für die Arbeit mit XML-Schemas benötigt werden, im Projekt aktiviert. Wenn Ihr Projekt dieses Profil nicht hat, werden Sie bei der Erstellung eines neuen XML-Schemas aufgefordert, es zu inkludieren. Sie können das XSD-Profil auch explizit zu einem Projekt hinzufügen, siehe Anwenden von UModel-Profilen 160.

Erstellen von XML-Schema-Diagrammen

So erstellen Sie ein neues XML-Schema-Diagramm:

1. Wählen Sie eine der folgenden Methoden:
   a. Klicken Sie mit der rechten Maustaste im Fenster "Modell-Struktur" 80 auf ein Paket und wählen Sie im Kontextmenü den Befehl **XML-Schema-Diagramm**.
   b. Klicken Sie mit der rechten Maustaste im Fenster "Diagramm-Struktur" 84 auf "Diagramme" oder "XML-Schema-Diagramme" und wählen Sie im Kontextmenü den Befehl **Neues Diagramm | XML-Schema-Diagramm**. Darauf wird ein Dialogfeld angezeigt, in dem Sie aufgefordert werden, den Owner (Eigentümer) des Diagramms auszuwählen. Wählen Sie ein Paket für das Diagramm aus und klicken Sie auf **OK**.
2. Wenn das aktuelle UModel-Projekt das XSD-Profil nicht enthält, wird ein Dialogfeld geöffnet, in dem Sie aufgefordert werden, es zu inkludieren. Klicken Sie auf **OK**, um das XSD-Profil in das aktuelle Projekt zu inkludieren, siehe auch Anwenden von UModel-Profilen 160.

Hinzufügen von neuen XML-Schema-Elementen

So fügen Sie XML-Schema-Elemente zu einem Diagramm hinzu:

- Klicken Sie auf eine bestimmte Symbolleisten-Schaltfläche und anschließend in das XML-Schema-Diagramm.

Um mehrere Elemente desselben Typs einzufügen, halten Sie die **Strg**-Taste gedrückt und klicken Sie mehrmals in das Diagramm.

 © 2016-2022 Altova GmbH

XML-Schema-Diagramme können, wie oben erwähnt, auf verschiedenen Ebenen der Projektstruktur erstellt werden. Wenn sich das Diagramm auf einer Ebene befindet, die die Platzierung bestimmter Elemente nicht zulässt, werden bestimmte Symbolleisten-Schaltflächen nicht benötigt. Für diese wird ein Tooltipp mit Informationen angezeigt, anstatt dass das Element hinzugefügt wird.

In der nachstehenden Tabelle finden Sie eine Liste aller Symbolleisten-Schaltflächen und ihrer Bedeutung.

	XSD Targetnamespace	Fügt einen XSD Target Namespace hinzu. Klicken auf diese Schaltfläche ist sinnvoll, wenn das Diagramm direkt unter einer XSD Namespace Root erstellt wurde.
	XSD Schema	Fügt eine XML-Schema-Definition (XSD) hinzu. Klicken auf diese Schaltfläche ist sinnvoll, wenn das Diagramm unter einem XSD Target Namespace erstellt wurde.
	Element (global)	Fügt ein globales Element zum Diagramm hinzu. Wenn Sie ein Element hinzufügen, wird im Attributbereich automatisch eine Eigenschaft mit demselben Namen wie das Element generiert. Definieren Sie, dass der Typ der Eigenschaft den Typ des Elements definiert.
	Group	Fügt eine benannte Modellgruppe zum Diagramm hinzu.
	Complex Type	Fügt einen globalen ComplexType zum Diagramm hinzu. In der UML-Terminologie ist dies eine Klasse, auf die die Stereotype «`global`» und «`complexType`» angewendet wurden.
	Complex Type mit Simple Content	Fügt einen globalen ComplexType mit einfachem Inhalt zum Diagramm hinzu. In der UML-Terminologie ist dies ein Datentyp, auf den die Stereotype «`global`», «`complexType`» und «`simpleContent`» angewendet wurden.
	Simple Type	Fügt einen globalen SimpleType hinzu.
	List	Fügt einen list-Typ hinzu.
	Union	Fügt einen union-Typ hinzu.
	Enumeration	Fügt eine Enumeration hinzu.
	Attribute	Fügt ein Attribut hinzu.
	Attribute Group	Fügt eine Attributgruppe hinzu.
	Notation	Fügt einen notation-Typ hinzu.
	Import	Fügt eine Import-Beziehung hinzu.
	Include	Fügt eine Include-Beziehung hinzu.
	Redefine	Fügt eine Redefine-Beziehung hinzu.
	Restriction	Fügt eine Restriction-Beziehung hinzu.

	Extension	Fügt eine Extension-Beziehung hinzu.
	Substitution	Fügt eine Substitution -Beziehung hinzu.
	Kommentar	Fügt einen Kommentar hinzu. Kommentare werden in Annotationen konvertiert, wenn Sie anhand des Modells die Schema-Datei generieren. Durch Auswahl des gewünschten Stereotyps aus dem Fenster "Eigenschaften" können Sie den Annotationstyp definieren.
	Anmerkung	Fügt eine erklärende Anmerkung hinzu.
	Anmerkung verknüpfen	Verknüpft eine Anmerkung mit einem anderen Element im Diagramm.

Eine schrittweise Anleitung zur Modellieung eines Schemas finden Sie unter Beispiel: Erstellen und Generieren eines XML-Schemas 444.

8.3.1.3 Beispiel: Erstellen und Generieren eines XML-Schemas

In diesem Beispiel wird beschrieben, wie Sie mit UModel Schritt für Schritt ein neues XML-Schema modellieren. Nachdem Sie das Schema grafisch mittels UML modelliert haben, werden Sie die Schema-Datei generieren. Dabei lernen Sie, wie Sie das im Codefragment unten gezeigte Schema **product.xsd** erstellen und generieren.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.altova.com/umodel"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:prod="http://www.altova.com/umodel">
   <xs:simpleType name="SizeType">
      <xs:restriction base="xs:integer">
         <xs:maxInclusive value="10"/>
         <xs:minInclusive value="1"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:complexType name="ProductType">
      <xs:sequence>
         <xs:element name="number" type="xs:integer">
         </xs:element>
         <xs:element name="size" type="prod:SizeType">
         </xs:element>
      </xs:sequence>
      <xs:attribute name="createdAt" type="xs:date">
      </xs:attribute>
   </xs:complexType>
   <xs:element name="product" type="prod:ProductType">
   </xs:element>
</xs:schema>
```

product.xsd

Wie oben gezeigt, hat das Schema **product.xsd** zwei Namespace-Deklarationen:

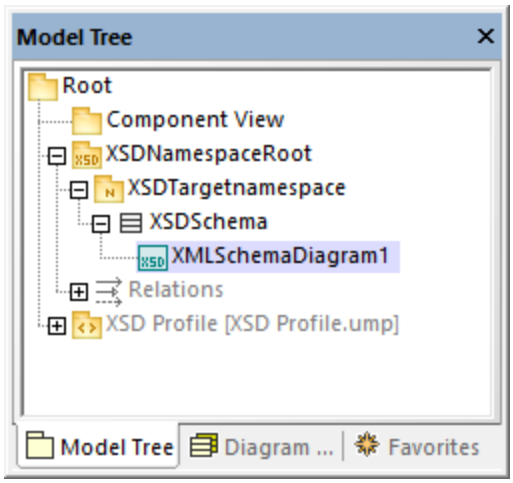
 © 2016-2022 Altova GmbH

1. den XML-Schema-Standard-Namespace `http://www.w3.org/2001/XMLSchema`, der auf das Präfix "xs" gemappt ist.
2. den sekundären Namespace `http://www.altova.com/umodel`, der auf das Präfix "prod" gemappt ist, welcher auch der Target Namespace ist.

Des Weiteren hat das XML-Schema ein globales `product`-Element, einen complexType `ProductType` und einen simpleType `SizeType`.

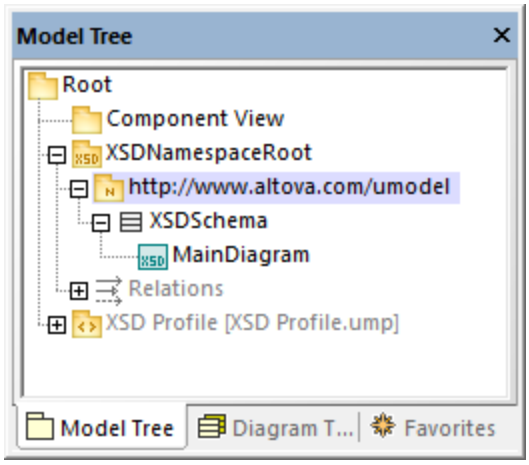
Deklarieren von Namespaces und Dateikodierung

Um fortzufahren, erstellen Sie ein neues UModel-Projekt. Klicken Sie mit der rechten Maustaste auf das **Root**-Paket und wählen Sie im Kontextmenü den Befehl **Neues Diagramm | XML-Schema-Diagramm**. Wenn Sie aufgefordert werden, das UModel XSD-Profil zu inkludieren, klicken Sie auf **OK**.



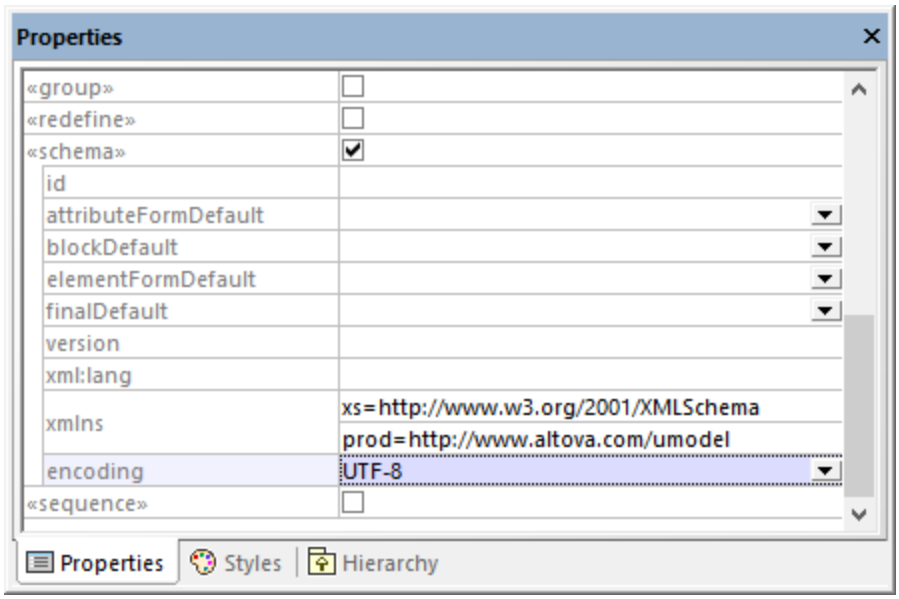
Benennen Sie "XMLSchemaDiagram1" im Fenster Modell-Struktur 80 in "MainDiagram" um. Dies ist das Diagramm, in dem mit Ausnahme der Namespace-Deklarationen die meisten Schema-Komponenten erstellt werden.

Benennen Sie als nächstes "XSDTargetNamespace" in "http://www.altova.com/umodel" um (da dies der erforderliche Target Namespace ist). Damit wird der Target Namespace des neuen Schemas deklariert.



Die beiden "xmlns" Namespaces und die UTF-8-Kodierung können folgendermaßen definiert werden:

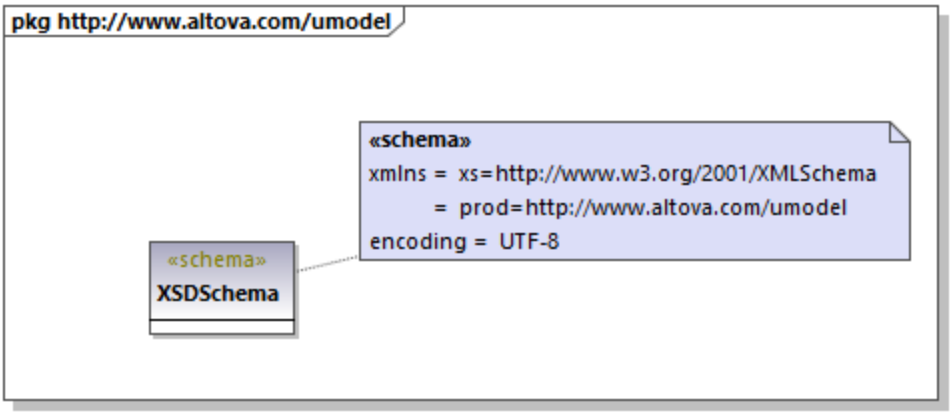
1. Wählen Sie in der Modell-Struktur das **XSDSchema** aus.
2. Klicken Sie im Fenster "Eigenschaften" mit der rechten Maustaste auf die Eigenschaft **xmlns** und wählen Sie den Befehl **Eigenschaftswert hinzufügen | xlmns**.
3. Bearbeiten Sie die Eigenschaften **xmlns** und **Kodierung** wie unten gezeigt.



Optional können Sie schnell auf Namespace-Ebene ein neues XML-Schema-Diagramm erstellen, das dieselben Informationen folgendermaßen visuell darstellt:

1. Klicken Sie in der Modell-Struktur mit der rechten Maustaste auf den Namespace "http://www.altova.com/umodel" und wählen Sie im Kontextmenü den Befehl **Neues Diagramm | XML-Schema-Diagramm**.
2. Wenn ein Meldungsfeld mit dem folgenden Text erscheint: *"Möchten Sie das 'XML-Schema-Diagramm' zu einem neuen 'XSD-Schema' hinzufügen?"*, klicken Sie auf **Nein**.
3. Ziehen Sie das XML-Schema aus der Modell-Struktur in das Diagramm.

 © 2016-2022 Altova GmbH

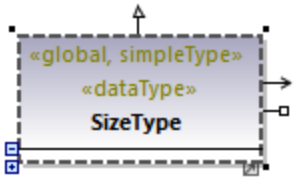


Der Namespace und die Kodierung werden, wie oben gezeigt, als Eigenschaftswerte 146 gespeichert und können auch über das Diagrammfenster bearbeitet werden.

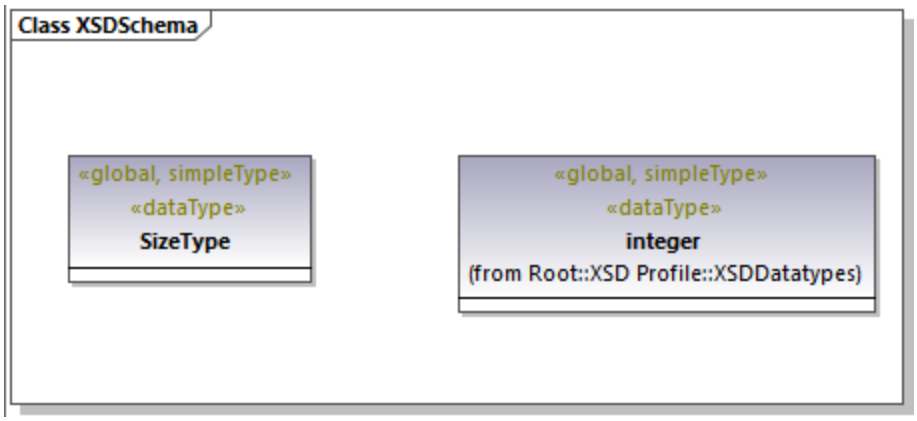
Hinzufügen eines simpleType

In den folgenden Schritten wird der simpleType `SizeType` zum XML-Schema hinzugefügt. Dies ist ein Typ, der den Basistyp `xs:integer` einschränkt, daher fügen wir auch den Basistyp zum Diagramm hinzu und erstellen eine Restriction-Beziehung.

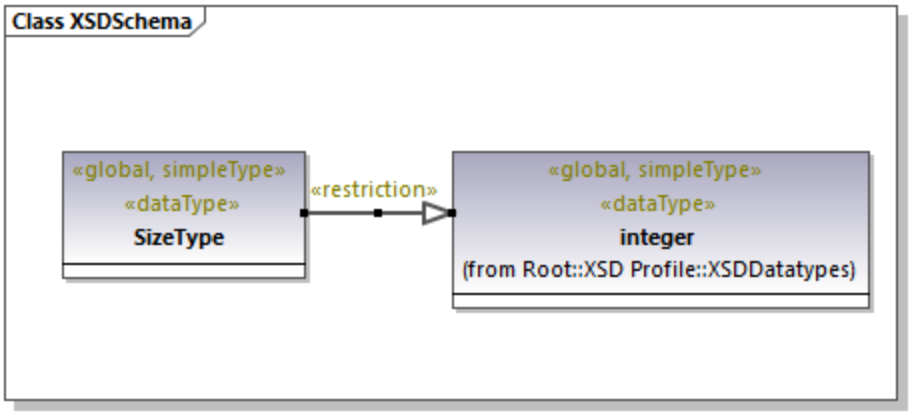
1. Doppelklicken Sie in der Modell-Struktur auf **MainDiagram**, um es zu öffnen.
2. Klicken Sie auf die Symbolleisten-Schaltfläche **XSD Simple Type** und anschließend in das Diagramm.
3. Benennen Sie den neu hinzugefügten SimpleType in `SizeType` um.



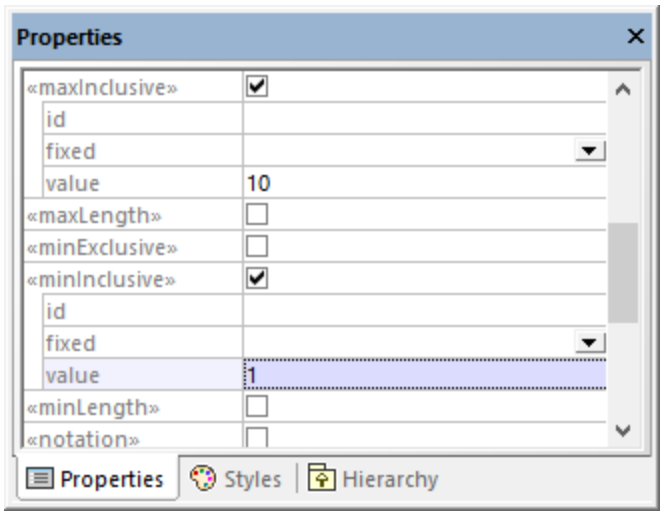
4. Klicken Sie in die Modell-Struktur und drücken Sie **Strg + F.**. Daraufhin wird das Dialogfeld "Suchen" aufgerufen. Geben Sie die ersten Buchstaben von "integer" ein und wählen Sie den Typ `integer` aus dem "XSDDataTypes"-Paket des XSD-Profils aus.
5. Ziehen Sie den Typ `integer` in das Diagramm.



6. Klicken Sie auf die Symbolleisten-Schaltfläche **Restriction** und ziehen Sie den Cursor von `SizeType` auf `integer`. Dadurch wird die Restriction-Beziehung erstellt, siehe auch Erstellen von Beziehungen 133.



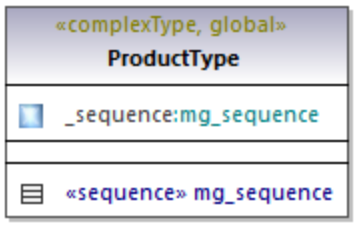
7. Um die Werte `minInclusive` und `maxInclusive` zu definieren, wählen Sie den SimpleType aus und bearbeiten Sie die Eigenschaften desselben Namens im Fenster "Eigenschaften".



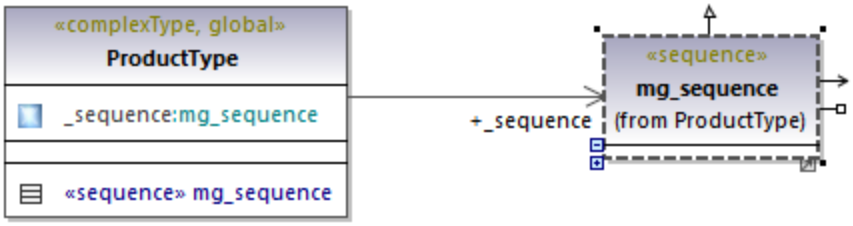
Hinzufügen eines complexType

In den folgenden Schritten wird der complexType `ProductType` zum XML-Schema hinzugefügt. Alle diese Schritte werden ebenfalls im **MainDiagram** durchgeführt.

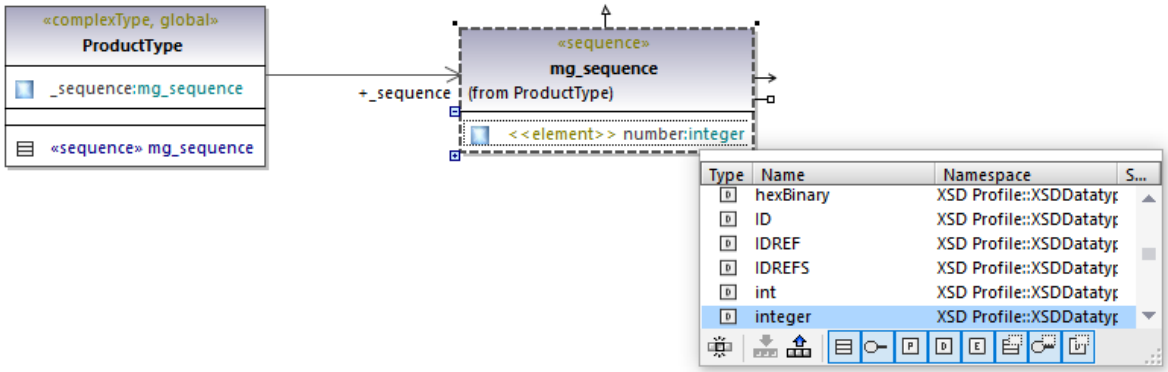
1. Klicken Sie auf die Symbolleisten-Schaltfläche **XSD Complex Type** und anschließend in das Diagramm.
2. Benennen Sie den ComplexType in `ProductType` um.
3. Klicken Sie mit der rechten Maustaste auf den ComplexType und wählen Sie im Kontextmenü den Befehl **Neu | XSD Sequence**.



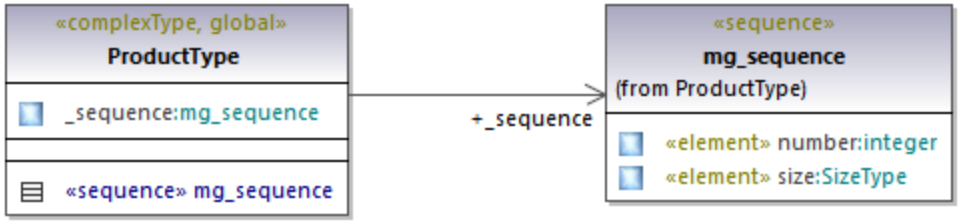
4. Ziehen Sie die Klasse «sequence» aus dem ComplexType in das Diagramm.



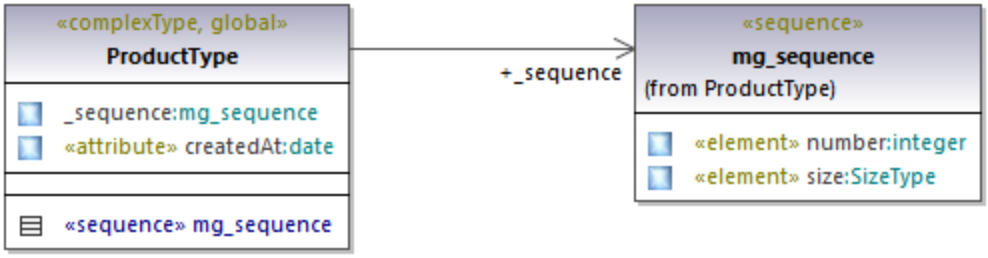
5. Rechtsklicken Sie auf die Sequenz und wählen Sie **Neu | XSD Element (local)**.
6. Ändern Sie den Elementnamen in **number** und definieren Sie als Typ `integer`. Der Typ `integer` ist ein XML-Schema-Basistyp aus dem XSD-Profil. Eine Anleitung zum Definieren des Typs eines Elements finden Sie unter Typ-Autokomplettierung in Klassen 130.



7. Erstellen Sie auf dieselbe Art, wie oben beschrieben, das Element **size** vom Typ `SizeType`. Beachten Sie, dass `SizeType` der zuvor erstellte SimpleType ist.



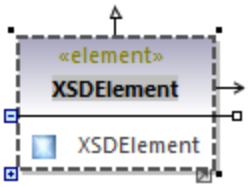
8. Klicken Sie mit der rechten Maustaste im Diagramm auf den ComplexType und wählen Sie im Kontextmenü den Befehl Neu | XSD Attribute (local).
9. Ändern Sie den Attributnamen in **createdAt** und definieren Sie als Typ `date`.



Hinzufügen eines Elements

Nachdem wir nun alle erforderlichen Typen des Schemas definiert haben, können Sie folgendermaßen ein Produktelement vom Typ `ProductType` hinzufügen:

1. Klicken Sie auf die Symbolleisten-Schaltfläche **XSD Element (global)** und anschließend in das Diagramm. Beachten Sie, dass eine Klasse mit dem Stereotyp `«element»` und einer einzigen Eigenschaft hinzugefügt wird.

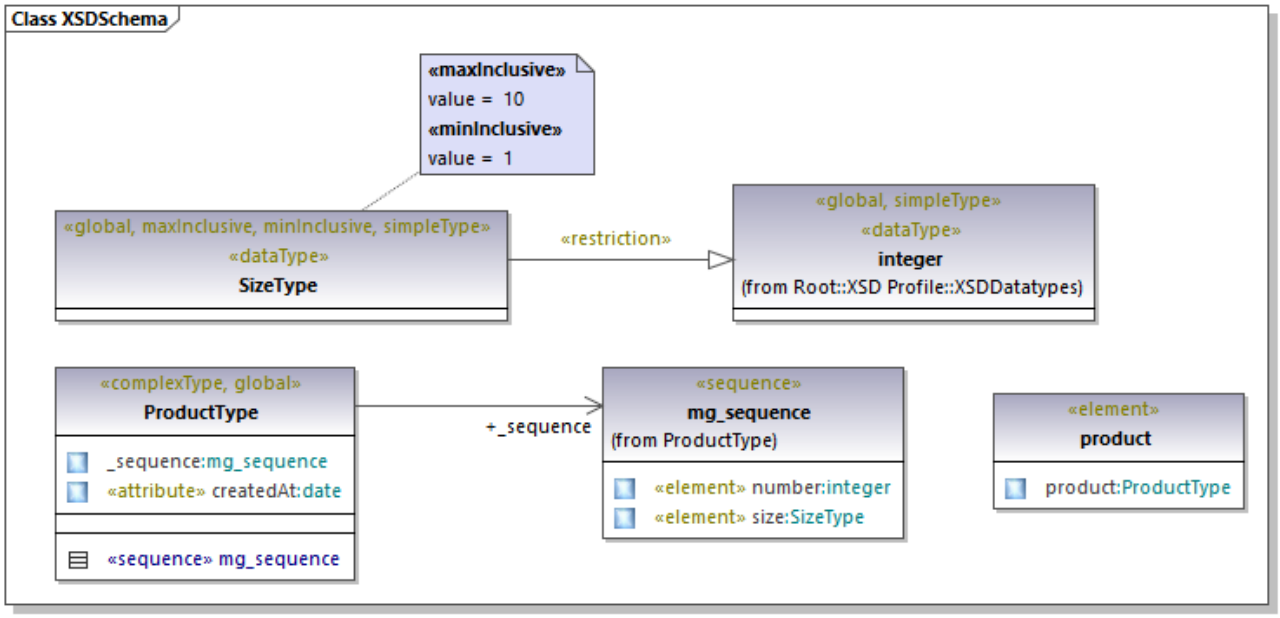


2. Benennen Sie die Eigenschaft in **product** um und ändern Sie ihren Typ in `ProductType`.

Fertiges Design

Mit den obigen Schritten haben wir den Design-Teil des Schemas abgeschlossen. Ihr komplettes Schema-Design sollte nun folgendermaßen aussehen:

 © 2016-2022 Altova GmbH

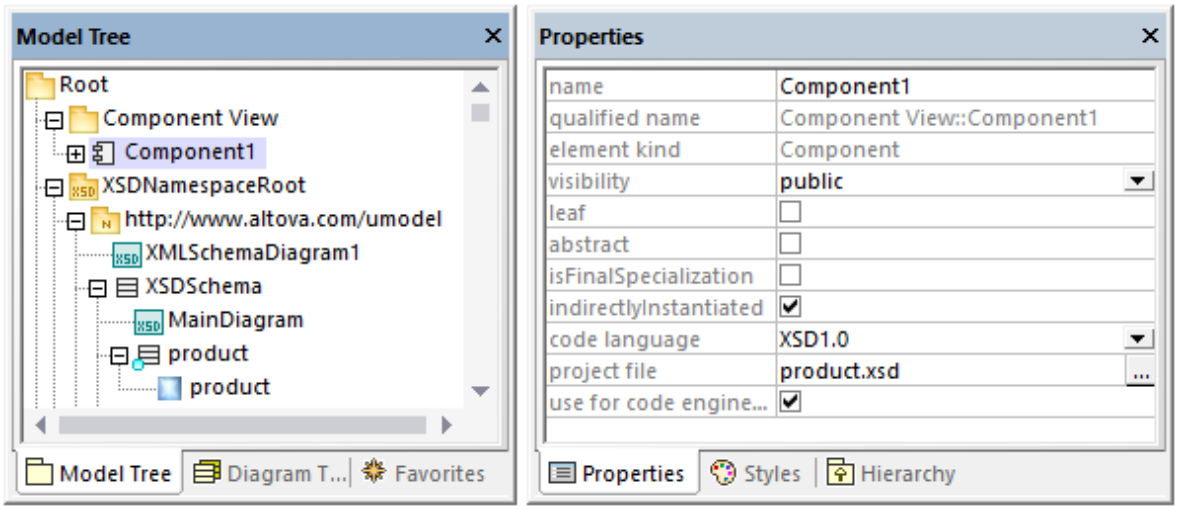


Vorbereitung für das Code Engineering

Damit anhand des Modells eine Schema-Datei generiert werden kann, wollen wir nun eine Code Engineering-Komponente hinzufügen, die die Informationen für die Schemagenerierung liefert. Die Code Engineering-Komponente ähnelt anderen UModel-Projektarten, siehe auch Hinzufügen einer Code Engineering-Komponente 171.

Klicken Sie in der Modell-Struktur mit der rechten Maustaste auf das Paket "Component View" und fügen Sie ein neues Element vom Typ **Komponente** hinzu. Die Eigenschaften der Komponente müssen nun wie unten gezeigt geändert werden:

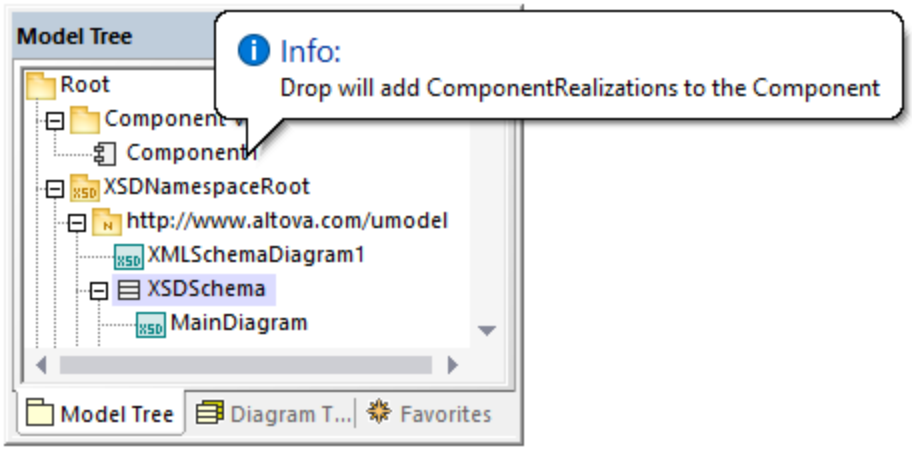
1. Die Eigenschaft **für Code Engineering verwenden** muss aktiviert werden.
2. Die Eigenschaft **Codesprache** der Code Engineering-Komponente muss auf "XSD 1.0" gesetzt werden.
3. Die Eigenschaft **Projektdatei** der Code Engineering-Komponente muss auf die zu generierende Schema-Datei (in diesem Beispiel **product.xsd**) verweisen.



Anmerkung: Wenn die Eigenschaft **Projektdatei** fehlt, geben Sie in die Eigenschaft **Verzeichnis** **product.xsd**ein und drücken Sie die **Eingabetaste**. Daraufhin wird ein Meldungsfeld angezeigt, in dem Sie aufgefordert werden, statt dessen eine Projektdatei zu referenzieren. Klicken Sie zur Bestätigung auf **Ja**.

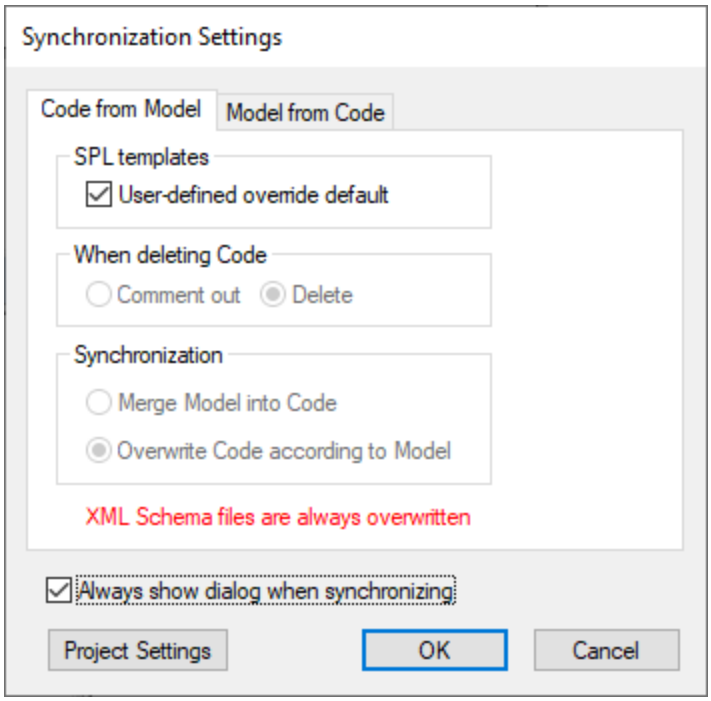
Schließlich muss das XML-Schema von der Code Engineering-Komponente wie unter Hinzufügen einer Code Engineering-Komponente 171 beschrieben, realisiert werden. Die schnellste Art, um die **Komponentenrealisierungsbeziehung** in diesem Beispiel zu erstellen, ist die folgende:

- Ziehen Sie das **XSDSchema**-Schema in der Modell-Struktur über die Code Engineering-Komponente (**Component1**) und lassen Sie die Maustaste los, wenn ein Tooltipp wie der folgende angezeigt wird:



Sie können nun die Schema-Datei generieren. Drücken Sie dazu entweder **F12** oder wählen Sie den Menübefehl **Projekt | Überschreibe Programmcode von UModel Projekt**. Beachten Sie, dass eine Zusammenführung im Fall von XML-Schemas nicht unterstützt wird, daher wird im Dialogfeld eine Meldung in Rot mit diesem Inhalt angezeigt.

 © 2016-2022 Altova GmbH



Das neue XML-Schema wird im selben Ordner wie Ihr UModel-Projekt generiert.

9 Austausch von Metadaten zwischen XMI und XML

Altova Website: Austausch von UModel-Projekten über XMI

Sie können UModel-Projekte in XML Metadata Interchange (XMI)-Dateien exportieren und XMI-Dateien als UModel-Projekte importieren. Dies ermöglicht die Interoperabilität mit anderen UML-Tools, die XMI unterstützen. Die folgenden XMI-Versionen werden unterstützt:

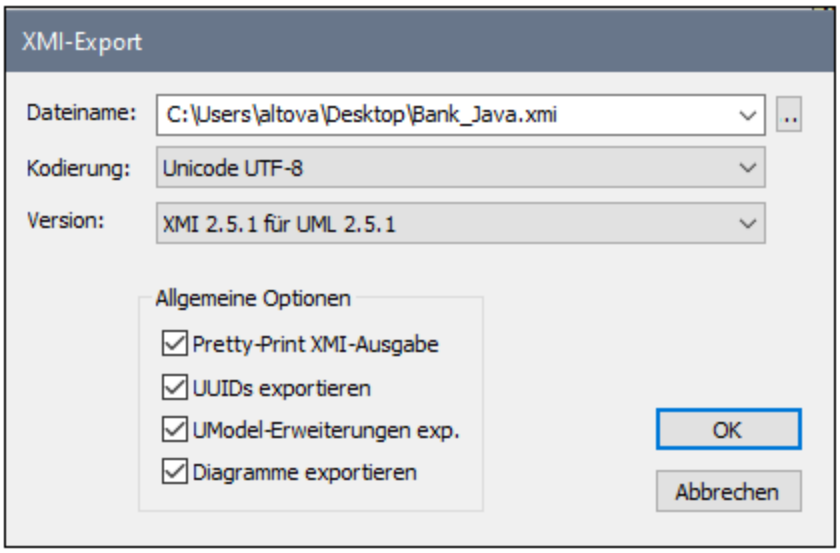
- XMI 2.1 für UML 2.0
- XMI 2.1 für UML 2.1.2
- XMI 2.1 für UML 2.2
- XMI 2.1 für UML 2.3
- XMI 2.4.1 für UML 2.4.1
- XMI 2.4.1 für UML 2.5
- XMI 2.5.1 für UML 2.5.1

So importieren Sie eine XMI-Datei in UModel:

- Klicken Sie im Menü **Datei** auf **Aus XMI-Datei importieren**.

So exportieren Sie ein UModel-Projekt in eine XMI-Datei:

- Klicken Sie im Menü **Datei** auf **In XMI-Datei exportieren**.



Anmerkungen:

- Beim Export inkludierte Dateien werden selbst solche, die als "Durch Referenz 166" definiert werden, ebenfalls exportiert.
- Wenn Sie beabsichtigen, generierten XMI-Code wieder in UModel zu importieren, stellen Sie sicher, dass Sie das Kontrollkästchen **UModel-Erweiterungen exportieren** aktivieren.

 © 2016-2022 Altova GmbH

Im nachstehenden Abschnitt werden die Optionen, die beim Export von Projekten in XMI zur Verfügung stehen, beschrieben.

Pretty-Print XMI-Ausgabe

Wenn Sie diese Option aktivieren, wird die XMI-Datei mit Einrückung der XML-Tags und mit Zeilenschaltungen ausgegeben.

UUIDs exportieren

In XMI sind drei Elementidentifikationensversionen definiert: IDs, UUIDs und Labels.

- IDs sind innerhalb des XMI-Dokuments eindeutig und werden von den meisten UML-Tools unterstützt. UModel exportiert standardmäßig diese ID-Typen, d.h. keines der Kontrollkästchen muss aktiviert werden.
- UUID sind Universally Unique Identifiers und bieten eine Methode, um jedem Element eine GUID (Global Unique Identification) zuzuweisen, d.h. UUIDs sind nicht auf bestimmte XMI-Dokumente beschränkt. UUIDs werden durch Auswahl des Kontrollkästchens "UUIDs exportieren" generiert.
- UUIDs werden im Standardformat UUID/GUID gespeichert (z.B. "6B29FC40-CA47-1067-B31D-00DD010662DA", "550e8400-e29b-41d4-a716-446655440000",...)
- Labels werden von UModel nicht unterstützt.

Anmerkung: Beim XMI-Importvorgang werden automatisch beide ID-Typen unterstützt.

UModel-Erweiterungen exportieren

In XMI ist ein "Erweiterungsmechanismus" definiert, mit Hilfe dessen jede Applikation ihre toolspezifischen Erweiterungen zur UML-Spezifikation exportieren kann. Andere UML-Tools können nur die Standard-UML-Daten importieren (die UModel-Erweiterungen werden ignoriert). Diese UModel-Erweiterungsdaten stehen nur bei Import in UModel zur Verfügung.

Daten wie z.B. die Dateinamen von Klassen oder Elementfarben sind nicht Teil der UML-Spezifikation und müssen daher in XMI gelöscht oder unter "Extensions" gespeichert werden. Wenn sie als Erweiterungen exportiert wurden und wieder importiert werden, werden alle Dateinamen und Farben, wie definiert, importiert. Wenn für den Export keine Erweiterungen verwendet werden, gehen diese UModel-spezifischen Daten verloren.

Beim Import eines XMI-Dokuments wird das Format automatisch ermittelt und das Modell generiert.

Diagramme exportieren

Exportiert UModel-Diagramme als "Erweiterungen" in die XMI-Datei. Die Option **UModel-Erweiterungen exportieren** muss aktiv sein, damit das Diagramm als Erweiterung gespeichert werden kann.

10 Versionskontrolle

Die Versionskontrollunterstützung in UModel steht über die Microsoft Source Control Plug-in API (vormals bekannt als MSSCCI API), Version 1.1, 1.2 sowie 1.3 zur Verfügung. Dadurch können Sie Versionskontrollbefehle wie z.B. "Einchecken" oder "Auschecken" direkt über UModel an praktisch jedem Versionskontrollsystem ausführen, das nativen Clients oder Drittanbieter-Clients über das Microsoft Source Control Plug-in API die Verbindung zu UModel gestattet.

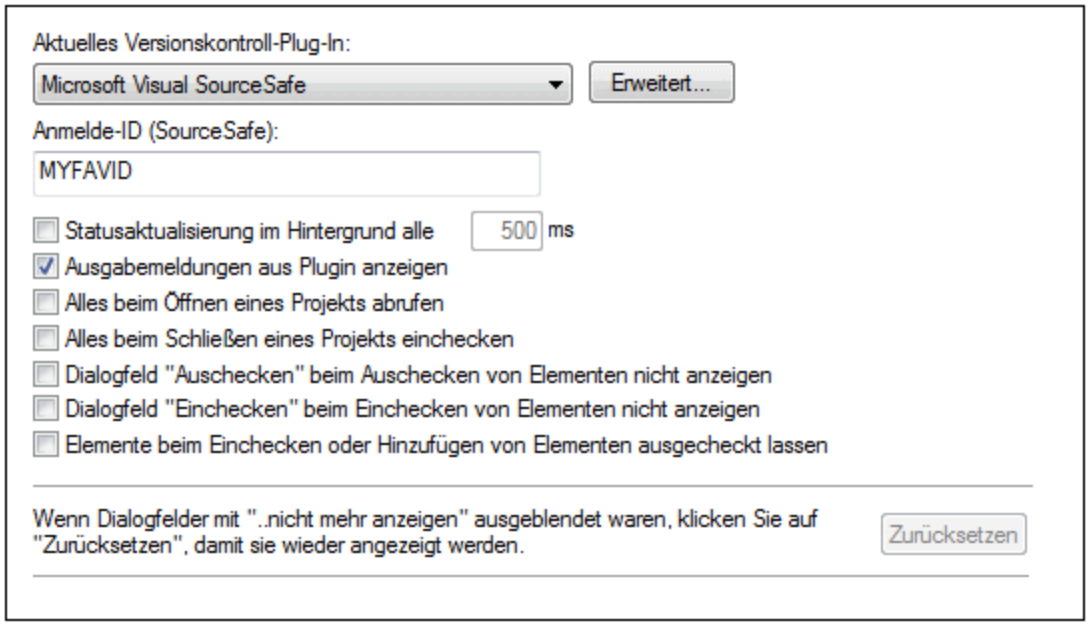
Als Versionskontrollsystemanbieter kann jedes kommerzielle oder nicht kommerzielle Plug-in, das die Microsoft Source Control Plug-in API unterstützt und mit einem kompatiblen Versionskontrollsystem verbunden werden kann, verwendet werden. Eine Liste von von Altova getesteten Versionskontrollsystemen und Plug-ins finden Sie unter Unterstützte Versionskontrollsysteme 459.

Installieren und Konfigurieren des Versionskontrollanbieters

Um die auf Ihrem System verfügbaren Versionskontrollsysteme anzuzeigen, gehen Sie folgendermaßen vor:

1. Klicken Sie im Menü **Extras** auf **Optionen**.
2. Klicken Sie auf das Register **Versionskontrolle**.

Alle mit der Microsoft Source Code Control Plug-in API kompatiblen Versionskontroll-Plug-ins werden in der Dropdown-Liste **Aktuelles Versionskontroll-Plug-in** angezeigt.



Wenn auf Ihrem System kein kompatibles Plug-in gefunden wurde, wird die folgende Meldung angezeigt:

```
"Die Registrierung des installierten Versionskontrollproviders konnte nicht gefunden werden
oder ist unvollständig."
```

Bei einigen Versionskontrollsystemen wird das Versionskontroll-Plug-in nicht automatisch installiert. In diesem Fall müssen Sie es gesondert installieren. Eine genauere Anleitung dazu finden Sie in der Dokumentation zum

 © 2016-2022 Altova GmbH

jeweiligen Versionskontrollsystem. Ein Plug-in (Anbieter), das mit der Microsoft Source Code Control Plug-in API kompatibel ist, sollte auf Ihrem Betriebssystem unter dem folgenden Registrierdateieintrag zu finden sein:

```
HKEY_LOCAL_MACHINE\SOFTWARE\SourceCodeControlProvider\InstalledSCCProviders
```

Wenn das Plug-in korrekt installiert wurde, steht es automatisch in der Liste der Plug-ins in UModel zur Verfügung.

Aufrufen der Versionskontrollbefehle

Die Versionskontrollbefehle stehen über das Menü **Projekt | Versionskontrolle** zur Verfügung.

Probleme im Zusammenhang mit den Ressourcen / der Geschwindigkeit

Bei sehr großen Versionskontrolldatenbanken kann die automatische Durchführung von Statusaktualisierungen im Hintergrund etwas länger dauern.

Um schneller arbeiten zu können, können Sie versuchen, auf dem Register "Versionskontrolle" (Aufruf über **Extras | Optionen**) das Feld "**Statusaktualisierung im Hintergrund alle xxx Sek**." zu deaktivieren oder das Aktualisierungsintervall zu vergrößern.

Anmerkung: Die **64-Bit**-Version Ihrer Altova-Applikation unterstützt automatisch alle in dieser Dokumentation aufgelisteten 32-Bit-Versionskontrollsysteme. Bei Verwendung einer 64-Bit-Altova-Applikation zusammen mit einem 32-Bit-Versionskontrollsystem ist die Option **Statusaktualisierung im Hintergrund alle xxx Sek** automatisch deaktiviert und kann nicht ausgewählt werden.

Vergleich mit Altova DiffDog

Sie können viele Versionskontrollsysteme (einschließlich Git und TortoiseSVN) für die Verwendung mit Altova DiffDog als Vergleichstool konfigurieren. Nähere Informationen zu DiffDog finden Sie unter https://www.altova.com/de/diffdog.html. Die Dokumentation zu DiffDog finden Sie unter https://www.altova.com/de/documentation.html.

10.1 Einrichten der Versionskontrolle

So richten Sie eine Versionskontrolle ein und stellen Dateien in einem UModel-Projekt unter eine Versionskontrolle:

1. Installieren Sie ein Versionskontrollprogramm (siehe Unterstützte Versionskontrollsysteme 459), falls noch keines installiert ist. Richten Sie die Versionskontrolldatenbank (das Repository), in der Sie Ihre Arbeit speichern möchten, ein.
2. Erstellen Sie einen Ordner für den lokalen Arbeitsbereich, der die Arbeitsdateien enthalten soll, die unter Versionskontrolle gestellt werden sollen. Der Ordner, der alle Ihre Arbeitsbereichordner und -Dateien enthält, wird als ihr lokaler Ordner und der Pfad zum lokalen Ordner als der lokale Pfad bezeichnet. Der Ordner wird an einen bestimmten Ordner im Repository gebunden.
3. Erstellen Sie in Ihrer Altova-Applikation einen Applikationsprojektordner, zu dem Sie die unter Versionskontrolle zu stellenden Dateien hinzufügen müssen. Diese Gliederung der Dateien in einem Applikationsprojekt ist abstrakt. Die Dateien in einem Projekt referenzieren physische lokal (vorzugsweise in einem einzigen Ordner, falls nötig mit Unterordnern) gespeicherte Dateien.
4. In der Datenbank des Versionskontrollsystems (die auch als Versionskontrolle oder Repository bezeichnet wird) wird ein Ordner erstellt, der an den lokalen Ordner gebunden ist. Dieser (als gebundener Ordner bezeichnete) Ordner repliziert die Struktur des lokalen Ordners, sodass sich alle unter Versionskontrolle zu stellenden Dateien an der richtigen Stelle im gebundenen Ordner befinden. Der gebundene Ordner wird normalerweise erstellt, wenn Sie zum ersten Mal eine Datei oder ein Applikationsprojekt zur Versionskontrolle hinzufügen.

 © 2016-2022 Altova GmbH

10.2 Unterstützte Versionskontrollsysteme

In der nachfolgenden Liste sind die von UModel unterstützten Versionskontrollsysteme zusammen mit ihren entsprechenden Versionskontroll-Clients (SCCs) aufgelistet. Die Liste ist alphabetisch nach Versionskontrollsystem geordnet.

- Altova hat in UModel die Microsoft Source Control Plug-in API (Version 1.1, 1.2 und 1.3) implementiert und die Unterstützung für die aufgelisteten Treiber und Versionskontrollsysteme getestet. Aller Voraussicht nach wird UModel diese Produkte, falls diese aktualisiert werden, weiterhin unterstützen.
- Auch Versionskontroll-Clients, die nicht in der nachstehenden Liste erwähnt sind, die aber die Microsoft Source Control Plug-in API implementieren, sollten ebenfalls mit UModel funktionieren.

Versionskontrollsystem	Versionskontroll-Clients
AccuRev 4.7.0 Windows	AccuBridge for Microsoft SCC 2008.2
Bazaar 1.9 Windows	Aigenta Unified SCC 1.0.6
Borland StarTeam 2008	Borland StarTeam Cross-Platform Client 2008 R2
Codice Software Plastic SCM Professional 2.7.127.10 (Server)	Codice Software Plastic SCM Professional 2.7.127.10 (SCC Plugin)
Collabnet Subversion 1.5.4	• Aigenta Unified SCC 1.0.6 • PushOK SVN SCC 1.5.1.1 • PushOK SVN SCC x64 Version 1.6.3.1 • TamTam SVN SCC 1.2.24
ComponentSoftware CS-RCS (PRO) 5.1	ComponentSoftware CS-RCS (PRO) 5.1
Dynamsoft SourceAnywhere for VSS 5.3.2 Standard/Professional Server	Dynamsoft SourceAnywhere für VSS 5.3.2 Client
Dynamsoft SourceAnywhere Hosted	Dynamsoft SourceAnywhere Hosted Client (22252)
Dynamsoft SourceAnywhere Standalone 2.2 Server	Dynamsoft SourceAnywhere Standalone 2.2 Client
Git	PushOK GIT SCC Plug-in (siehe Versionskontrolle mit Git 482)
IBM Rational ClearCase 7.0.1 (LT)	IBM Rational ClearCase 7.0.1 (LT)
March-Hare CVSNT 2.5 (2.5.03.2382)	Aigenta Unified SCC 1.0.6
March-Hare CVS Suite 2008	• Jalindi Igloo 1.0.3 • March-Hare CVS Suite Client 2008 (3321) • PushOK CVS SCC NT 2.1.2.5 • PushOK CVS SCC x64 Version 2.2.0.4 • TamTam CVS SCC 1.2.40
Mercurial 1.0.2 für Windows	Sergey Antonov HgSCC 1.0.1

Versionskontrollsystem	Versionskontroll-Clients
Microsoft SourceSafe 2005 mit CTP	Microsoft SourceSafe 2005 mit CTP
Microsoft Visual Studio Team System 2008/2010 Team Foundation Server	Microsoft Team Foundation Server 2008/2010 MSSCCI Provider
Perforce 2008 P4S 2008.1	Perforce P4V 2008.1
PureCM Server 2008/3a	PureCM Client 2008/3a
QSC Team Coherence Server 7.2.1.35	QSC Team Coherence Client 7.2.1.35
Reliable Software Code Co-Op 5.1a	Reliable Software Code Co-Op 5.1a
Seapine Surround SCM Client/Server für Windows 2009.0.0	Seapine Surround SCM Client 2009.0.0
Serena Dimensions Express/CM 10.1.3 für Win32 Server	Serena Dimensions 10.1.3 for Win32 Client
Softimage Alienbrain Server 8.1.0.7300	Softimage Alienbrain Essentials/Advanced Client 8.1.0.7300
SourceGear Fortress 1.1.4 Server	SourceGear Fortress 1.1.4 Client
SourceGear SourceOffsite Server 4.2.0	SourceGear SourceOffsite Client 4.2.0 (Windows)
SourceGear Vault 4.1.4 Server	SourceGear Vault 4.1.4 Client
VisualSVN Server 1.6	• Aigenta Unified SCC 1.0.6 • PushOK SVN SCC 1.5.1.1 • PushOK SVN SCC x64 Version 1.6.3.1 • TamTam SVN SCC 1.2.24

© 2016-2022 Altova GmbH

10.3 Versionskontrollbefehle

Die Versionskontrollbefehle von UModel werden in den folgenden Abschnitten anhand von Visual SourceSafe angezeigt. In den Beispielen in diesem Abschnitt wird das UModel-Projekt **Bank_CSharp.ump** (und damit verknüpfte Codedateien) aus dem Ordner C:\Benutzer\<Benutzername>\Dokumente\Altova\UModel2023\UModelExamples verwendet. Beachten Sie, dass ein Versionskontrollprojekt nicht dasselbe ist wie ein UModel-Projekt. Versionskontrollprojekte sind verzeichnisunabhängig, während UModel-Projekte logische Konstruktionen ohne direkte Verzeichnisabhängigkeit sind.

Gehen Sie folgendermaßen vor, um die Versionskontrollbefehle aufzurufen:

- Verwenden Sie den Menübefehl **Projekt | Versionskontrolle**
- Verwenden Sie das **Kontextmenü** in der Modell-Struktur
- Klicken Sie auf die Schaltflächen der Versionskontroll-Symbolleiste. Die Symbolleiste wird über **Extras | Anpassen | Symbolleisten** aktiviert.

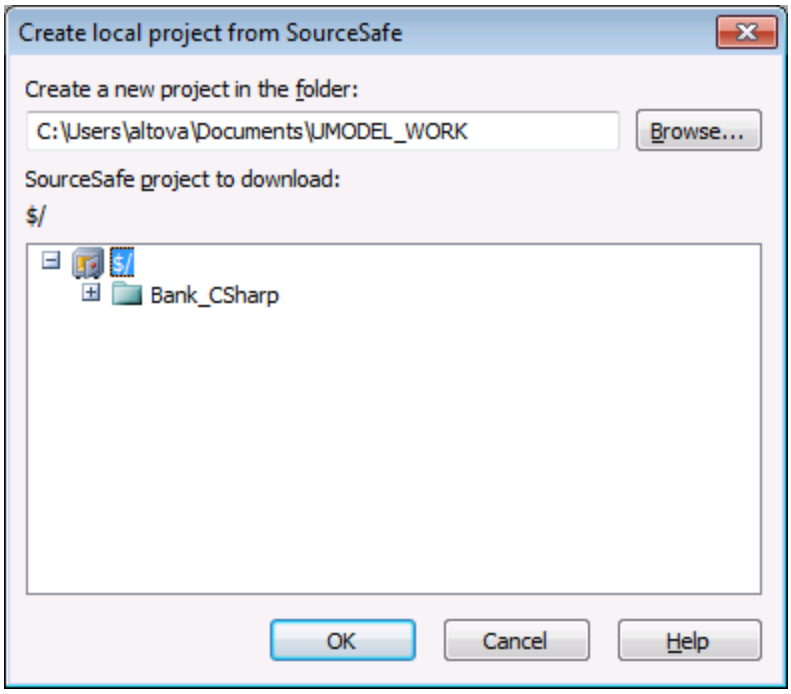
Die Beschreibung der im Folgenden beschriebenen Versionskontrollbefehle gelten für die **Standalone**-Version von UModel. In der Visual Studio und der Eclipse Version von UModel werden die Versionskontrollbefehle und Menübefehle verwendet, die in diesen IDEs zur Verfügung stehen.

Aus Versionskontrolle öffnen 461
Versionskontrolle aktivieren 464
Aktuellste Version holen 465
Abrufen 465
Ordner abrufen 466
Auschecken 468
Einchecken 469
Auschecken rückgängig... 470
Zu Versionskontrolle hinzufügen 471
Von Versionskontrolle ausgliedern 474
Aus Versionskontrolle freigeben 475
Verlauf anzeigen 476
Unterschiede anzeigen 478
Eigenschaften anzeigen 479
Status aktualisieren 480
Versionskontrollmanager 480
Versionskontrolle wechseln 480

10.3.1 Aus Versionskontrolle öffnen

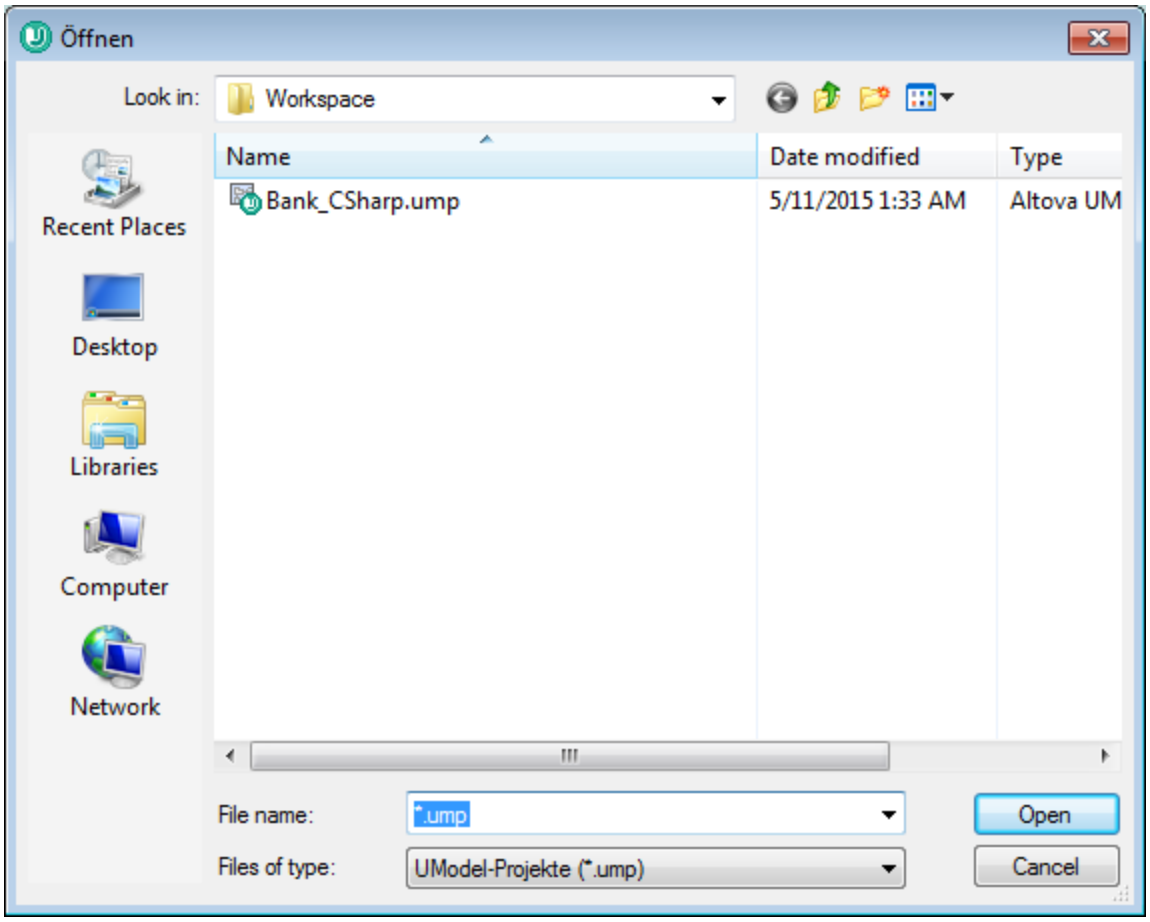
Mit dem Befehl "Aus Versionskontrolle öffnen" wird ein lokales Projekt anhand einer bestehenden Versionskontrolldatenbank erstellt und unter Versionskontrolle - in diesem Fall SourceSafe - gestellt.

1. Wählen Sie **Projekt** | **Versionskontrolle** | **Aus Versionskontrolle öffnen**. Daraufhin wird das Anmeldedialogfeld geöffnet, in dem Sie Ihre Anmeldedaten eingeben, bevor Sie fortfahren können. Daraufhin wird das Dialogfeld "Create local project from SourceSafe" angezeigt.
2. Definieren Sie das Verzeichnis, das das neue lokale Projekt enthalten soll, z.B. c:\temp\ssc. Dies wird zum neuen **Arbeitsverzeichnis** oder dem Auscheck-Ordner.



3. Wählen Sie das gewünschte SourceSafe-Projekt aus, z.B. Bank_CSharp. Wenn der hier von Ihnen definierte Ordner noch nicht existiert, erscheint ein Dialogfeld, in dem Sie aufgefordert werden, ihn zu erstellen.
4. Klicken Sie auf **Ja,** um das neue Verzeichnis zu erstellen. Das Dialogfeld "Öffnen" wird angezeigt.

© 2016-2022 Altova GmbH



5. Wählen Sie die UModel-Projektdatei **Bank_CSharp.ump** aus und klicken Sie auf "Öffnen".

   Bank_CSharp.ump wird nun in UModel geöffnet und die Datei wird unter Versionskontrolle gestellt. Angezeigt wird dies in der Modell-Struktur durch ein Schlosssymbol über dem Root-Ordner. Der Root-Ordner repräsentiert sowohl die Projektdatei als auch das Arbeitsverzeichnis für Versionskontrolloperationen.

   Das Verzeichnis "BankCSharp" wurde lokal erstellt. Sie können nun mit diesen Dateien normal arbeiten.

Anmerkung:

Informationen, wie Sie die beim Synchronisieren von Code generierten **Codedateien** unter Versionskontrolle stellen, finden Sie unter: Hinzufügen zur Versionskontrolle 471

Symbole der Versionskontrolle:

oder

Das Vorhängeschloss-Symbol zeigt an, dass diese Datei oder dieser Ordner **unter Versionskontrolle steht**, aber derzeit nicht ausgecheckt ist.

oder

Das **rote** Kontrollhäkchen zeigt an, dass diese Datei **ausgecheckt** ist, d.h. dass die UModel-Projektdatei (oder Codedatei) zur Bearbeitung ausgecheckt wurde.
Das Sternchen in der Titelleiste der Applikation bedeutet, dass diese Datei verändert wurde. Beim Schließen werden Sie aufgefordert, die Datei zu speichern.

oder

Das Personensymbol zeigt an, dass die Datei(en) von jemand anderem im Netzwerk oder von Ihnen in ein anderes Arbeitsverzeichnis ausgecheckt wurde(n).

10.3.2 Versionskontrolle aktivieren

Mit Hilfe dieses Befehls können Sie die Versionskontrolle für ein UModel-Projekt aktivieren oder deaktivieren. Der Befehl steht über das Projekt-Menü **Projekt | Versionskontrolle | Versionskontrolle aktivieren** zur Verfügung. Bei Auswahl dieser Option für eine Datei oder einen Ordner, wird die Versionskontrolle für das gesamte UModel-Projekt aktiviert/deaktiviert.

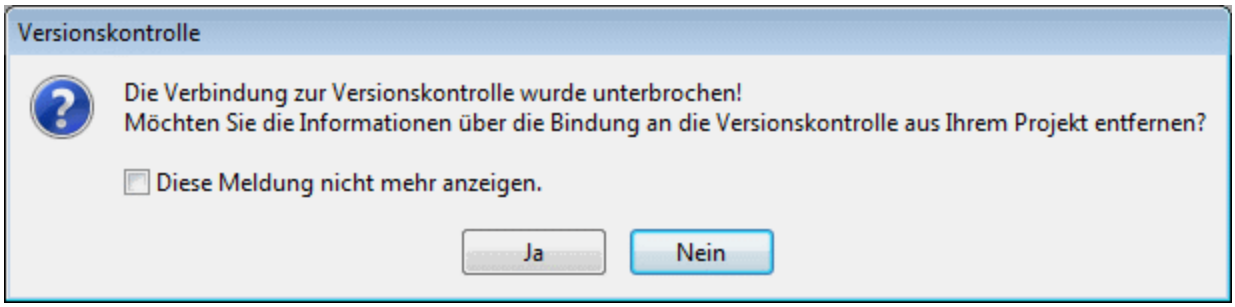
So aktivieren Sie die Versionskontrolle für ein Projekt:

1. Wählen Sie die Menüoption **Projekt | Versionkontrolle** und aktivieren Sie das Kontrollkästchen **Versionskontrolle aktivieren**. Der vorherige Ein-/Auscheck-Status der verschiedenen Dateien wird abgerufen und im Fenster "Modell-Struktur" angezeigt.

So deaktivieren Sie die Versionskontrolle für ein Projekt:

1. Wählen Sie die Menüoption **Projekt | Versionkontrolle** und deaktivieren Sie das Kontrollkästchen **Versionskontrolle aktivieren**.

 © 2016-2022 Altova GmbH



Sie werden nun gefragt, ob Sie die Binding-Informationen aus dem Projekt entfernen möchten.

Um die Versionskontrolle **provisorisch** zu deaktivieren, wählen Sie **Nein**.

Um die Versionskontrolle für dieses Projekt **permanent** zu deaktivieren, wählen Sie **Ja**.

10.3.3 Aktuellste Version holen

Mit diesem Befehl wird die aktuellste Version der ausgewählten Datei(en) aus dem Versionskontrollspeicher in das Arbeitsverzeichnis geholt. Die Dateien werden als schreibgeschützte Dateien und nicht ausgecheckt abgerufen.

Wenn die betroffenen Dateien derzeit ausgecheckt sind, geschehen je nach Version des Versionskontroll-Plugin verschiedene Dinge: Es geschieht nichts, neue Daten werden in Ihrer lokalen Datei zusammengeführt oder Ihre Änderungen werden überschrieben.

Dieser Befehl funktioniert ähnlich wie der Befehl "Abrufen", doch wird das Dialogfeld "Versionskontrolle - Abrufen" nicht angezeigt. Sie können daher keine erweiterten Abrufoptionen definieren.

Beachten Sie, dass dieser Befehl bei Ausführung an einem Ordner automatisch eine rekursive Operation "Aktuellste Version holen" ausführt, d.h. alle anderen Dateien, die sich in der Pakethierarchie unterhalb der aktuellen befinden, sind davon betroffen.

So rufen Sie die neueste Version einer Datei ab:

1. Wählen Sie in der Modell-Struktur die Dateien aus, für die Sie die neueste Version holen möchten.
2. Wählen Sie **Projekt** | **Versionskontrolle** | **Aktuellste Version holen**.

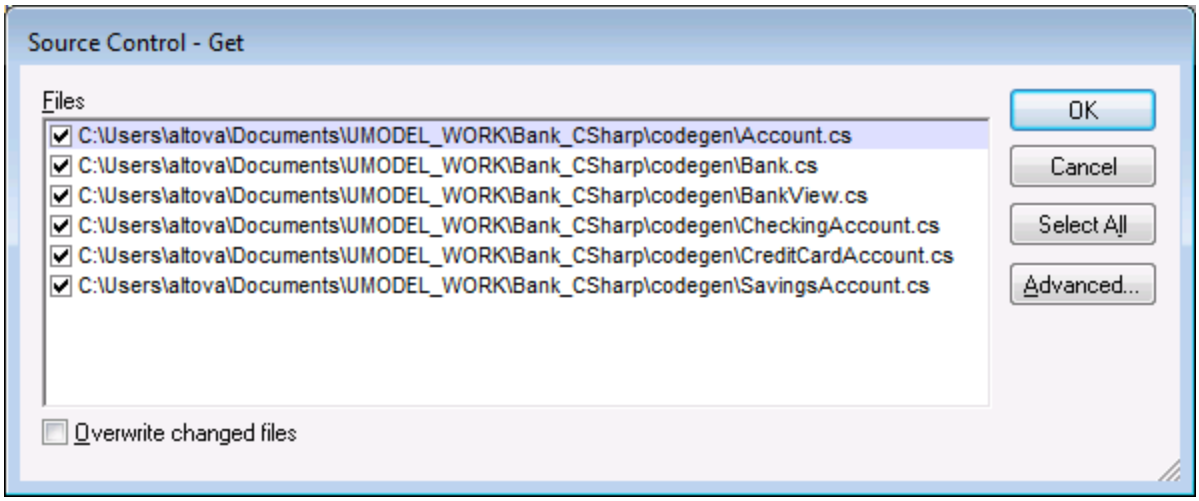
10.3.4 Abrufen

Ruft eine schreibgeschützte Kopie der ausgewählten Dateien ab und platziert Sie in den Arbeitsordner. Standardmäßig sind die Dateien nicht zur Bearbeitung ausgecheckt.

Verwenden von "Abrufen":

- Wählen Sie in der Modell-Struktur die gewünschten Dateien aus.

- Wählen Sie den Befehl **Projekt** | **Versionskontrolle** | **Abrufen.**

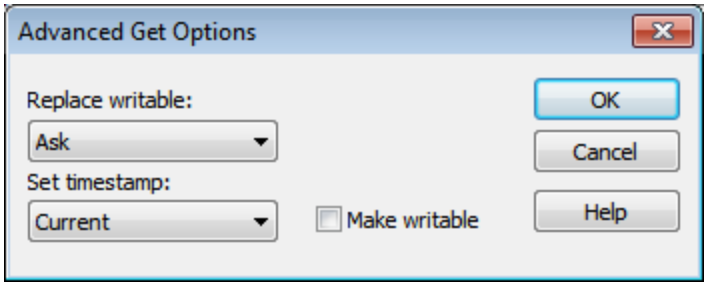


Geänderte Dateien überschreiben
Überschreibt lokal geänderte Dateien mit jenen aus der Versionskontrolldatenbank.

Alle auswählen
Wählt alle Dateien im Listenfeld aus.

Erweitert

Mit Hilfe dieser Option können Sie die Optionen **Replace writable** und **Set timestamp** in den entsprechenden Auswahllisten auswählen.



Mit dem Kontrollkästchen "Make writable" wird das "read-only" Attribut der abgerufenen Dateien entfernt.

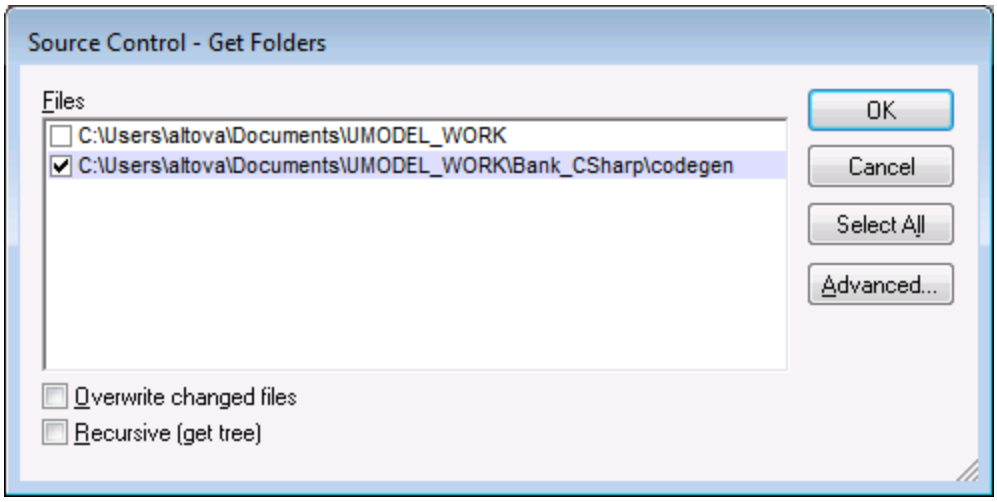
10.3.5 Ordner abrufen

Ruft schreibgeschützte Kopien von Dateien in den ausgewählten Ordnern ab und platziert Sie in den Arbeitsordner. Die Dateien werden standardmäßig nicht zur Bearbeitung ausgecheckt.

 © 2016-2022 Altova GmbH

Verwenden von "Ordner abrufen":

- Wählen Sie in der Modell-Struktur die gewünschten Dateien aus.
- Wählen Sie den Befehl **Projekt** | **Versionskontrolle** | **Abrufen.**

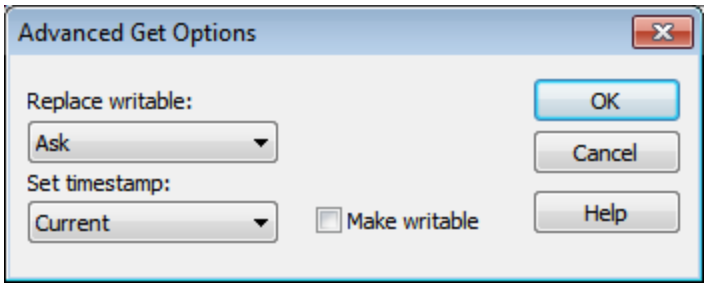


Geänderte Dateien überschreiben
Überschreibt Dateien, die lokal geändert wurden, mit den Dateien aus der Versionskontrolldatenbank.

Rekursiv (Struktur abrufen)
Ruft alle Dateien aus der Ordnerstruktur ab, die sich unterhalb des ausgewählten Ordners befinden.

Erweitert

Mit Hilfe dieser Option können Sie die Optionen **Replace writable** und **Set timestamp** in den entsprechenden Auswahllisten auswählen.



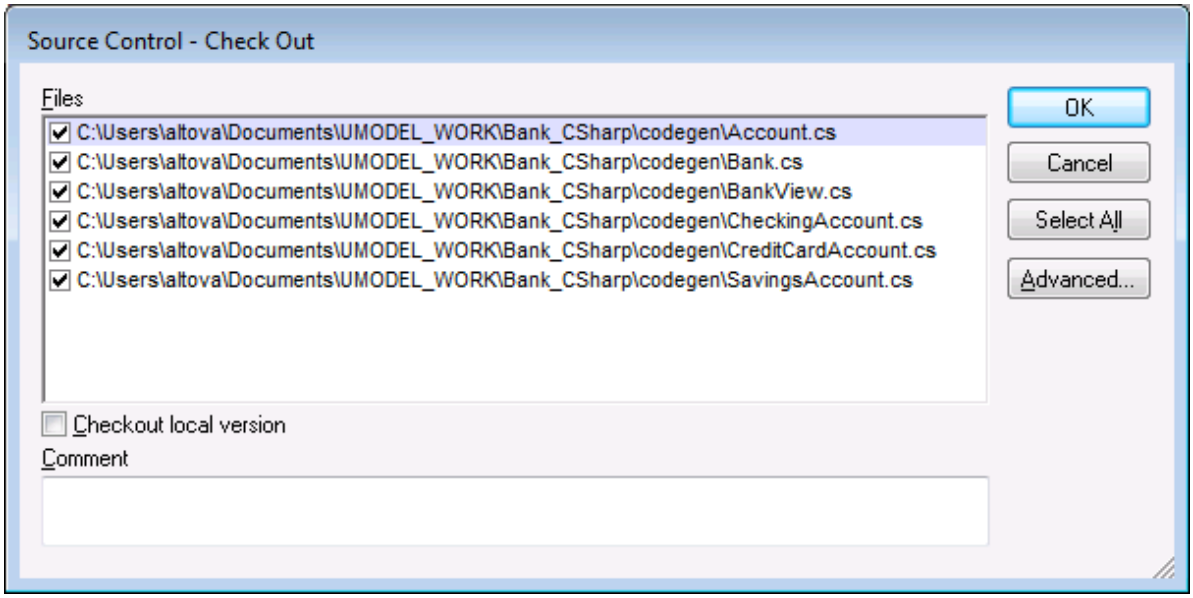
Mit dem Kontrollkästchen "Make writable" wird das "read-only" Attribut der abgerufenen Dateien entfernt.

10.3.6 Auschecken

Mit diesem Befehl checken Sie die aktuellste Version der ausgewählten Datei(en) aus und platzieren beschreibbare Kopien davon in das Arbeitsverzeichnis. Die Dateien werden für andere Benutzer als "ausgecheckt" markiert.

So checken Sie Dateien aus:

- Wählen Sie die gewünschte Datei bzw. den gewünschten Ordner in der Modell-Struktur aus.
- Wählen Sie **Projekt** | **Versionskontrolle** | **Auschecken**.



Anmerkung:
Sie können die Anzahl der auszucheckenden Dateien ändern, indem Sie die einzelnen Kontrollkästchen in der Liste der Dateien aktivieren.

Aktivieren Sie die Option **Lokale Version auschecken,** um nur die lokale Version der Dateien, nicht die aus der Versionskontrolldatenbank auszuchecken.

Folgende Objekte können ausgecheckt werden:

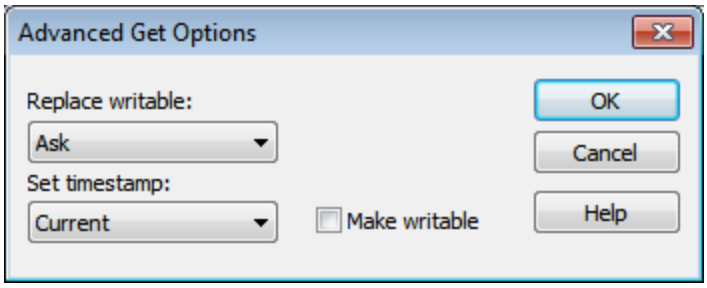
- Einzelne Dateien - klicken Sie in der Modell-Struktur auf die Datei (oder mehrere Dateien mittels Strg + Klick)
- Projektordner - klicken Sie in der Modellstruktur auf den Ordner (oder mehrere Ordner mittels Strg + Klick)

oder
Das rote Kontrollhäkchen zeigt an, dass die Datei/der Ordner ausgecheckt ist.#

 © 2016-2022 Altova GmbH

Erweitert

Mit Hilfe dieser Option können Sie die Optionen **Replace writable** und **Set timestamp** in den entsprechenden Auswahllisten auswählen.



Mit dem Kontrollkästchen "Make writable" wird das "read-only" Attribut der abgerufenen Dateien entfernt.

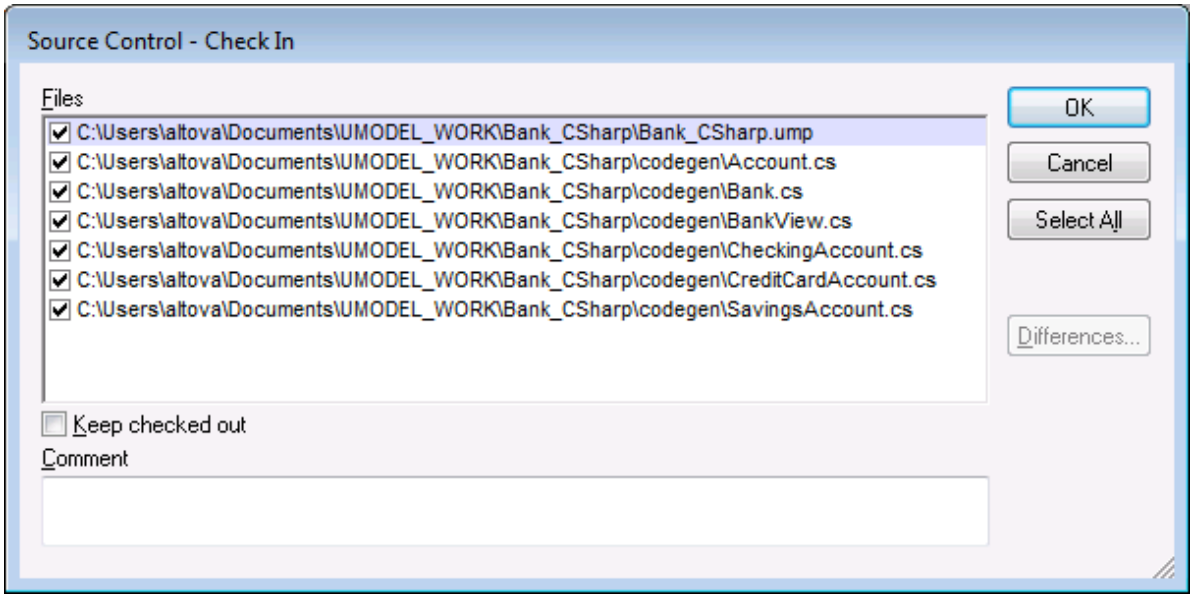
10.3.7 Einchecken

Mit diesem Befehl checken Sie zuvor ausgecheckte Dateien, d.h. Ihre lokal aktualisierten Dateien, wieder ein und stellen die Dateien zurück in das Versionskontrollprojekt.

So checken Sie Dateien ein:

- Wählen Sie die gewünschten Dateien in der Modell-Struktur aus.
- Wählen Sie **Projekt** | **Versionskontrolle** | **Einchecken**.

Kürzel: Rechtsklicken Sie im Projektfenster auf ein ausgechecktes Objekt, und wählen Sie "Einchecken" im Kontextmenü.



Anmerkung:

Sie können die Anzahl der einzucheckenden Dateien ändern, indem Sie die einzelnen Kontrollkästchen in der Liste der Dateien aktivieren.

Folgende Objekte können eingecheckt werden:

- Einzelne Dateien - klicken Sie in der Modell-Struktur auf die Datei (oder mehrere Dateien mittels Strg + Klick)
- Ordner - klicken Sie in der Modellstruktur auf die Ordner (mittels Strg + Klick)

oder

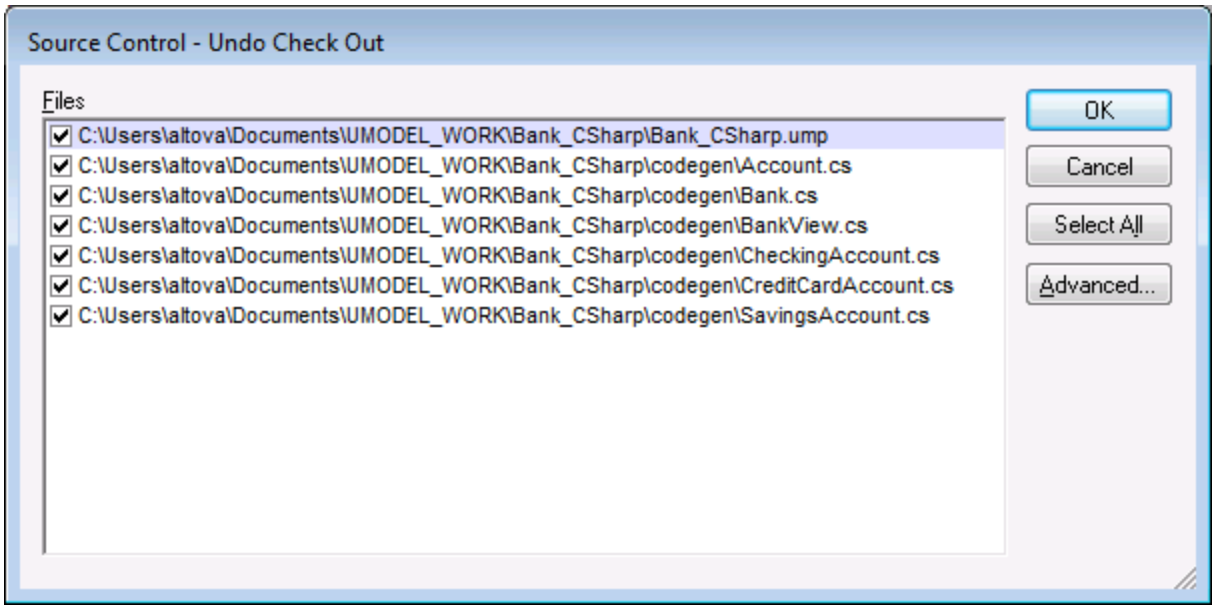
Das Vorhängeschloss-Symbol zeigt an, dass sich die Datei/der Ordner **unter Versionskontrolle** befindet, aber derzeit nicht ausgecheckt ist.

10.3.8 Auschecken rückgängig...

Mit diesem Befehl werden Änderungen in zuvor ausgecheckten Dateien, also lokal aktualisierten Dateien **verworfen**. Es werden die alten Dateien aus der Versionskontrolldatenbank beibehalten.

So machen Sie das Auschecken rückgängig:

- Wählen Sie die Dateien in der Modellstruktur aus
- Wählen Sie **Projekt | Versionskontrolle | Auschecken rückgängig**.



Anmerkung:
Sie können die Anzahl der Dateien ändern, indem Sie die einzelnen Kontrollkästchen in der Liste der Dateien aktivieren.

Die Option "Auschecken rückgängig" kann auf die folgenden Dateien angewendet werden:

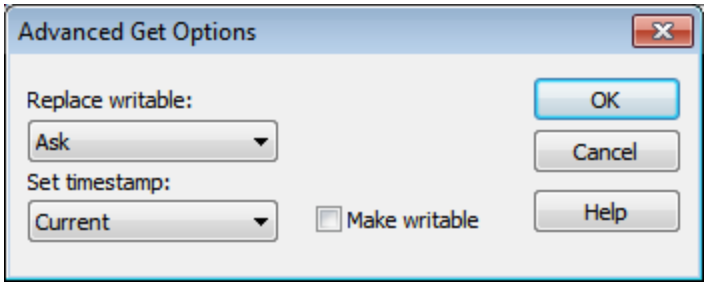
- Einzelne Dateien - klicken Sie in der Modell-Struktur auf die Datei (oder mehrere Dateien mittels Strg + Klick)

 © 2016-2022 Altova GmbH

- Ordner - klicken Sie in der Modellstruktur auf die Ordner (mittels Strg + Klick)

Erweitert

Mit Hilfe dieser Option können Sie die Optionen **Replace writable** und **Set timestamp** in den entsprechenden Auswahllisten auswählen.



Mit dem Kontrollkästchen "Make writable" wird das "read-only" Attribut der abgerufenen Dateien entfernt.

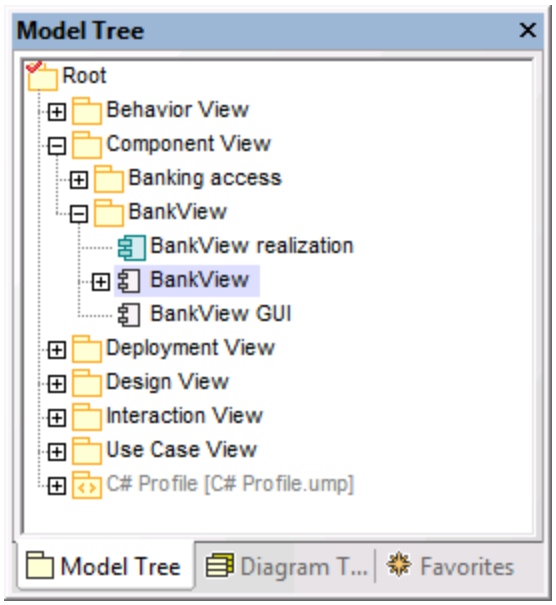
10.3.9 Zu Versionskontrolle hinzufügen

Fügt die ausgewählten Dateien oder Ordner zur Versionskontrolldatenbank hinzu und stellt sie unter Versionskontrolle. Wenn Sie ein neues UModel-Projekt hinzufügen, werden Sie aufgefordert den Arbeitsbereichsordner und den Pfad anzugeben, unter dem das Projekt gespeichert werden soll.

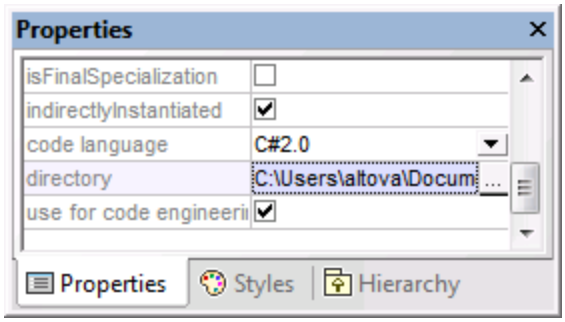
Nachdem Sie die UModel-Projektdatei (*.**ump**) unter Versionskontrolle gestellt haben, können Sie anschließend die beim Code Engineering erzeugten **Codedateien** ebenfalls zur Versionskontrolle hinzufügen. Bitte beachten Sie, dass die generierten Codedateien und das UModel-Projekt in das oder unterhalb desselben SourceSafe-Arbeitsverzeichnisses gelegt werden müssen. Das in diesem Abschnitt verwendete Arbeitsverzeichnis ist **c:\temp\ssc\Bank_CSharp**.

So fügen Sie mit UModel generierte Codedateien zur Versionskontrolle hinzu:

1. Erweitern Sie in der Modell-Struktur den Ordner "Component View" und navigieren Sie zur Komponente "BankView".

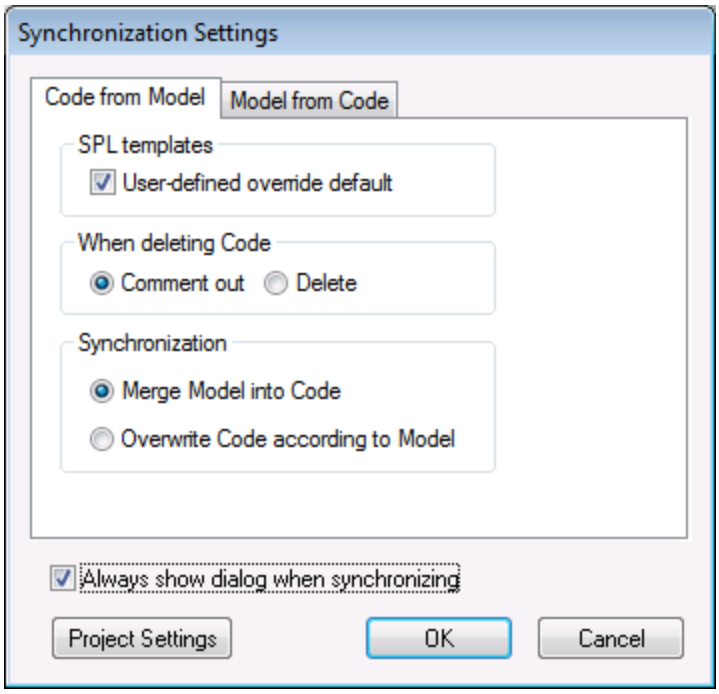


2. Klicken Sie auf die BankView-Komponente und klicken Sie im Fenster "Eigenschaften" auf das Durchsuchen-Symbol neben dem Feld "Verzeichnis".

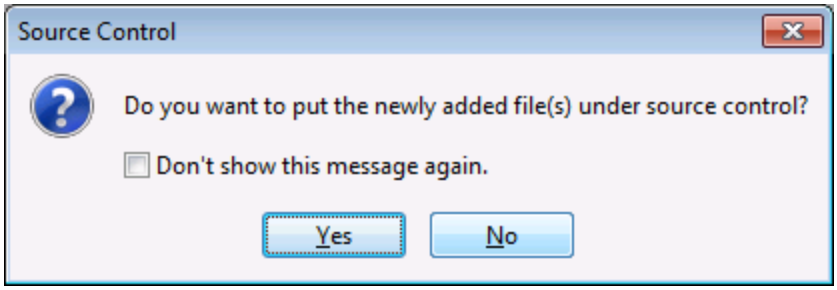


3. Ändern Sie das Code Engineering-Verzeichnis z.B. in **C:\Users\Altova\Documents\UMODEL_WORK\codegen**.
4. Wählen Sie den Menübefehl **Projekt | Merge Programmcode aus UModel-Projekt**.
5. Ändern Sie gegebenenfalls die Synchronisierungseinstellungen und bestätigen Sie mit OK.

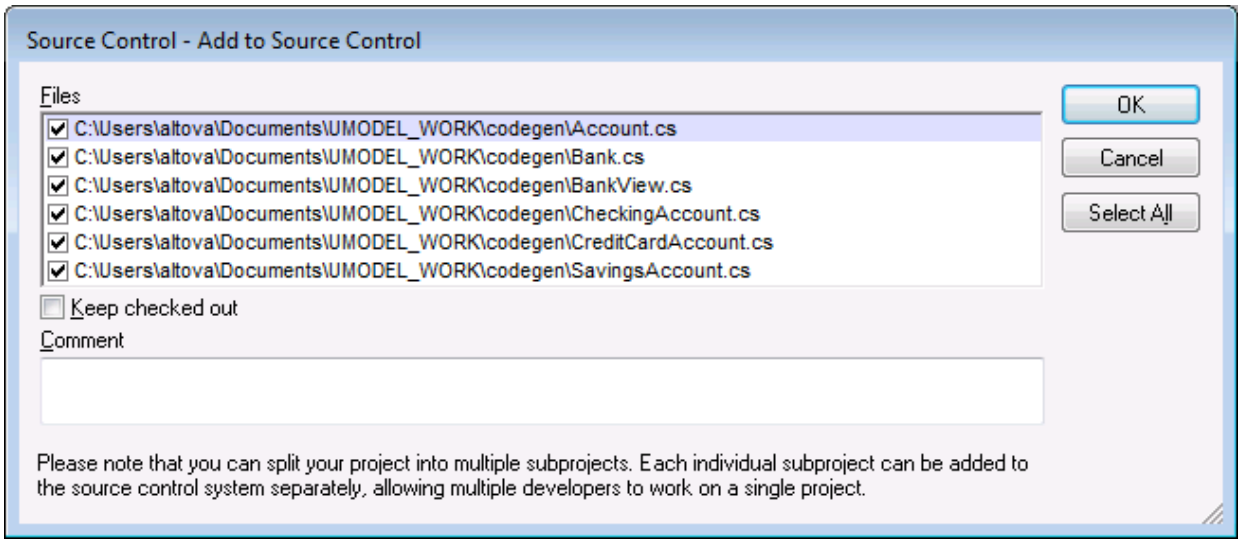
© 2016-2022 Altova GmbH



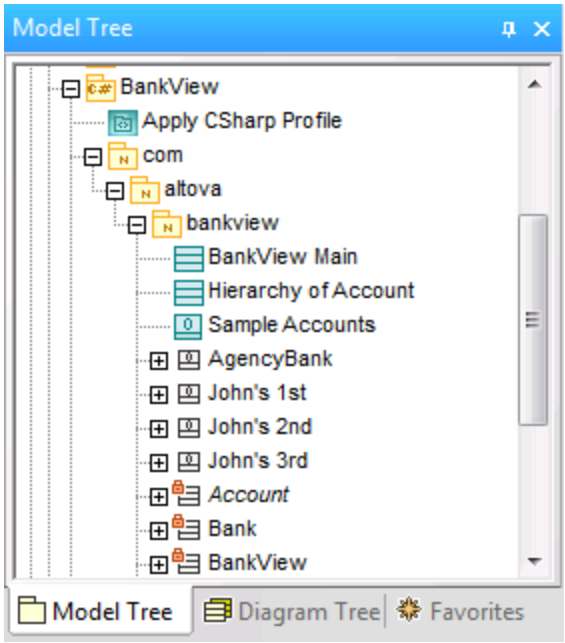
Im Meldungsfenster wird der Code aus dem Projektvorgang angezeigt.
Es erscheint ein Meldungsfeld, in dem Sie gefragt werden, ob Sie die neu erstellten Dateien unter Versionskontrolle stellen möchten.



6. Klicken Sie auf Ja.
7. Daraufhin wird das Dialogfeld "Zu Versionskontrolle hinzufügen" geöffnet, in dem Sie die Dateien auswählen können, die unter Versionskontrolle gestellt werden sollen.



8. Klicken Sie auf OK, sobald Sie die gewünschten Dateien ausgewählt haben.
   Neben den einzelnen unter Versionskontrolle gestellten Klassen/Dateiquellen erscheint nun ein Vorhängeschloss-Symbol.



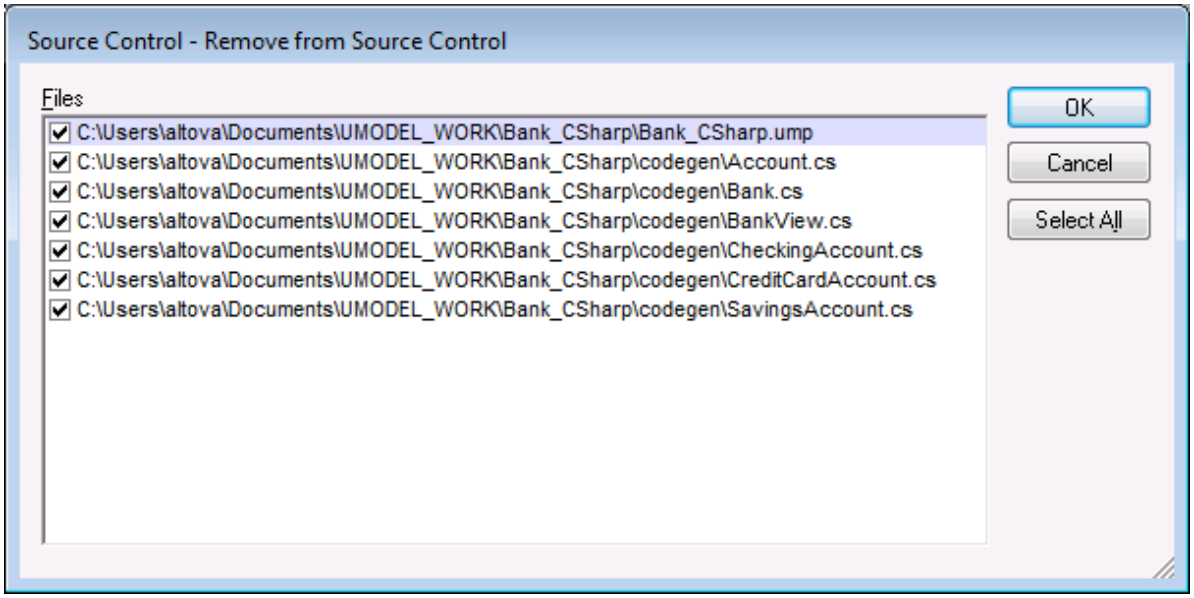
10.3.10 Von Versionskontrolle ausgliedern

Mit diesem Befehl **entfernen** Sie zuvor hinzugefügte Dateien aus dem Versionskontrollprodukt. Diese Dateiarten werden zwar in der Modell-Struktur weiterhin angezeigt, können aber nicht ausgecheckt werden. Mit Hilfe des Befehls "Zu Versionskontrolle hinzufügen" können Sie die Dateien wieder unter Versionskontrolle stellen.

 © 2016-2022 Altova GmbH

So entfernen Sie Dateien aus der Versionskontrolle:

- Wählen Sie die gewünschten Dateien in der Modellstruktur aus
- Wählen Sie **Projekt** | **Versionskontrolle** | **Von Versionskontrolle ausgliedern**.



Anmerkung:
Sie können die Anzahl der auszugliedernden Dateien ändern, indem Sie die einzelnen Kontrollkästchen in der Liste der Dateien aktivieren.

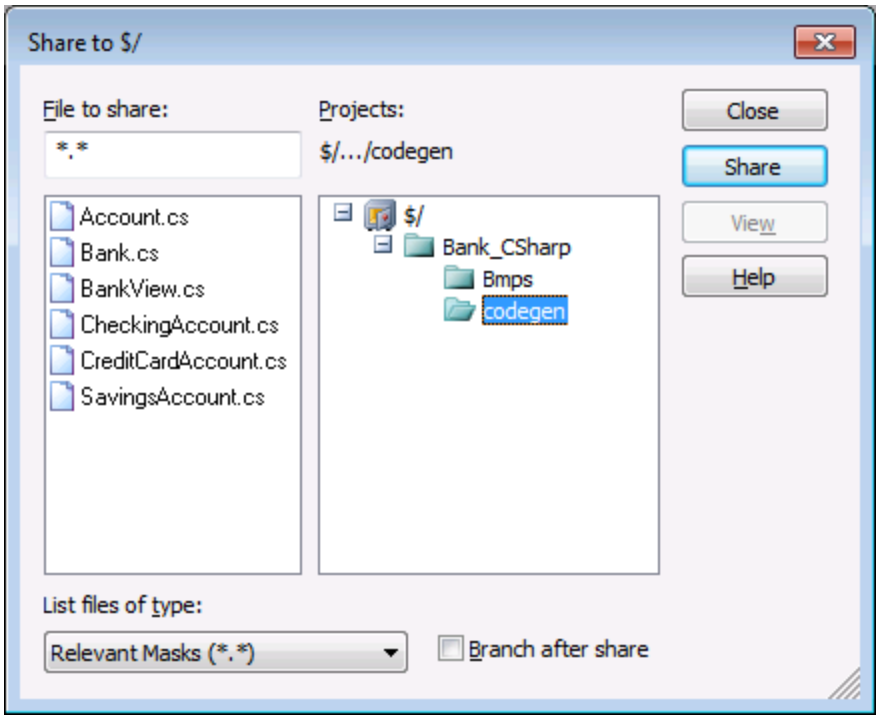
Folgende Objekte könne aus der Versionskontrolle ausgegliedert werden:
- Einzelne Dateien - klicken Sie auf die Datei (oder mehrere Dateien mittels Strg + Klick)
- Ordner - klicken Sie auf das Ordnersymbol

10.3.11 Aus Versionskontrolle freigeben

Mit diesem Befehl werden Dateien aus anderen Projekten/Ordnern im ausgewählten Ordner in die Versionskontrollspeicherschnittstelle in Form von freigegebenen Dateien einem Branch eingefügt.
Um dem Freigeben-Befehl verwenden zu können, müssen Sie für das Projekt, das Sie freigeben möchten, Ein- und Auscheck-Rechte besitzen.

So geben Sie eine Datei aus der Versionskontrolle frei:

1. Wählen Sie den gewünschten Ordner in der Modellstruktur aus und wählen Sie den Befehl **Projekt | Versionskontrolle | Aus Versionskontrolle freigeben,** z.B. die Komponente BankView im Ordner Component View.
2. Wählen Sie im Listenfeld "Projekte" den Projektordner aus, der die Datei enthält.



3. Wählen Sie im Listenfeld "Files to share" die gewünschte Datei aus und klicken Sie auf die Schaltfläche "Freigeben".
   Die Datei wird nun aus der Liste "Files to share" entfernt.
4. Klicken Sie zum Fortfahren auf "Schließen".

Branch after share
Gibt die Datei frei und erstellt einen neuen Branch zum Erstellen einer separaten Version.

10.3.12 Verlauf anzeigen

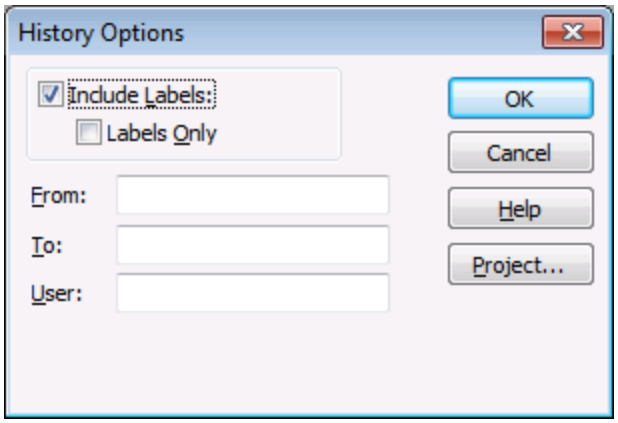
Mit diesem Befehl zeigen Sie den Verlauf einer unter Versionskontrolle stehenden Datei an. Sie können damit eine detaillierte Verlaufsliste anzeigen, genauere Verlaufsinformationen anzeigen, einen Vergleich durchführen oder frühere Versionen der Datei abrufen.

So zeigen Sie den Verlauf einer Datei an:

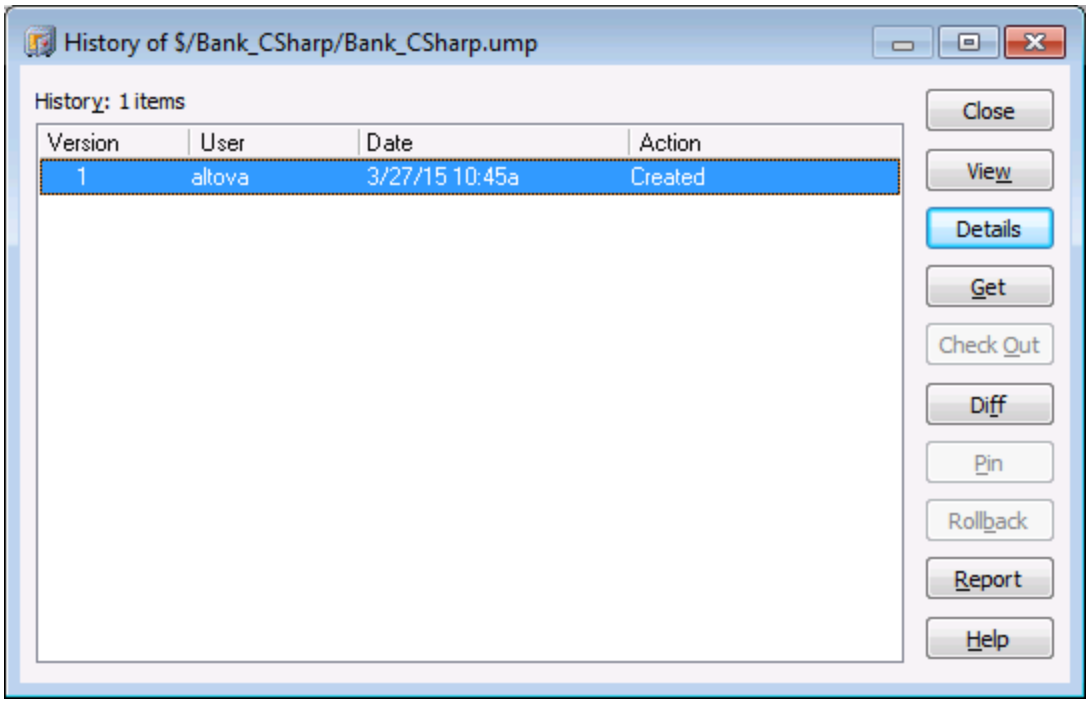
1. Klicken Sie im Fenster "Modell-Struktur" auf die Datei, deren Verlauf Sie ansehen möchten.
2. Wählen Sie die Menüoption **Projekt | Versionskontrolle | Verlauf anzeigen**.

   Es erscheint ein Dialogfeld, in dem Sie nähere Informationen eingeben müssen.

 © 2016-2022 Altova GmbH



3. Wählen Sie den gewünschten Eintrag und bestätigen Sie mit OK.



Dieses Dialogfeld bietet verschiedene Möglichkeiten, um bestimmte Versionen der ausgewählten Datei zu vergleichen und abzurufen. Wenn Sie auf einen Eintrag in der Liste doppelklicken, wird das Dialogfeld "Verlauf" für diese Datei geöffnet.

Schließen
Schließt dieses Dialogfeld.

Ansicht
Öffnet ein weiteres Dialogfeld, in dem Sie die Art des Ansichtsprogramms auswählen können, in dem Sie die Datei anzeigen möchten.

Details
Öffnet ein Dialogfeld, in dem Sie die Eigenschaften[479] der gerade aktiven Datei sehen.

Abrufen

Damit können Sie eine der vorherigen Versionen der Datei aus der Versionsliste in das Arbeitsverzeichnis holen.

Auschecken
Damit können Sie die **neueste** Version der Datei auschecken.

Diff
Öffnet das Dialogfeld Vergleichsoptionen 478, in dem Sie die Vergleichsoptionen zur Anzeige der Unterschiede zwischen den beiden Dateiversionen definieren können.

Mit Hilfe von Strg + Klick können Sie zwei Dateiversionen in diesem Fenster markieren. Klicken Sie anschließend auf Diff, um die Unterschiede zwischen den beiden Dateien anzuzeigen.

Pin
Markiert eine Version der Datei bzw. hebt die Markierung auf. Damit können Sie die Dateiversion definieren, die für den Dateivergleich verwendet werden soll.

Rollback
Führt ein Rollback für die ausgewählte Version der Datei durch.

Report
Generiert einen Verlaufsbericht, den Sie an den Drucken, die Datei oder die Zwischenablage senden können.

Hilfe
Öffnet die Online-Hilfe des Versionskontrollanbieters für das Plugin.

10.3.13 Unterschiede anzeigen

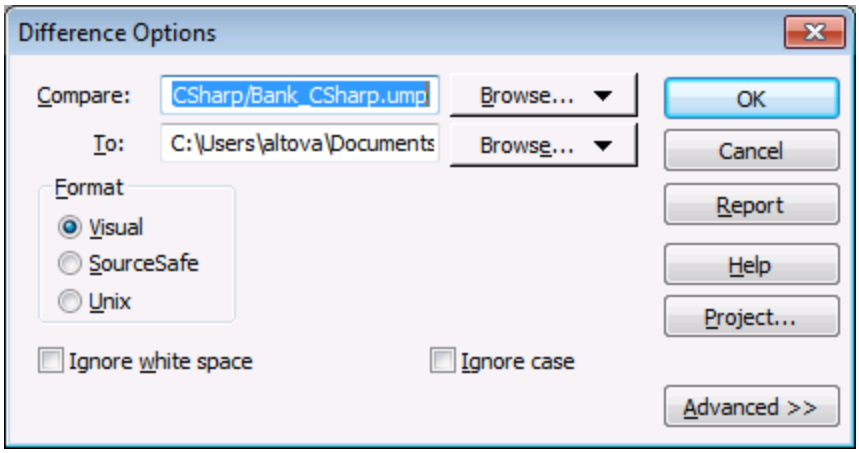
Dieser Befehl ermöglicht die Anzeige von Unterschieden zwischen der in der Versionskontroll-Ablage befindlichen und der gleichnamigen Datei, die Sie **ein-/ausgecheckt** haben.

Wenn Sie eine der Dateien im Dialogfeld "Verlauf" mit einem Pin markiert haben, so wird diese Datei im Textfeld "Vergleichen" verwenden. Sie können mit Hilfe der Durchsuchen-Schaltfläche 2 beliebige Dateien auswählen.

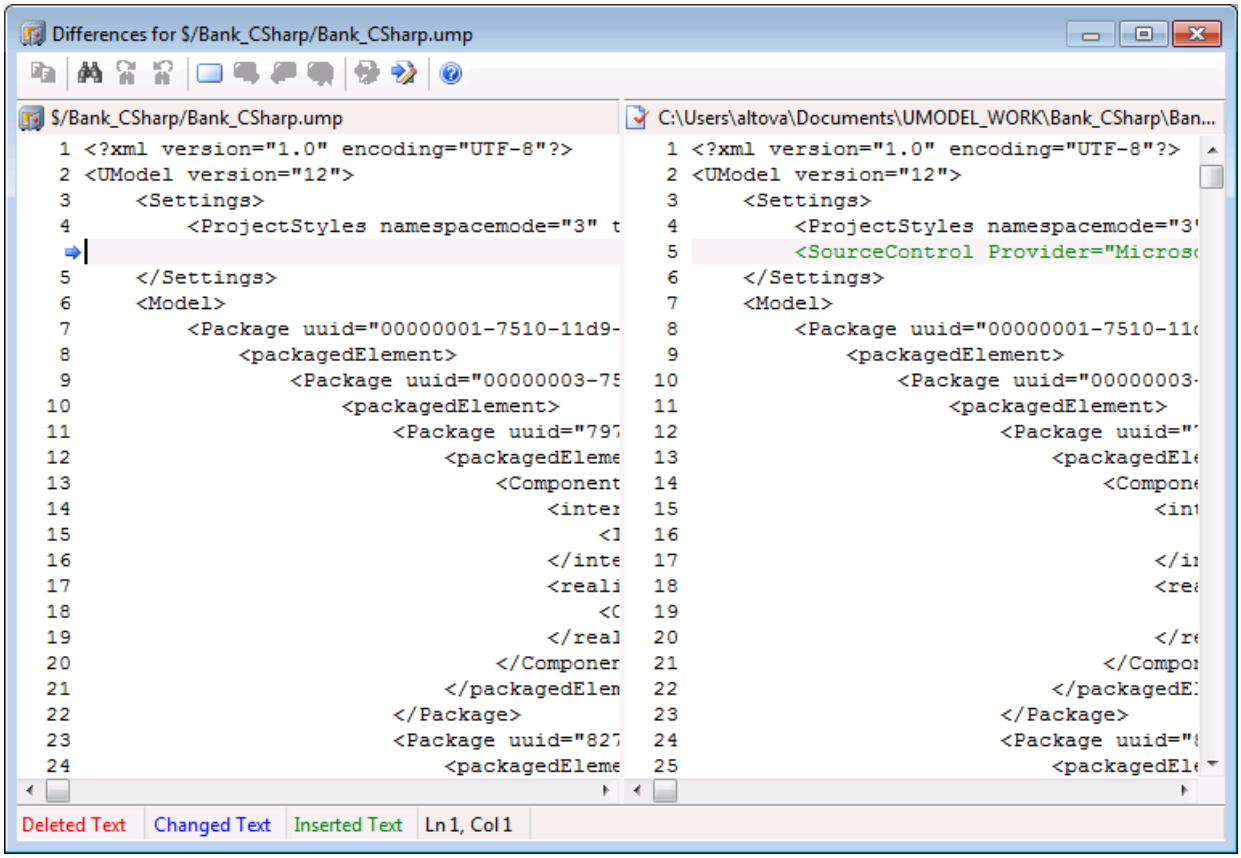
So zeigen Sie die Unterschiede zwischen zwei Dateien an:

1. Klicken Sie in der Modell-Struktur auf eine Datei.
2. Wählen Sie die Menüoption **Projekt | Versionskontrolle | Unterschiede anzeigen.**
   Es erscheint ein Dialogfeld, in dem Sie weitere Informationen eingeben können.

 © 2016-2022 Altova GmbH



3. Wählen Sie die gewünschten Einträge und bestätigen Sie mit OK.



Die Unterschiede zwischen beiden Dateien werden in beiden Fenstern farblich markiert (in diesem Beispiel wird MS Source-Safe verwendet).

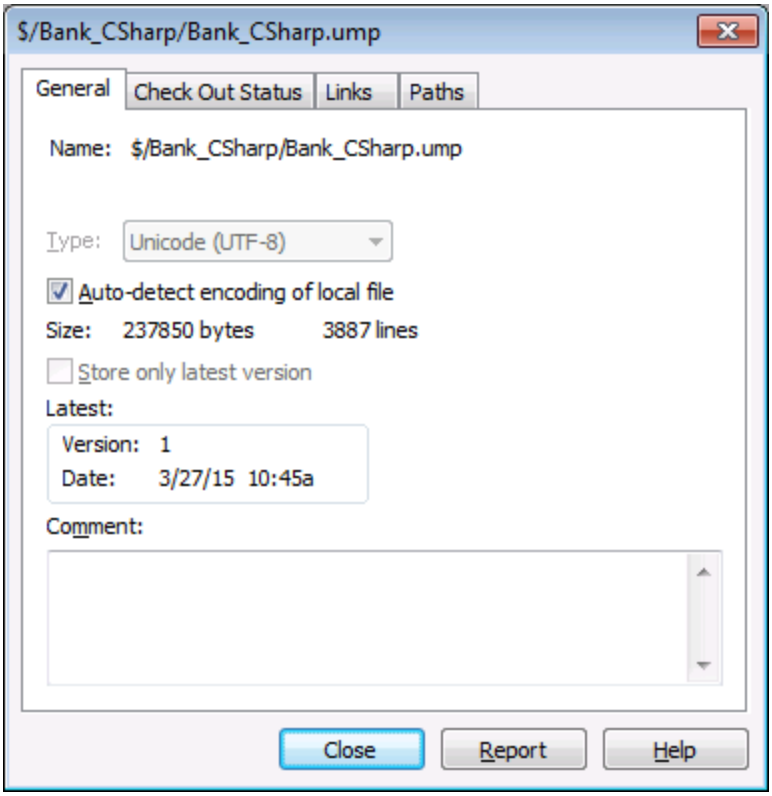
10.3.14 Eigenschaften anzeigen

Mit diesem Befehl können Sie die Eigenschaften der ausgewählten Datei anzeigen. Die Anzeige kann je nach verwendetem Versionskontrollprodukt unterschiedlich sein.

So zeigen Sie die Eigenschaften der aktuell ausgewählten Datei an:

- Wählen Sie **Projekt** | **Versionskontrolle** | **Eigenschaften**.

Dieser Befehl kann jeweils nur an einer einzelnen Datei ausgeführt werden.



10.3.15 Status aktualisieren

Mit diesem Befehl **aktualisieren** Sie den Status aller Projektdateien unabhängig von ihrem derzeitigen Status.

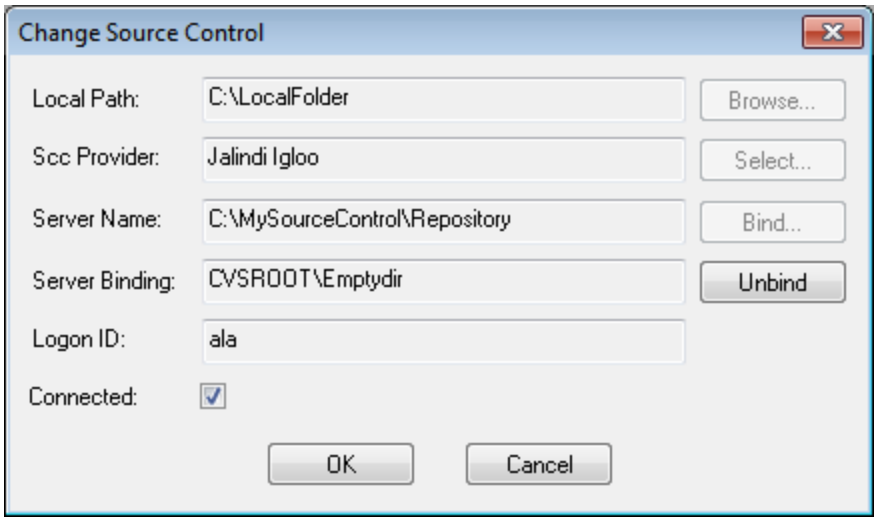
10.3.16 Versionskontrollmanager

Mit diesem Befehl **starten** Sie die ursprüngliche Oberfläche Ihrer Versionskontroll-Software.

10.3.17 Versionskontrolle wechseln

Über dieses Dialogfeld können Sie das Versionskontrollsystem-Binding wechseln. Klicken Sie dazu zuerst auf die Schaltfläche "Bindung aufheben" und anschließend (optional) auf die Schaltfläche "Auswählen", um ein neues Versionskontrollsystem auszuwählen. Klicken Sie anschließend auf die Schaltfläche "Binden", um eine Bindung zu einem neuen Pfad in der Speicherschnittstelle zu erstellen.

 © 2016-2022 Altova GmbH

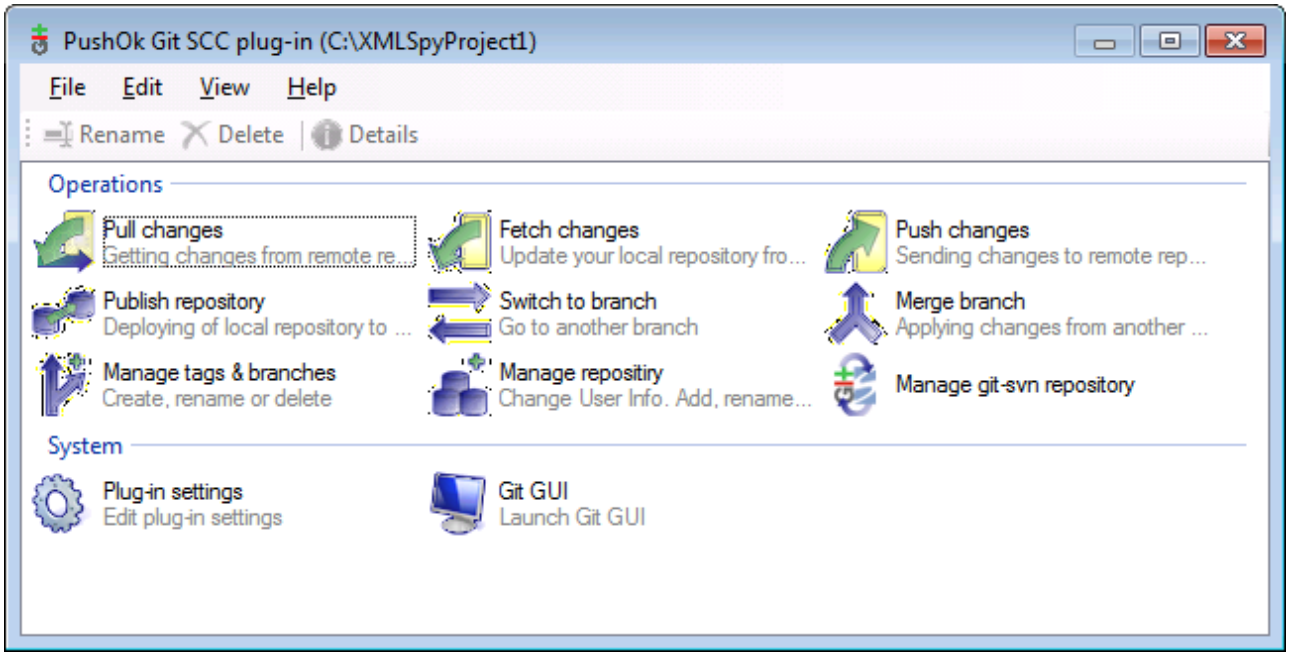

Change Source Control
Local Path:
C:\LocalFolder
Browse...
Scc Provider:
Jalindi Igloo
Select...
Server Name:
C:\MySourceControl\Repository
Bind...
Server Binding:
CVSROOT\Emptydir
Unbind
Logon ID:
ala
Connected:
OK
Cancel

10.4 Versionskontrolle mit Git

Unterstützung für Git als Versionskontrollsystem in UModel steht in Form eines Drittanbieter-Plug-in namens **GIT SCC Plug-in** (http://www.pushok.com/software/git.html) zur Verfügung.

> Zum Zeitpunkt der Verfassung dieser Dokumentation steht das **GIT SCC Plug-in** zum Experimentieren zur Verfügung. Um das Plug-in verwenden zu können, müssen Sie beim Plug-in-Anbieter registriert sein.

Mit Hilfe des GIT SCC Plug-in können Sie über die Befehle im Menü **Projekt | Versionskontrolle** von UModel mit einem Git Repository arbeiten. Beachten Sie, dass die Befehle im Menü **Projekt | Versionskontrolle** von UModel von der Microsoft Source Control Plug-in API (MSSCCI API), für die eine andere Art von Design als von Git verwendet wird, bereitgestellt werden. Daher bildet das Plug-in eine Zwischenschaltung zwischen "Visual Source Safe"-Funktionalitäten und Git-Funktionalitäten. Das bedeutet einerseits, dass ein Befehl wie z.B. **Aktuellste Version holen** für Git eventuell so nicht verwendet werden kann. Andererseits gibt es einige neue Git-spezifische Aktionen, die über das Plug-in im Dialogfeld "Versionskontrollmanager" zur Verfügung gestellt werden (Menü **Projekt | Versionskontrolle | Versionskontrollmanager** von UModel).



Das Dialogfeld "Versionskontrollmanager"

Andere häufig benötigte Versionskontrollbefehle stehen direkt im Menü **Projekt | Versionskontrolle** zur Verfügung.

In den folgenden Abschnitten wird die Anfangskonfiguration des Plug-in sowie der grundlegende Arbeitsablauf beschrieben:

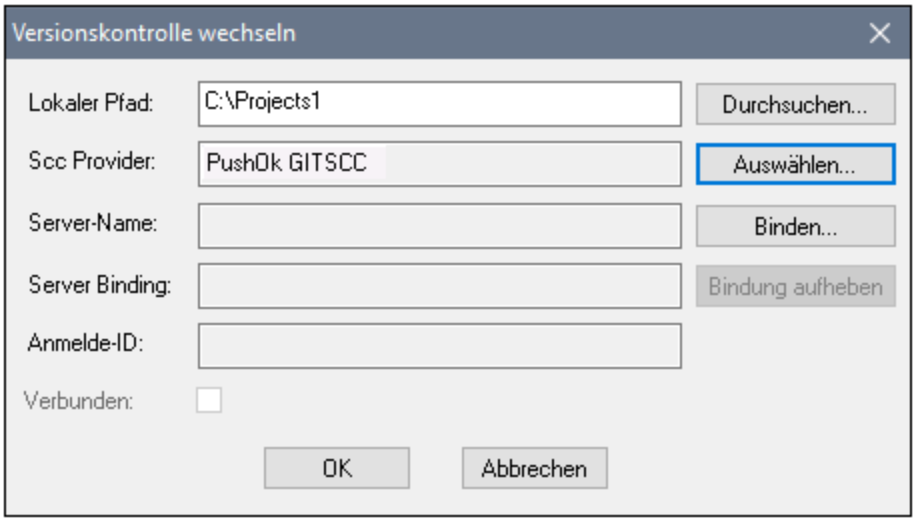
- Aktivieren der Git-Versionskontrolle mit dem Git SCC Plug-in 483
- Hinzufügen eines Projekts zur Git-Versionskontrolle 483
- Klonen eines Projekts anhand der Git-Versionskontrolle 485

 © 2016-2022 Altova GmbH

10.4.1 Aktivieren der Git-Versionskontrolle mit dem Git SCC Plug-in

Um die Git-Versionskontrolle mit UModel zu aktivieren, muss zuerst das Drittanbieter-**PushOK GIT SCC Plug-in** installiert, registriert und als Versionskontrollanbieter ausgewählt werden. Dies geschieht folgendermaßen:

1. Laden Sie die Installationsdatei für das Plug-in von der Website des Anbieters (http://www.pushok.com) herunter, starten Sie sie und befolgen Sie die Installationsanweisungen.
2. Klicken Sie im Menü **Projekt** von UModel auf **Versionskontrolle wechseln** und vergewissern Sie sich, dass **PushOk GITSCC** als Versionskontroll-Provider ausgewählt ist. Wenn **Push Ok GITSCC** in der Liste der Provider nicht angezeigt wird, war die Installation des Plug-in wahrscheinlich nicht erfolgreich. Lesen Sie in diesem Fall in der Dokumentation des Anbieters nach, wie Sie das Problem lösen können.



3. Wenn ein Dialogfeld angezeigt wird, in dem Sie aufgefordert werden, das Plug-in zu registrieren, klicken Sie auf **Registrierung** und befolgen Sie die Anweisungen des Assistenten, um die Registrierung abzuschließen.

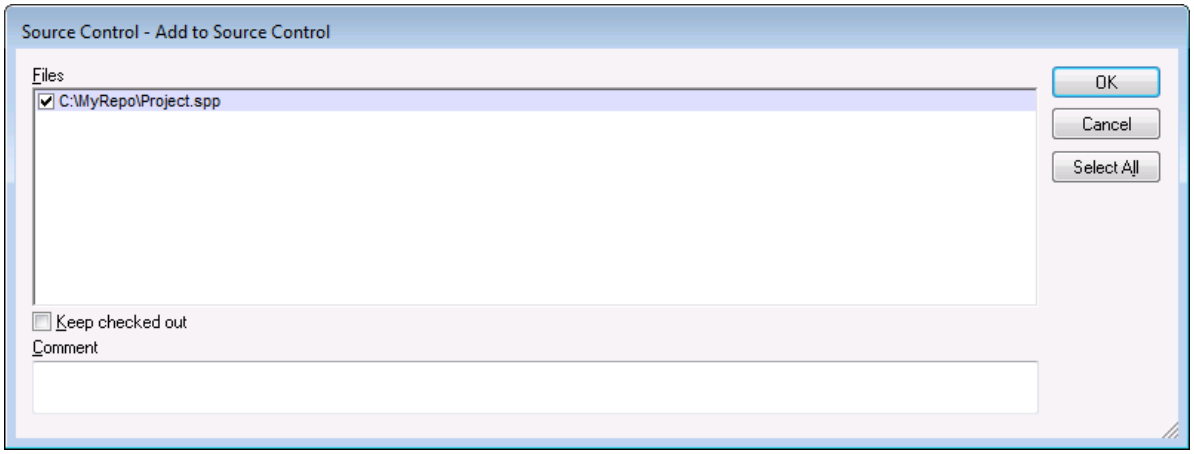
10.4.2 Hinzufügen eines Projekts zur Git-Versionskontrolle

Sie können UModel -Projekte als Git Repositories speichern. Die Struktur der zum Projekt hinzugefügten Dateien oder Ordner würde anschließend der Struktur des Git Repository entsprechen.

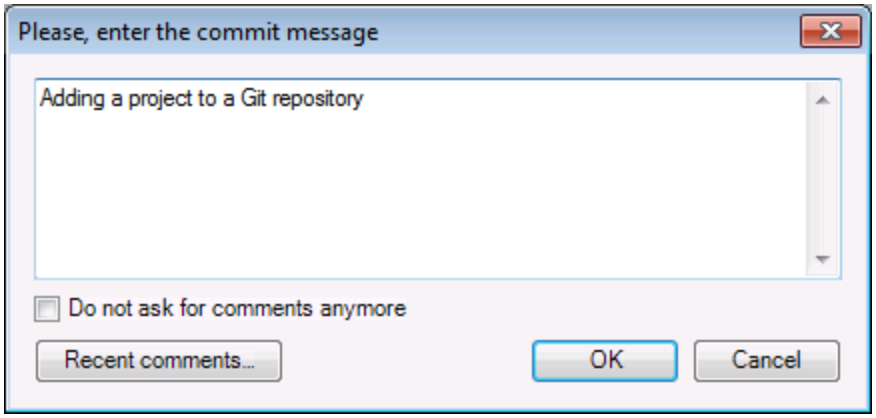
So fügen Sie ein Projekt zu einer Git-Versionskontrolle hinzu:

1. Stellen Sie sicher, dass das **PushOK GIT SCC Plug-in** als Versionskontrollanbieter ausgewählt ist (siehe Aktivieren der Git-Versionskontrolle mit dem Git SCC Plug-in 483).
2. Erstellen Sie ein neues leeres Projekt und vergewissern Sie sich, dass es keine Validierungsfehler aufweist (d.h. dass bei Auswahl des Befehls **Projekt | Projektsyntax überprüfen** keine Fehler oder Warnungen angezeigt werden).
3. Speichern Sie das Projekt in einem lokalen Ordner, z.B. unter `C:\MyRepo\Project.ump`.
4. Klicken Sie im Fenster der **Modell-Struktur** auf den **Root**-Node.

5. Klicken Sie im Menü **Projekt** unter **Versionskontrolle** auf **Zu Versionskontrolle hinzufügen**.

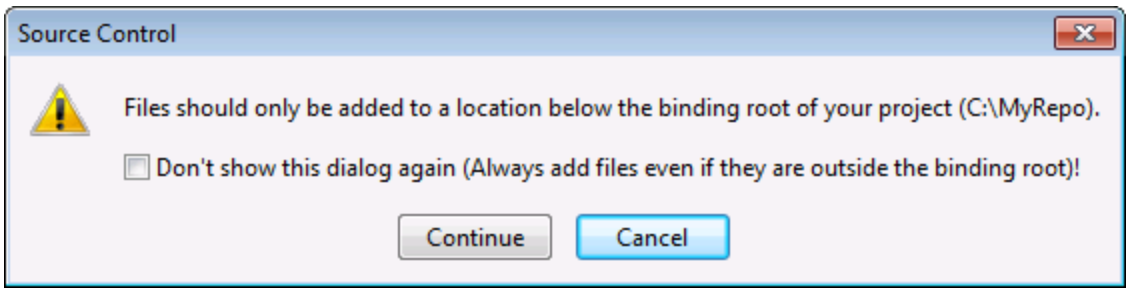


6. Klicken Sie auf **OK**.



7. Geben Sie den Text Ihrer Commit-Meldung ein und klicken Sie auf **OK**.

Sie können jetzt Modellierungselemente (Diagramme, Klassen, Pakete usw.) zu Ihrem Projekt hinzufügen. Beachten Sie, dass alle Projektdateien und -ordner sich unter dem Root-Ordner des Projekts befinden müssen. Wenn das Projekt z.B. im Ordner `C:\MyRepo` angelegt wurde, so sollten nur Dateien unter `C:\MyRepo` zur Projekt hinzugefügt werden. Wenn Sie versuchen, Dateien, die sich außerhalb des Projekt-Root-Ordners befinden, zum Projekt hinzuzufügen, wird eine Warnmeldung angezeigt:

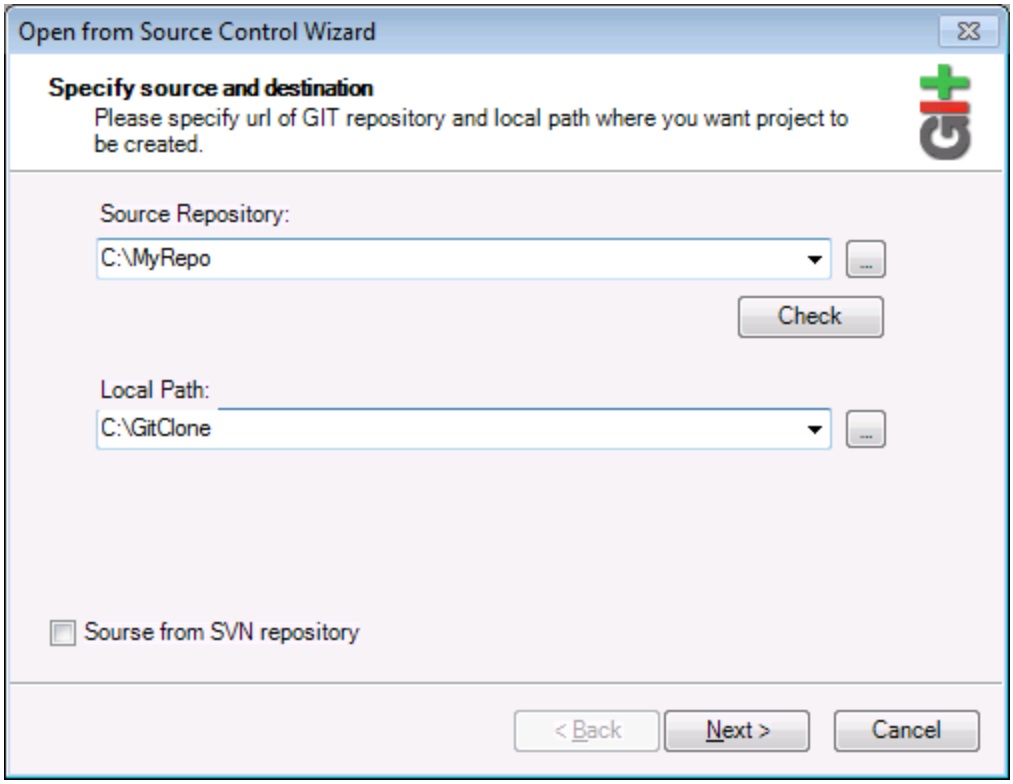


 © 2016-2022 Altova GmbH

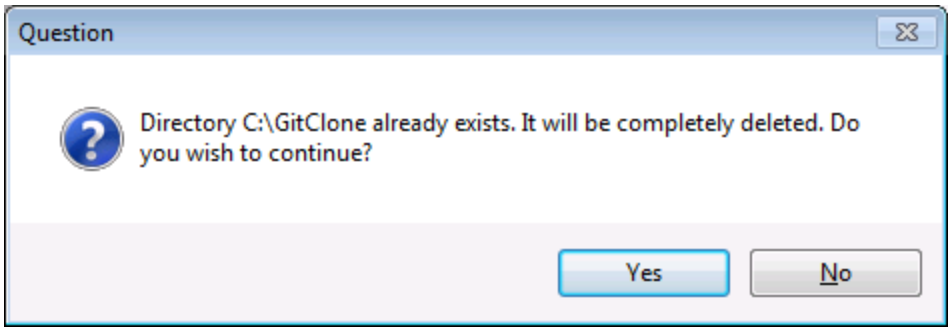
10.4.3 Klonen eines Projekts anhand der Git-Versionskontrolle

Projekte, die bereits zur Git-Versionskontrolle hinzugefügt wurden (siehe Hinzufügen eines Projekts zur Git-Versionskontrolle 483) können folgendermaßen über das Git Repository geöffnet werden:

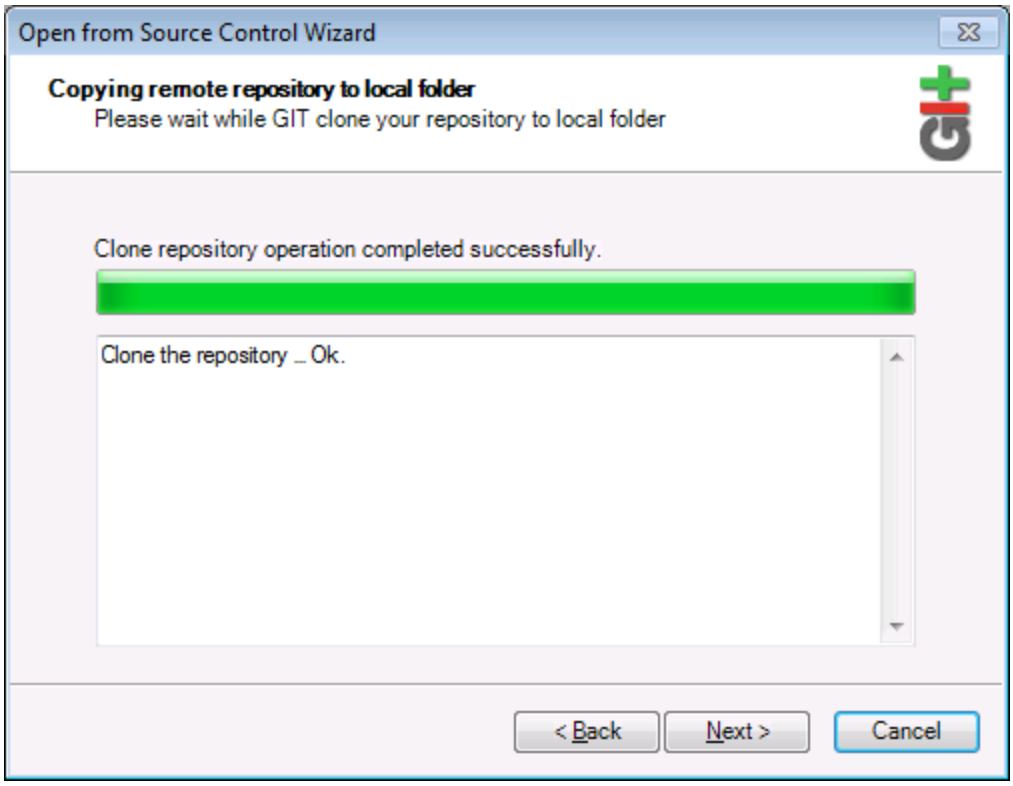
1. Vergewissern Sie sich, dass das **PushOK GIT SCC Plug-in** als Versionskontroll-Provider ausgewählt ist (siehe Aktivieren der Git-Versionskontrolle mit dem Git SCC Plug-in 483).
2. Klicken Sie im Menü **Projekt** auf **Versionskontrolle | Aus Versionskontrolle öffnen**.
3. Geben Sie den Pfad oder die URL des Versionskontroll-Repository ein. Klicken Sie auf **Überprüfen,** um die Gültigkeit des Pfads oder der URL zu überprüfen.



4. Geben Sie unter **Lokaler Pfad** den Pfad zu dem lokalen Ordner ein, in dem das Projekt erstellt werden soll und klicken Sie auf **Weiter**. Wenn der lokale Ordner vorhanden ist, wird (auch wenn er leer ist) das folgende Dialogfeld aufgerufen:



5. Klicken Sie zur Bestätigung auf **Ja** und anschließend auf **Weiter**.



6. Stellen Sie die restlichen Schritte des Assistenten fertig, wie für Ihr Projekt erforderlich.
7. Nach der Fertigstellung wird ein Durchsuchen-Dialogfeld angezeigt, in dem Sie aufgefordert werden, das UModel-Projekt (*.ump)-Datei zu öffnen. Wählen Sie die Projektdatei aus, um den Projektinhalt in UModel zu laden.

© 2016-2022 Altova GmbH

11 UModel Diagrammsymbole

Der folgende Abschnitt enthält eine Kurzübersicht über die Symbole, die in den einzelnen Modelldiagrammen zur Verfügung stehen.

Die Symbole sind in zwei Gruppen unterteilt:

- **Hinzufügen** - Zeigt eine Liste von Elementen an, die zum Diagramm hinzugefügt werden können.
- **Beziehung** - Zeigt eine Liste von Beziehungsarten an, die zwischen Elementen im Diagramm erstellt werden können.

11.1 Aktivitätsdiagramm

Hinzufügen

Aktion (Aufrufverhalten-Aktion)
Aktion (Aufrufoperationsaktion)
Ereignisannahmeaktion
Ereignisannahmeaktion (Zeitereignis)
Signalsendeaktion

Verzweigungsknoten (Verzweigung)
Verbindungsknoten
Startknoten
Aktivitätsendknoten
Endknoten für Kontrollflüsse
Parallelisierungsknoten (vertikal)
Parallelisierungsknoten (horizontal)
Synchronisationsknoten
Synchronisationsknoten (horizontal)

Inputpin
Outputpin
Wert-Pin

Objektknoten
Pufferknoten
Datenspeicherknoten
Aktivitätsbereich (horizontal)
Aktivitätsbereich (vertikal)
Aktivitätsbereich 2-dimensional

Kontrollfluss
Objektfluss
Ausnahme-Handler

Aktivität
Aktivitätsparameterknoten
StrukturierterAktivitätsknoten
Mengenverarbeitungsbereich
Erweiterungsknoten
Unterbrechungsbereich

Anmerkung
Anmerkung verknüpfen

© 2016-2022 Altova GmbH

11.2 Klassendiagramm

Beziehung

Assoziation
Aggregation
Komposition
Assoziationsklasse
Abhängigkeit
Verwendung
Schnittstellenrealisierung
Generalisierung

Hinzufügen

Paket
Klasse
Schnittstelle
Enumeration
Datentyp
Primitivtyp
Profil
Stereotyp
Profilzuweisung
Instanzspezifikation

Anmerkung
Anmerkung verknüpfen

11.3 Kommunikationsdiagramm

Hinzufügen

Lebenslinie
Nachricht (Aufruf)
Nachricht (Antwort)
Nachricht (Erstellung)
Nachricht (Löschung)

Anmerkung
Anmerkung verknüpfen

© 2016-2022 Altova GmbH

11.4 Kompositionsstrukturdiagramm

Hinzufügen

Kollaboration
Kollaborationsanwendung
Part (Eigenschaft)
Klasse
Schnittstelle
Port

Beziehung

Konnektor
Abhängigkeit (Rollenbindung)
Schnittstellenrealisierung
Verwendung

Anmerkung
Anmerkung verknüpfen

11.5 Komponentendiagramm

Hinzufügen

Paket
Schnittstelle
Klasse
Komponente
Artefakt

Beziehung

Realisierung
Schnittstellenrealisierung
Verwendung
Abhängigkeit

Anmerkung
Anmerkung verknüpfen

© 2016-2022 Altova GmbH

11.6 Deployment-Diagramm

Hinzufügen

Paket
Komponente
Artefakt
Knoten
Gerät
Ausführungsumgebung

Beziehung

Manifestation
Deployment
Assoziation
Generalisierung
Abhängigkeit

Anmerkung
Anmerkung verknüpfen

11.7 Interaktionsübersichtsdiagramm

Hinzufügen

Aufrufverhalten - Aktion (Interaktion)
Aufrufverhalten - Aktion (Interaktionsverwendung)
Verzweigungsknoten
Verbindungsknoten
Startknoten
Aktivitätsendknoten
Parallelisierungsknoten
Parallelisierungsknoten (Horizontal)
Synchronisationsknoten
Synchronisationsknoten (Horizontal)
Zeitdauerbedingung

Beziehung

Kontrollfluss

Anmerkung
Anmerkung verknüpfen

© 2016-2022 Altova GmbH

11.8 Objektdiagramm

Beziehung

Assoziation
Assoziationsklasse
Abhängigkeit
Verwendung
Schnittstellenrealisierung
Generalisierung

Hinzufügen

Paket
Klasse
Schnittstelle
Enumeration
Datentyp
Primitivtyp
Instanzspezifikation

Anmerkung
Anmerkung verknüpfen

11.9 Paketdiagramm

Hinzufügen

Paket
Profil

Beziehung

Abhängigkeit
Paketimport
Paketmerge
Profilzuweisung

Anmerkung
Anmerkung verknüpfen

© 2016-2022 Altova GmbH

11.10 Profildiagramm

Hinzufügen

Profil
Stereotyp

Beziehung

Generalisierung
Profilzuweisung
Paketimport
Elementimport

Anmerkung
Anmerkung verknüpfen

11.11 Protokoll-Zustandsdiagramm

Hinzufügen

Einfacher Zustand
Zusammengesetzter Zustand
Orthogonaler Zustand
Unterautomatenzustand

Endzustand
Anfangszustand

Eintrittspunkt
Austrittspunkt
Entscheidung
Kreuzung
Beendung
Gabelung
Gabelung (horizontal)
Vereinigung
Vereinigung (horizontal)
Verbindungspunktreferenz

Beziehung

Protocol Transition

Anmerkung
Anmerkung verknüpfen

© 2016-2022 Altova GmbH

11.12 Sequenzdiagramm

Hinzufügen

Lebenslinie
Combined Fragment
Combined Fragment (Alternativen)
Combined Fragment (Loop)
Interaktionsverwendung
Gate
Zustandsinvariante
Zeitdauerbedingung
Zeitbedingung

Nachricht (Aufruf)
Nachricht (Antwort)
Nachricht (Erstellung)
Nachricht (Löschung)

Asynchrone Nachricht (Aufruf)
Asynchrone Nachricht (Antwort)
Asynchrone Nachricht (Löschung)

Anmerkung
Anmerkung verknüpfen

Keine Nachrichtennummerierung
Einfache Nachrichtennummerierung
Hierarchische Nachrichtennummerierung

Verschieben zusammengehöriger Nachrichten ein/aus
Automatische Erstellung von Antworten auf Nachrichten (Call) ein/aus
Automatische Erstellung von Operationen in der Zielkomponente durch Eingabe von Operationsnamen ein-/ausschalten

11.13 Zustandsdiagramm

Hinzufügen

Einfacher Zustand
Zusammengesetzter Zustand
Orthogonaler Zustand
Unterautomatenzustand

Anfangszustand
Endzustand
Eintrittspunkt
Austrittspunkt
Entscheidung
Kreuzung
Beendung
Gabelung
Gabelung (horizontal)
Vereinigung
Vereinigung (horizontal)
Deep history
Shallow history
Verbindungspunktreferenz

Beziehung

Transition

Anmerkung
Anmerkung verknüpfen

Automatische Erstellung von Operationen in der Zielkomponente durch Eingabe von Operationsnamen ein-/ausschalten

© 2016-2022 Altova GmbH

11.14 Zeitverlaufsdiagramm

Hinzufügen

Lebenslinie (Zustand/Bedingung)
Lebenslinie (Allgemeiner Wert)
Tick-Symbol
Auslösendes Ereignis
Zeitdauerbedingung
Zeitbedingung

Nachricht (Aufruf)
Nachricht (Antwort)
Asynchrone Nachricht (Aufruf)

Anmerkung
Anmerkung verknüpfen

11.15 Use Case-Diagramm

Hinzufügen

Paket
Akteur
Use Case

Beziehung

Assoziation
Generalisierung
Include
Extend

Anmerkung
Anmerkung verknüpfen

© 2016-2022 Altova GmbH

11.16 XML-Schema-Diagramm

Hinzufügen

XSD Target Namespace
XSD Schema
XSD Element (global)
XSD Group
XSD ComplexType
XSD ComplexType (simpleContent)
XSD SimpleType
XSD List
XSD Union
XSD Enumeration
XSD Attribute
XSD AttributeGroup
XSD Notation
XSD Import

Beziehung

XSD Include
XSD Redefine
XSD Restriction
XSD Extension
XSD Substitution

Anmerkung
Anmerkung verknüpfen

12 Menüreferenz

Im folgenden Abschnitt sind alle Menüs und Menüoptionen in UModel mit einer kurzen Beschreibung dazu aufgelistet.

© 2016-2022 Altova GmbH

12.1 Datei

Neu

Löscht das Diagrammregister, falls ein früheres Projekt vorhanden ist, und erstellt ein neues UModel-Projekt.

Öffnen

Öffnet ein zuvor definiertes Modellierungsprojekt. Wählen Sie im Dialogfeld "Öffnen" eine zuvor gespeicherte Projektdatei (***.ump**) aus. Siehe Erstellen, Öffnen und Speichern von Projekten 153 und Öffnen von Projekten über eine URL 154.

Neu laden

Dient zum Neuladen des aktuellen Projekts und zum Speichern oder Verwerfen der Änderungen, die seit dem letzten Öffnen der Projektdatei vorgenommen wurden.

Speichern

Speichert das aktuell aktive Modellierungsprojekt unter dem aktiven Dateinamen.

Speichern unter

Speichert das aktuell aktive Modellierungsprojekt unter einem anderen Namen bzw. dient zum Speichern des Projekts unter einem Namen, wenn Sie das Projekt zum ersten Mal speichern.

Kopie speichern unter

Damit können Sie eine Kopie des aktuell aktiven UModel-Projekts unter einem anderen Dateinamen speichern.

Diagramm als Bild speichern

Öffnet das Dialogfeld "Speichern unter..." und speichert das aktuell aktive Diagramm als .png-Datei. Es können auch sehr große .png-Dateien im Gigabyte-Bereich gespeichert werden.

Alle Diagramme als Bilder speichern

Speichert alle Diagramme des derzeit aktiven Projekts als .png-Dateien.

Aus XMI-Datei importieren

Importiert eine zuvor exportierte XMI-Datei. Wenn die Datei mit UModel erzeugt wurde, werden alle Erweiterungen usw. beibehalten.

In XMI-Datei exportieren

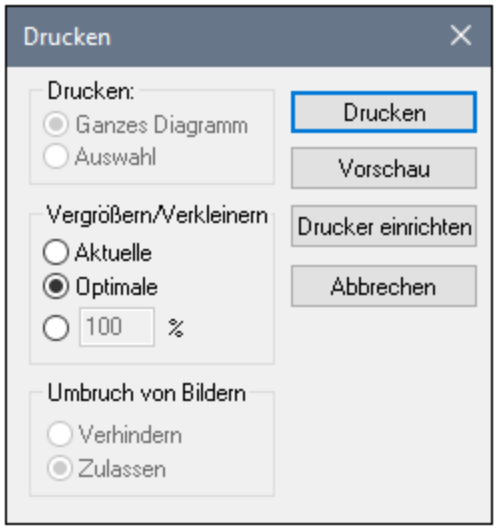
Exportiert das Modell als XMI-Datei. Sie können die UML-Version sowie die zu exportierenden IDs auswählen. Nähere Informationen siehe XMI - XML Metadata Interchange 454.

Als Mail senden

Öffnet Ihre Standard-Mailapplikation und fügt das aktuelle UModel Projekt als Anhang ein.

Drucken

Öffnet das Dialogfeld "Drucken", über das Sie das aktuelle Diagramm (oder eine Auswahl aus dem Diagramm) drucken können.



Bei Auswahl der Option **Aktuelle** wird der aktuell definierte Zoomfaktor des Modellierungsprojekts beibehalten und die Gruppe "Umbruch von Bildern" wird aktiviert. Bei Auswahl von **Optimale** wird das Modellierungsprojekt auf Seitengröße vergrößert/verkleinert. Sie können auch einen numerischen Zoom-Faktor angeben. Bei Auswahl der Option **Verhindern** wird verhindert, dass Modellelemente über Seiten hinweg umbrochen werden. Sie werden auf einer Seite angezeigt.

Alle Diagramme drucken

Öffnet das Dialogfeld "Drucken" und druckt alle in der aktuellen Projektdatei enthaltenen UML-Diagramme.

Druckvorschau

Öffnet dasselbe Druckdialogfeld mit denselben Einstellungen wie oben beschrieben.

Druckereinrichtung

Öffnet das Dialogfeld "Druckereinrichtung", in dem Sie einstellen können, welchen Drucker und welche Papiereinstellungen Sie verwenden möchten.

Letzte Dateien

In diesem Abschnitt des Menüs **Datei** werden bis zu vier zuletzt verwendete Dateien aufgelistet.

Beenden

Mit dem Befehl **Beenden** wird UModel beendet. Wenn eine Ihrer aktuellen Dateien nicht gespeicherte Änderungen enthält, fordert Sie UModel auf, die Änderungen zu speichern.

© 2016-2022 Altova GmbH

12.2 Bearbeiten

Rückgängig

UModel gestattet eine unbegrenzte Anzahl an Rückgängig-Schritten, sodass Sie Ihren Modellierungsvorgang Schritt für Schritt zurückverfolgen können.

Wiederherstellen

Mit Hilfe des Befehls "Wiederherstellen" können Sie zuvor rückgängig gemachte Befehle wiederholen. Sie können sich innerhalb des Verlaufs der rückgängig gemachten Schritte vorwärts und rückwärts bewegen.

Ausschneiden/Kopieren/Löschen

Diese sind die Windows-Standardtextbearbeitungsbefehle. Sie können diese nicht nur für Text, sondern auch für Modellierungselemente verwenden, siehe Umbenennen, Verschieben und Kopieren von Elementen 109.

Nur in Diagramm einfügen

Fügt einen Link (oder eine "Ansicht") des kopierten Elements zum aktuellen Diagramm nicht aber zur Modell-Struktur hinzu, siehe Umbenennen, Verschieben und Kopieren von Elementen 109.

Nur aus Diagramm löschen

Löscht die ausgewählten Modellelemente aus dem aktuell aktiven Diagramm. Die gelöschten Elemente werden nicht aus dem Modellierungsprojekt gelöscht und stehen auf dem Register "Modell-Struktur" zur Verfügung. Beachten Sie, dass diese Option nicht zum Löschen von Eigenschaften oder Operationen aus einer Klasse verfügbar ist. Eigenschaften oder Operationen können ausgewählt und direkt aus der Klasse gelöscht werden.

Alles markieren

Markiert alle Modellelemente des aktuell aktiven Diagramms. Entspricht der Tastenkombination Strg+A .

Suchen

Dient zum Suchen von bestimmtem Text im aktuellen Fenster, siehe Suchen und Ersetzen von Text 111.

Weitersuchen **F3**

Sucht die nächste Instanz desselben Suchstrings im derzeit aktiven Fenster.

Vorheriges suchen (Umschalt+F3)

Sucht die vorherige Instanz desselben Suchstrings auf dem derzeit aktiven Register oder im derzeit aktiven Diagramm.

Ersetzen

Dient zum Suchen und Ersetzen eines beliebigen Modellierungselements im Projekt, siehe Suchen und Ersetzen von Text 111.

Als Bitmap kopieren

Kopiert das derzeit aktive Diagramm in die Zwischenablage, von wo aus Sie es in die Applikation Ihrer Wahl einfügen können.

Auswahl als Bitmap kopieren

Kopiert die aktuell ausgewählten Diagrammelemente in die Zwischenablage, von wo aus Sie sie in die Applikation Ihrer Wahl einfügen können.

© 2016-2022 Altova GmbH

12.3 Projekt

Projektsyntax überprüfen

Überprüft die UModel-Projektsyntax. Siehe Überprüfen der Projektsyntax 173.

Versionskontrolle

Nähere Informationen und Anleitungen zu Versionskontrollservern und Clients finden Sie unter Versionskontrolle 456.

Quellverzeichnis importieren

Öffnet den "Quellverzeichnis importieren"-Assistenten. Ein Beispiel dazu finden Sie unter Reverse Engineering (Code zu Modell) 69.

Quellprojekt importieren

Öffnet den Assistenten "Quellprojekt importieren", siehe Importieren von Quellcode 194.

Binärtypen importieren

Öffnet das Dialogfeld "Binärtypen importieren", über das Sie Java-, C#- und VB-Binärdateien importieren können. Nähere Informationen dazu finden Sie unter Importieren von Java-, C#- und VB-Binärdateien 206.

XML-Schemaverzeichnis importieren

Öffnet das Dialogfeld "XML-Schemaverzeichnis importieren", über das Sie alle XML-Schemas in diesem Verzeichnis und optional dazu alle XML-Schemas in den Unterverzeichnissen dieses Verzeichnisses importieren können.

XML-Schema-Datei importieren

Öffnet das Dialogfeld "XML-Schema-Datei importieren" zum Importieren von Schema-Dateien. Nähere Informationen finden Sie unter XML-Schema-Diagramme 435.

Sequenzdiagramme von Code generieren...

Siehe Generieren mehrerer Sequenzdiagramme 377.

Code von Sequenzdiagrammen generieren

UModel kann Code anhand eines mit mindestens einer Operation verknüpften Sequenzdiagramms generieren. Nähere Informationen dazu finden Sie in diesem Abschnitt 378.

Zustandsautomatencode generieren

Sie können in UModel einen oder mehrere Zustandsautomaten auswählen, in denen Code generiert werden soll. Nähere Informationen finden Sie in diesem Kapitel 331.

Merge Programmcode aus UModel-Projekt / Überschreibe Programmcode aus UModel-Projekt

Aktualisiert Programmcode anhand des Modells (vorausgesetzt Ihr Projekt wurde für das Code Engineering vorbereitet, siehe Generieren von Programmcode 170). Der Name dieses Befehls kann entweder **Merge Programmcode aus UModel-Projekt** oder **Überschreibe Programmcode aus UModel-Projekt** lauten, je nachdem welche Einstellung im Dialogfeld "Synchronisierungseinstellungen" gewählt wurde. Das Dialogfeld "Synchronisierungseinstellungen" wird standardmäßig jedes Mal geöffnet, wenn Sie diesen Befehl aufrufen. Nähere Informationen dazu finden Sie unter Codesynchronisierungseinstellungen 225.

Merge UModel-Projekt aus Programmcode / Überschreibe UModel-Projekt aus Programmcode

Aktualisiert das Modell (das UModel-Projekt) anhand des Programmcodes. Der Name dieses Befehls kann entweder **Merge UModel-Projekt aus Programmcode** oder **Überschreibe UModel-Projekt aus Programmcode** lauten, je nachdem welche Einstellung im Dialogfeld "Synchronisierungseinstellungen" gewählt wurde. Das Dialogfeld "Synchronisierungseinstellungen" wird standardmäßig jedes Mal geöffnet, wenn Sie diesen Befehl aufrufen. Nähere Informationen dazu finden Sie unter Codesynchronisierungseinstellungen 225 .

Projekteinstellungen

Wenn Sie anhand von Programmcode ein UModel-Projekt generieren, können Sie die Projekteinstellungen 176 definieren bzw. ändern.

Synchronisierungseinstellungen

Öffnet das Dialogfeld "Synchronisierungseinstellungen", siehe Codesynchronisierungseinstellungen 225.

Projekt zusammenführen

Führt zwei UModel-Projektdateien in einem Modell zusammen. Die erste Datei, die Sie öffnen, ist die Datei, in die die zweite Datei überführt wird. Nähere Informationen dazu finden Sie unter Zusammenführen von UModel-Projekten 280.

Projekt zusammenführen (3-Weg)

UModel unterstützt die Zusammenführung mehrerer UModel-Projekte, die gleichzeitig von mehreren Entwicklern bearbeitet wurden, in einer 3-Weg-Projektzusammenführung 281.

Unterprojekt inkludieren

Siehe Inkludieren anderer UMdoel-Projekte 164.

Unterprojekt separat öffnen

Öffnet das ausgewählte Unterprojekt als neues Projekt.

© 2016-2022 Altova GmbH

Meldungen löschen

Löscht die Meldungen, Warnungen und Fehler, die im Fenster "Meldungen" 92 zur Syntaxüberprüfung und Codezusammenführung angezeigt werden.

Anmerkung: Fehlermeldungen weisen im Allgemeinen auf Probleme hin, die vor der Codegenerierung bzw. vor der Aktualisierung des Modellcodes während der Codegenerierung behoben werden müssen. Warnmeldungen weisen im Allgemeinen auf Probleme hin, die auch später behoben werden können. Fehlermeldungen und Warnmeldungen werden von der Syntaxüberprüfung, dem Compiler für die spezifische Sprache, dem UModel Parser, der die neu generierte Quelldatei liest, sowie beim Import von XMI-Dateien, generiert.

Dokumentation generieren

Dient zum Generieren von Dokumentation in den Formaten HTML, Microsoft Word und RTF für das derzeit offene Projekt. Nähere Informationen dazu finden Sie unter Generieren von UML-Dokumentation 290.

In keinem Diagramm verwendete Elemente auflisten

Erstellt eine Liste aller Elemente, die in keinem Diagramm des Projekts verwendet werden, siehe Überprüfen, wo und ob Elemente verwendet werden 113.

Freigegebene Pakete auflisten

Listet alle freigegebenen Pakete des aktuellen Projekts auf.

Inkludierte Pakete auflisten

Listet alle im aktuellen Projekt inkludierten Pakete auf.

12.4 Layout

Die Befehle des Menüs "Layout" gestatten Ihnen, die Elemente Ihrer Modelldiagramme anzuordnen und aneinander auszurichten, siehe Ausrichten von Modellierungselementen und Anpassen der Größe 128.

Ausrichten

Dieser Befehl dient je nach Auswahl des jeweiligen Befehls zum Ausrichten von Modellelementen entlang ihrer Ränder oder Mittelpunkte.

Gleichmäßig anordnen

Mit dieser Gruppe von Befehlen können Sie ausgewählte Elemente sowohl in horizontaler als auch vertikaler Richtung gleichmäßig anordnen.

Größe angleichen

Mit Hilfe dieser Befehle können Sie die Breite und Höhe der ausgewählten Elemente anhand des aktiven Elements anpassen.

Anordnen

Mit Hilfe dieser Gruppe von Befehlen können Sie die ausgewählten Elemente vertikal oder horizontal in einer Linie anordnen.

Linienart

Mit Hilfe dieser Gruppe von Befehlen können Sie die Art der Linie auswählen, mit der die verschiedenen Modellelemente verbunden werden sollen. Die Linien können jede Art von Abhängigkeit oder Assoziation darstellen, die in den verschiedenen Diagrammen verwendet werden.

Größe automatisch anpassen

Mit diesem Befehl werden die ausgewählten Elemente an die jeweils optimale Größe angepasst.

Automatisches Layout

Dieser Befehl dient zum Auswählen der Art, wie Modellelemente auf dem UML-Diagrammregister dargestellt werden sollen.

Zentriert	Zeigt die Modellierungselemente in der Mitte zentriert an.
Hierarchisch	Zeigt die Elemente nach ihren hierarchischen Beziehungen an. So wird z.B. eine übergeordnete Klasse oberhalb der davon abgeleiteten Klassen angezeigt. Die Optionen für hierarchisches Layout können über das Menü **Extras | Optionen,** Register **Ansicht,** Gruppe **Hierarchisch anordnen** angepasst werden.
Block	Zeigt die Elemente rechteckig nach Elementgröße gruppiert an.

 © 2016-2022 Altova GmbH

Textlabels neu positionieren

Positioniert die Namen der (ausgewählten) Modellelemente zurück an ihre Standardposition.

12.5 Ansicht

Mit Hilfe der Befehle in diesem Menü können Sie:

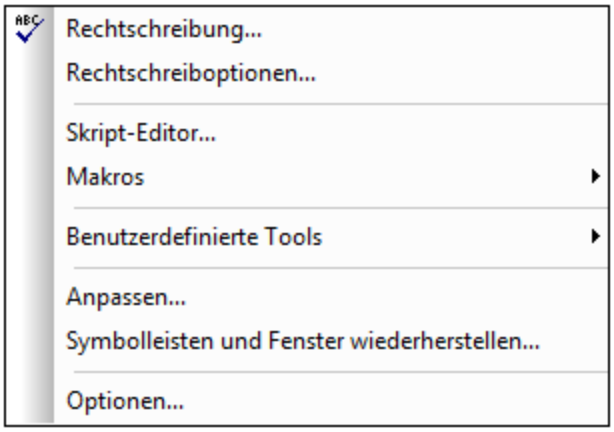
- jedes beliebige der UModel-Hilfsfenster ein- oder ausblenden, siehe Grafische Benutzeroberfläche von UModel 78
- die Sortierkriterien für die Modellelemente der Fenster "Modell-Struktur" 80 und "Favoriten" 85 definieren
- die Gruppierungskriterien der Diagramme im Fenster "Diagramm-Struktur" 84 definieren
- bestimmte UML-Elemente in den Fenstern "Favoriten" und "Modell-Struktur" ein- oder ausblenden
- den Zoomfaktor des aktuellen Diagramms definieren, siehe Vergrößern und Verkleinern von Diagrammen 132.

© 2016-2022 Altova GmbH

12.6 Extras

Mit den Befehlen im Menü "Extras" können Sie:

-
- Ihre Benutzeroberfläche anpassen 515: Ihre eigenen Symbolleisten, Tastaturkürzel, Menüs und Makros definieren
- die Symbolleisten und Fenster wieder in den Originalzustand versetzen.
- die globalen Programmeinstellungen/optionen 525 definieren.



12.6.1 Benutzerdefinierte Tools

Wenn Sie den Mauszeiger über den Befehl **Benutzerdefinierte Tools** platzieren, wird ein Untermenü mit benutzerdefinierten Befehlen angezeigt, die externe Applikationen verwenden. Sie können diese Befehle im Dialogfeld "Anpassen" auf dem Register "Extras" 518 erstellen. Wenn Sie auf einen dieser benutzerdefinierten Befehle klicken, wird die mit diesem Befehl verknüpfte Aktion ausgeführt.

Der Befehl **Benutzerdefinierte Tools | Anpassen** öffnet das Register "Extras" 518 im Dialogfeld "Anpassen" (Hier können Sie die benutzerdefinierten Befehle, die im Menü des Befehls **Benutzerdefinierte Tools** angezeigt werden, erstellen).

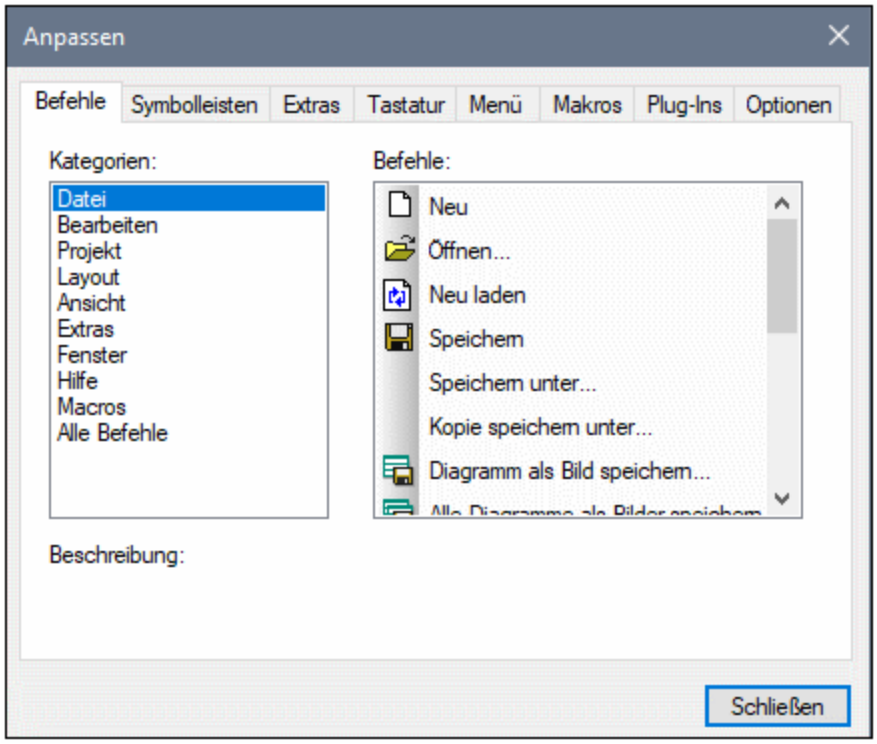
12.6.2 Anpassen

Mit dem Befehl **Anpassen** rufen Sie ein Dialogfeld auf, in dem Sie UModel an Ihre persönlichen Bedürfnisse anpassen können. Sie können folgende Dinge anpassen:

- Befehle 516
- Symbolleisten 517
- Extras 518
- Tastatur 521
- Menü 523
- Optionen 524

12.6.2.1 Befehle

Auf dem Register "**Befehle**" können Sie Ihre UModel-Menüs oder Symbolleisten anpassen.



So fügen Sie einen Befehl zu einer Symbolleiste oder einem Menü hinzu:

1. Klicken Sie im Menü **Extras** auf **Anpassen.**
2. Wählen Sie im Listenfeld "**Kategorien**" die Befehlskategorie aus. Die verfügbaren Befehle werden im Listenfeld "**Befehle**" angezeigt.
3. Klicken Sie auf einen Befehl in der Liste der **Befehle** und ziehen Sie ihn in ein vorhandenes Menü bzw. eine vorhandene Symbolleiste. Wenn Sie den Cursor über eine Position halten, an die der Befehl gezogen werden kann, wird ein I-Zeichen angezeigt.
4. Lassen Sie die Maustaste an der Position los, an der der Befehl eingefügt werden soll. An der Spitze des Mauszeigers erscheint beim Ziehen des Befehls eine kleine Schaltfläche. Das Häkchen unterhalb des Mauszeigers bedeutet, dass der Befehl nicht an der aktuellen Cursorposition abgelegt werden kann. An den Stellen, an die der Befehl gezogen werden kann (also über der Symbolleiste oder einem Menü) verschwindet das Häkchen.

Anmerkungen:

- Wenn Sie den Cursor beim Ziehen über ein Menü ziehen, wird dieses geöffnet, sodass Sie den Befehl an einer beliebigen Stelle im Menü ablegen können.
- Befehle können in Menüs oder Symbolleisten platziert werden. Wenn Sie Ihre eigene Symbolleiste erstellt haben, können Sie darin Ihre eigenen Befehle/Symbole einordnen.

 © 2016-2022 Altova GmbH

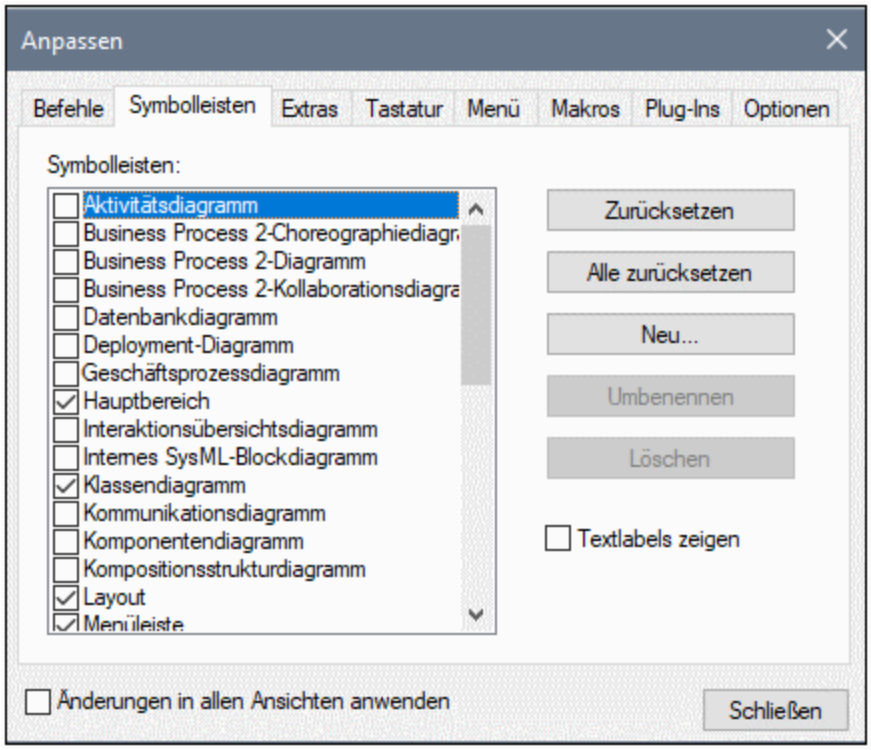
- Auf dieselbe Weise können Sie auch die Befehle in den Kontextmenüs 523 (wird durch Rechtsklick aufgerufen) bearbeiten. Klicken Sie auf das Register "Menü" und wählen Sie anschließend aus der Liste der Kontextmenüs das jeweilige Kontextmenü aus.

So löschen Sie einen Befehl oder ein Menü:

1. Klicken Sie im Menü **Extras** auf **Anpassen.**
2. Klicken Sie auf den Menüeintrag bzw. das zu löschende Symbol und ziehen Sie es mit der Maus.
3. Lassen Sie die Maustaste los, sobald das Häkchensymbol unterhalb des Mauszeigers erscheint. Der Befehl bzw. der Menüeintrag wird aus dem Menü bzw. der Symbolleiste gelöscht.

12.6.2.2 Symbolleisten

Auf dem Register "**Symbolleisten**" können Sie bestimmte Symbolleisten aktivieren oder deaktivieren sowie Ihre eigenen Symbolleisten erstellen.



Symbolleisten enthalten Befehle für die am häufigsten verwendeten Menübefehle. Zu jedem Symbol erhalten Sie eine kurze Erklärung in Form eines Tooltipps, wenn Sie den Mauszeiger direkt über das Element platzieren. In der Statusleiste wird eine detailliertere Beschreibung des Befehls angezeigt. Sie können die Symbolleisten von ihrer Standardposition an eine beliebige Stelle auf dem Bildschirm ziehen, wo sie als abgedocktes Fenster angezeigt werden. Alternativ dazu können Sie die Symbolleisten auch am linken oder rechten Rand des Hauptfensters andocken.

So aktivieren oder deaktivieren Sie eine Symbolleiste:

- Klicken Sie auf das Kontrollkästchen, um die jeweilige Symbolleiste zu aktivieren (bzw. zu deaktivieren).

So erstellen Sie eine neue Symbolleiste:

1. Klicken Sie auf die Schaltfläche **Neu...** und geben Sie der Symbolleiste im Dialogfeld "Symbolleistenname" einen Namen.
2. Fügen Sie über das Register Befehle 516 des Dialogfelds "Anpassen" Befehle zur Symbolleiste hinzu.

So setzen Sie die Menüleiste zurück

1. Klicken Sie auf den Menüleisteneintrag und
2. anschließend auf die Schaltfläche **Zurücksetzen**, um die Menübefehle auf den Zustand zum Zeitpunkt der Installation zurückzusetzen

So setzen Sie alle Symbolleisten- und Menübefehle zurück

1. Klicken Sie auf die Schaltfläche **Alle zurücksetzen**, um alle Symbolleistenbefehle auf den Zustand zum Zeitpunkt der Installation des Programms zurückzusetzen. Sie werden gefragt, ob alle Symbolleisten und Menüs zurückgesetzt werden sollen.
2. Klicken Sie auf "**Ja**", um den Vorgang zu bestätigen.

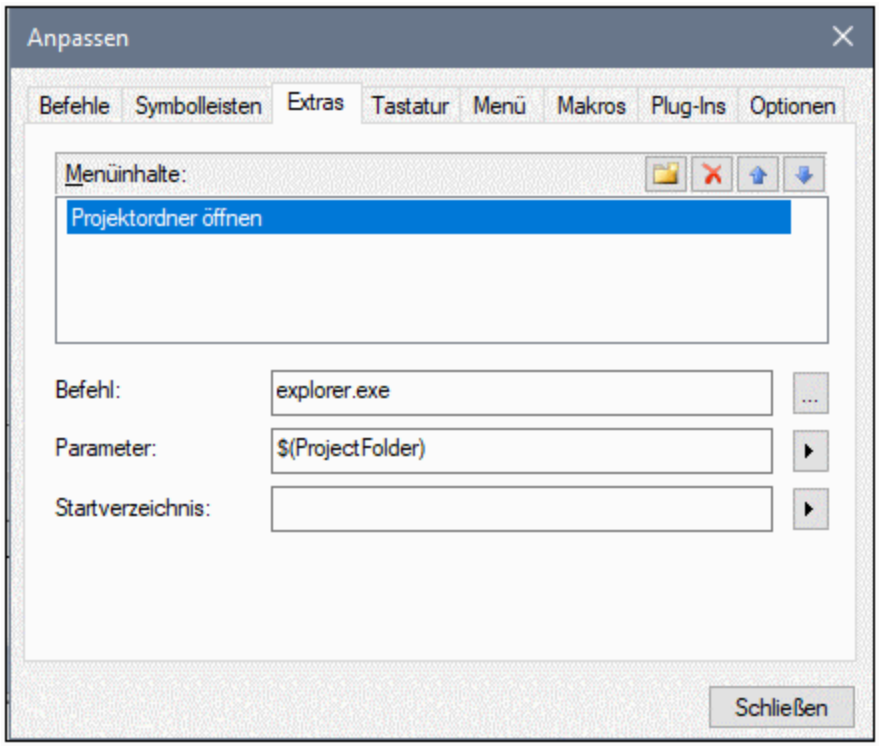
Bei Aktivierung der Option **Textlabels zeigen** wird erklärender Text unterhalb der Symbolleistenbefehle angezeigt.

12.6.2.3 Extras

Über das Register **Extras** können Sie benutzerdefinierte Menübefehle erstellen, mit denen Sie externe Tools direkt von UModel aus starten können. Die hier definierten Menübefehle werden im Menü **Extras | Benutzerdefinierte Tools** angezeigt. Bei externen Tools kann es sich um in Windows enthaltene Programme, wie Windows Explorer (**explorer.exe**), Notepad (**notepad.exe**) oder andere benutzerdefinierte ausführbare Dateien handeln. Sie können jedem benutzerdefinierten Tool optional Parameter zuweisen und das Verzeichnis, in dem das externe Tool initialisiert werden soll, definieren (um nach relativen Dateinamen suchen zu können).

So wird etwa mit der unten gezeigten Konfiguration ein neuer Menübefehl namens "Projektordner öffnen" hinzugefügt. Wenn Sie den Befehl starten, wird damit das Verzeichnis des aktuellen UModel-Projekts in Windows Explorer geöffnet.

 © 2016-2022 Altova GmbH



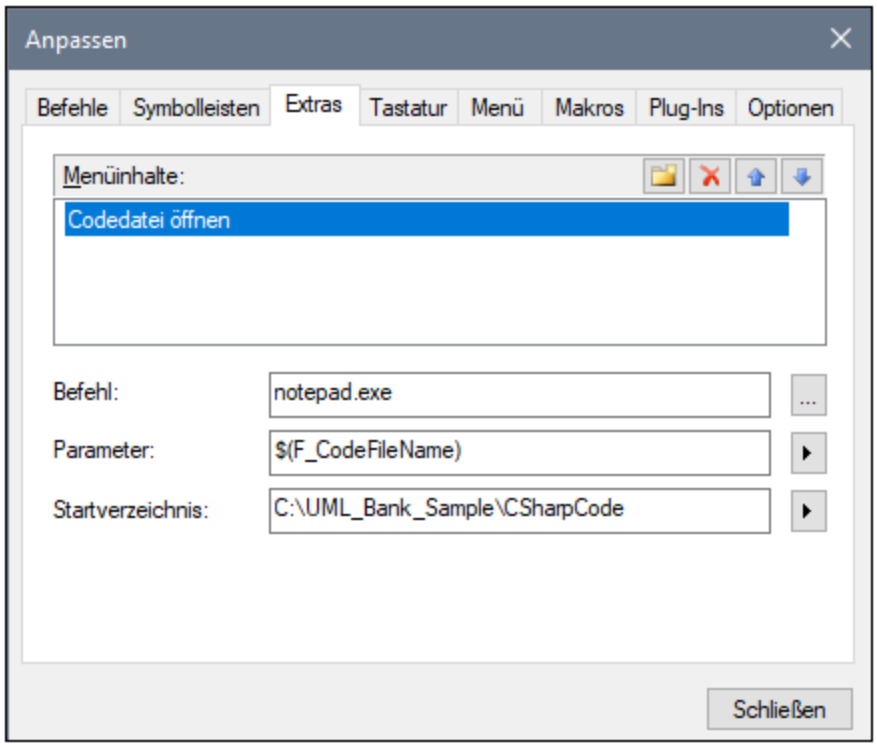
Wenn ein externes Tool Parameter (wie z.B. im Beispiel oben Windows Explorer erhält), können diese in das Eingabefeld "Parameter" eingegeben werden. Bei Eingabe mehrerer Parameter, müssen diese durch ein Leerzeichen getrennt werden. Bei den Werten, die Sie als Parameter bereitstellen können, kann es sich um reinen Text (hartcodierte Werte) handeln oder sie können mit der Schaltfläche ▸ aus einer Liste vordefinierte UModel-Variablen ausgewählt werden. Sie können jede der folgenden vordefinierten UModel-Variablen als Parameter verwenden:

Vordefinierte UModel-Variable	Aufgabe
Projektdateiname	Der Dateiname der aktiven UModel-Projektdatei, z.B. **Test.ump**.
Projektdateipfad	Der absolute Dateipfad der aktiven UModel-Projektdatei, z.B. **C:\MyDirectory\Test.ump**.
Fokussierte UML-Daten - Name	Der Name des UML-Elements, das sich im Fokus befindet, z.B. **Klasse1**.
Fokussierte UML-Daten – Qualifizierter UML-Name	Der qualifizierte Name des UML-Elements, das sich im Fokus befindet, z.B. **Paket1::Paket2::Klasse1**.
Fokussierte UML-Daten – Codedateiname	Der Codedateiname der UML-Klasse, -Schnittstelle oder Enumeration, die sich gerade im Fokus befindet, der im Fenster "Eigenschaften" (relativ zur realisierenden

Vordefinierte UModel-Variable	Aufgabe
	Komponente) angezeigt wird, z.B. **Klasse1.cs** oder **MeinNamespace\Klasse1.Java**.
Fokussierte UML-Daten – Codedateipfad	Der Codedateipfad der UML-Klasse, -Schnittstelle oder Enumeration, die sich gerade im Fokus befindet, der im Fenster "Eigenschaften" angezeigt wird, z.B., **C:\Temp\MySource\Klasse1.cs**.
Fokussierte UML-Daten – Codeprojektdateiname	Der Dateiname des Codeprojekts, zu dem die UML-Klasse, Schnittstelle oder Enumeration, die sich gerade im Fokus befindet, gehört. Der Codeprojektdateiname kann relativ zur UModel-Projektdatei sein und ist mit dem in den Eigenschaften der Komponente angezeigten identisch, z.B. **C:\Temp\MySource\MeinProjekt.vcproj** oder **MySource\MeinProjekt.vcproj**.
Fokussierte UML-Daten – Codeprojektdateipfad	Der Dateipfad des Codeprojekts, zu dem die UML-Klasse, Schnittstelle oder Enumeration, die sich gerade im Fokus befindet, gehört, z.B. **C:\Temp\MySource\MeinProjekt.vcproj**.
Projektordner	Das Verzeichnis, in dem das aktuelle UModel-Projekt gespeichert ist, z.B. **C:\Benutzer\<Benutzer>\Dokumente\Altova\UModel2023\UModelExamples**.
Temporärer Ordner	Das Verzeichnis, in dem die temporären Dateien der Applikation gespeichert sind, z.B. **C:\Benutzer\<Benutzer>\AppData\Local\Temp**.

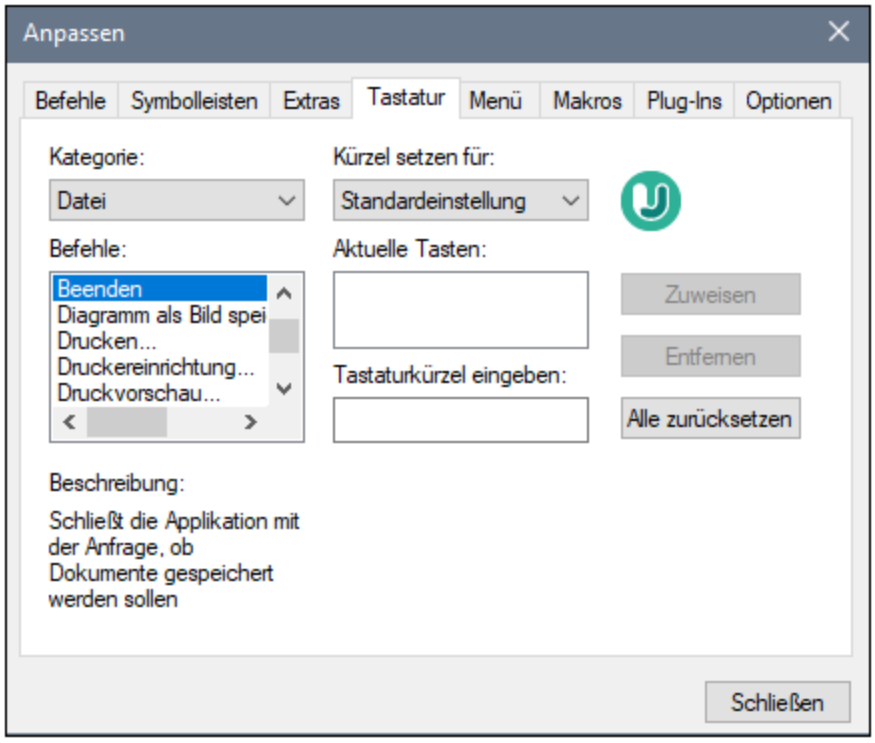
In einigen Fällen müssen Sie eventuell auch einen Wert in das Eingabefeld **Startverzeichnis** eingeben. So wird z.B. in der Konfiguration unten die Codedatei des aktuell in einem Diagramm ausgewählten Elements in Notepad geöffnet. (Beachten Sie: Damit dieser Befehl funktioniert, muss für das im Diagramm aktuell ausgewählte Element ein Wert (Dateiname) im Feld Codedateiname des Fensters "Eigenschaften" 86 definiert sein und diese Datei muss im Verzeichnis **C:\UML_Bank_Sample\CSharpCode** vorhanden sein).

 © 2016-2022 Altova GmbH



12.6.2.4 Tastatur

Auf dem Register "**Tastatur**" können Sie Tastaturkürzel für jeden beliebigen Befehl definieren (oder ändern).



So weisen Sie einem Befehl ein neues Tastaturkürzel zu:

1. Wählen Sie in der Auswahlliste **Kategorie** die Befehlskategorie aus.
2. Wählen Sie in der Liste **Befehle** den **Befehl** aus, dem Sie ein neues Tastaturkürzel zuweisen möchten.
3. Klicken Sie in das Textfeld **Tastaturkürzel eingeben**: und drücken Sie die Tasten, mit denen dieser Befehl aufgerufen werden soll. Die Tastenkombination wird sofort im Textfeld angezeigt. Falls diese Tastenkombination bereits zugewiesen wurde, so wird diese Funktion unterhalb des Textfelds angezeigt.
4. Klicken Sie auf die Schaltfläche **Zuweisen**, um das Tastaturkürzel permanent zuzuweisen.
   Das Tastaturkürzel erscheint nun im Listenfeld **Aktuelle Tasten**.
   (Um den Eintrag in diesem Textfeld zu **löschen,** drücken Sie eine der Steuerungstasten: **Strg, Alt** oder **Umschalttaste**.)

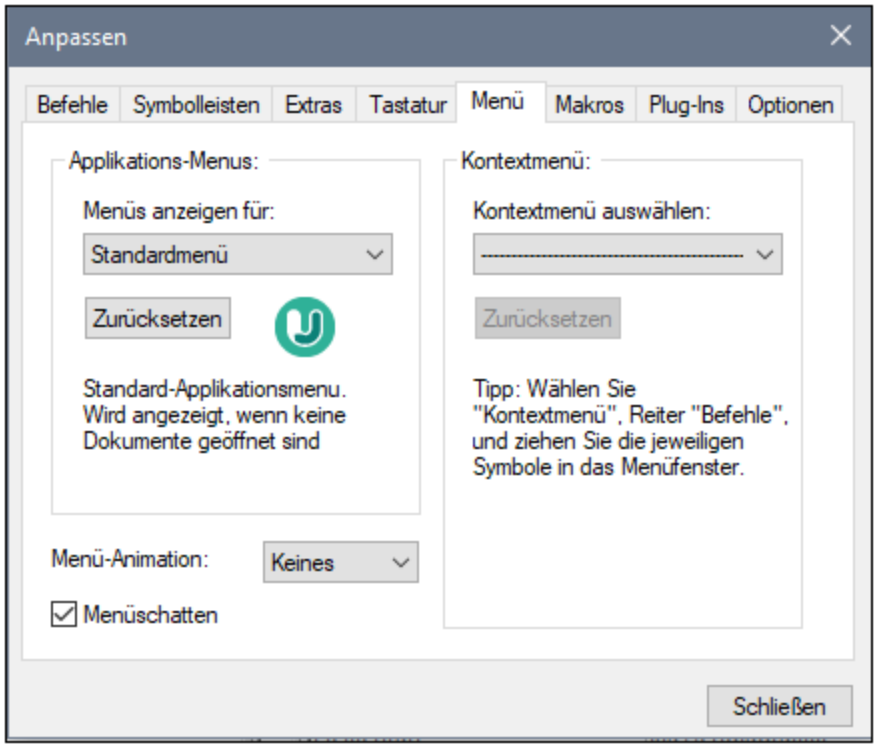
So heben Sie eine Tastenzuweisung auf (oder löschen ein Tastaturkürzel):

1. Klicken Sie in der Liste **Aktuelle Tasten** auf das zu löschende Tastaturkürzel.
2. Klicken Sie auf die jetzt aktiv gewordene Schaltfläche **Entfernen.**
3. Klicken Sie auf die Schaltfläche **Schließen**, um alle im Dialogfeld "Anpassen" vorgenommenen Änderungen zu bestätigen.

© 2016-2022 Altova GmbH

12.6.2.5 Menü

Auf dem Register **Menü** können Sie die Menüleisten sowie die Kontextmenüs anpassen.



Anpassen von Menüs

Die Menüleiste **Standardmenü** ist die Menüleiste, die angezeigt wird, wenn kein Projektfenster geöffnet ist. Die Menüleiste **UModel-Projekt** ist die Menüleiste, die angezeigt wird, wenn ein Projekt geöffnet ist. Jede Menüleiste kann separat angepasst werden. Änderungen, die an einer der Leisten vorgenommen wurden, haben keine Auswirkung auf andere Menüleisten.

Um eine Menüleiste anzupassen, wählen Sie diese in der Auswahlliste **Menüs anzeigen für** aus. Klicken Sie anschließend auf das Register **Befehle** und ziehen Sie die Befehle aus dem Listenfeld **Befehle** in die Menüleiste oder in eines der Menüs.

Löschen von Befehlen aus Menüs und Zurücksetzen der Menüleisten

So löschen Sie ein ganzes Menü oder einen Befehl in einem Menü:

1. Wählen Sie die gewünschte Menüleiste aus der Dropdown-Liste **Menüs anzeigen für** aus.
2. Während das Dialogfeld "Anpassen" geöffnet ist, (i) wählen Sie das Menü, das Sie aus der Menüleiste der Applikation löschen möchten, aus oder (ii) wählen Sie den Befehl aus, den Sie aus einem dieser Menüs löschen möchten.
3. Ziehen Sie entweder das Menü aus der Menüleiste oder den Menübefehl aus dem Menü oder (ii) klicken Sie mit der rechten Maustaste auf das Menü oder den Menübefehl und wählen Sie den Befehl **Löschen**.

Sie können jede Menüleiste in den Originalzustand zurücksetzen. Wählen Sie sie dazu aus der Dropdown-Liste **Menüs anzeigen für** aus und klicken Sie anschließend auf die Schaltfläche **Zurücksetzen**.

Anpassen der Kontextmenüs der Applikation

Die Kontextmenüs sind die Menüs, die angezeigt werden, wenn Sie mit der rechten Maustaste auf bestimmte Objekte auf der Benutzeroberfläche der Applikation klicken. Jedes dieser Kontextmenüs kann folgendermaßen angepasst werden:

1. Wählen Sie das Kontextmenü in der Dropdown-Liste **Kontextmenü auswählen** aus. Daraufhin wird das Kontextmenü angezeigt.
2. Klicken Sie auf das Register **Befehle**.
3. Ziehen Sie den gewünschten Befehl aus dem Listenfeld **Befehle** in das Kontextmenü.
4. Um einen Befehl aus dem Kontextmenü zu löschen, klicken Sie mit der rechten Maustaste auf diesen Befehl im Kontextmenü und wählen Sie den Befehl **Löschen**. Ziehen Sie den Befehl alternativ dazu mit der Maus aus dem Kontextmenü heraus.

Sie können jedes Kontextmenü in den Originalzustand zurücksetzen. Wählen Sie es dazu aus der Dropdown-Liste **Kontextmenü auswählen** aus und klicken Sie anschließend auf die Schaltfläche **Zurücksetzen**.

Menüschatten

Aktivieren Sie das Kontrollkästchen **Menüschatten**, wenn Menüs mit Schatten dargestellt werden sollen.

Wenn Sie animierte Menüs vorziehen, haben Sie die Wahl zwischen einer Reihe von Menüanimationen. Die Dropdown-Liste **Menu-Animation** enthält die folgenden Optionen:

- Keines (Standard)
- Entfalten
- Entrollen
- Abblenden

12.6.2.6 Optionen

Auf dem Register **Optionen** können Sie allgemeine Einstellungen vornehmen.

Wenn das Kontrollkästchen **Tooltip in Symbolleiste einblenden** aktiv ist, wird ein Tooltipp angezeigt, wenn der Mauszeiger über eine Symbolleisten-Schaltfläche platziert wird. In diesem Tooltipp wird eine kurze Beschreibung der Schaltflächenfunktion sowie das dazugehörige Tastaturkürzel, falls eines zugewiesen wurde, angezeigt.

Wenn das Kontrollkästchen **Große Symbole** aktiv ist, werden anstatt der Standardsymbole größere Symbole angezeigt.

12.6.3 Symbolleisten und Fenster wiederherstellen

Mit diesem Befehl wird UModel geschlossen und mit den Standardeinstellungen neu gestartet. Vor dem Schließen wird ein Dialogfeld angezeigt, in dem Sie gebeten werden, den Befehl zu bestätigen

 © 2016-2022 Altova GmbH

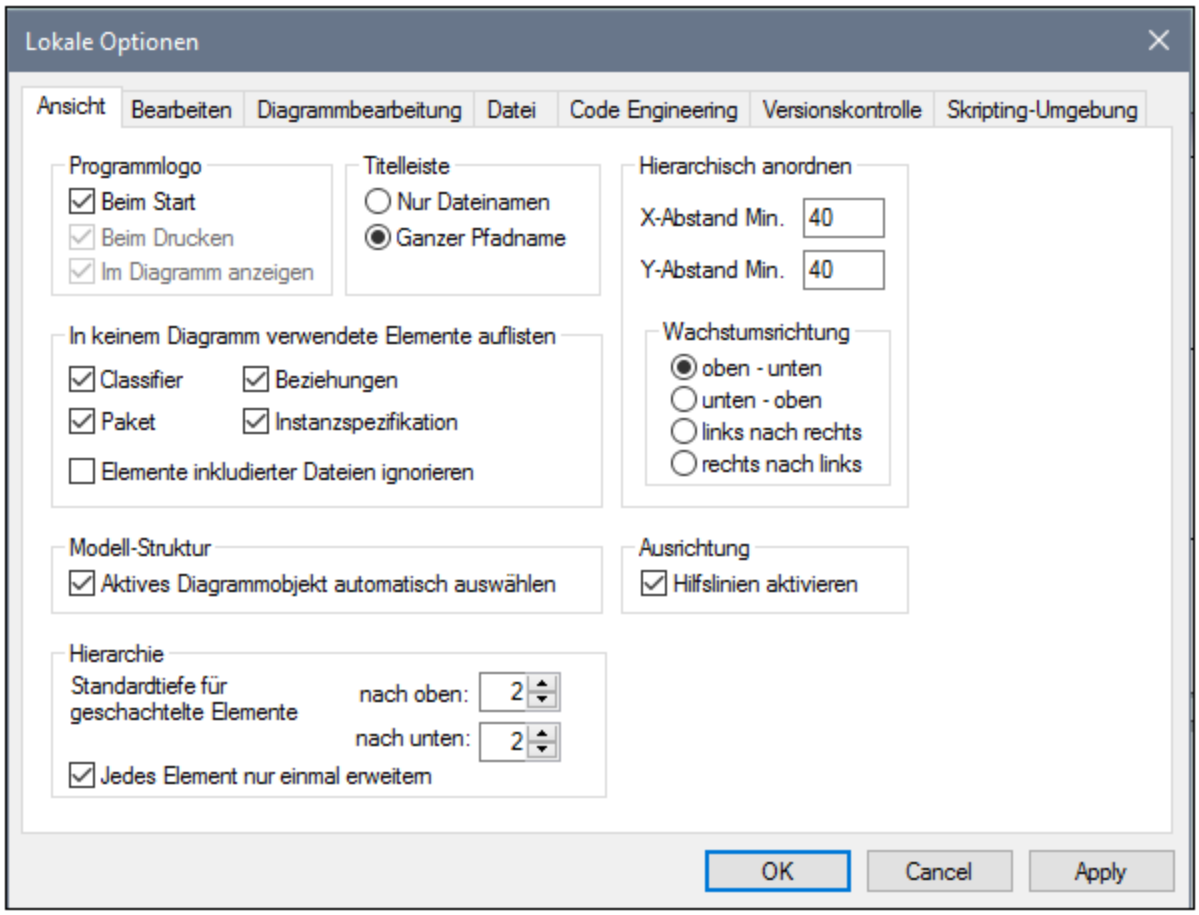
Dieser Befehl ist hilfreich, wenn Sie Symbolleisten oder Fenster verschoben oder ausgeblendet oder deren Größe angepasst haben und nun alle Originaleinstellungen wiederherstellen möchten.

12.6.4 Optionen

Wählen Sie den Menüeintrag **Extras | Optionen,** um Ihre Projektoptionen zu definieren.

Auf dem Register **Ansicht** können Sie folgende Einstellungen festlegen:

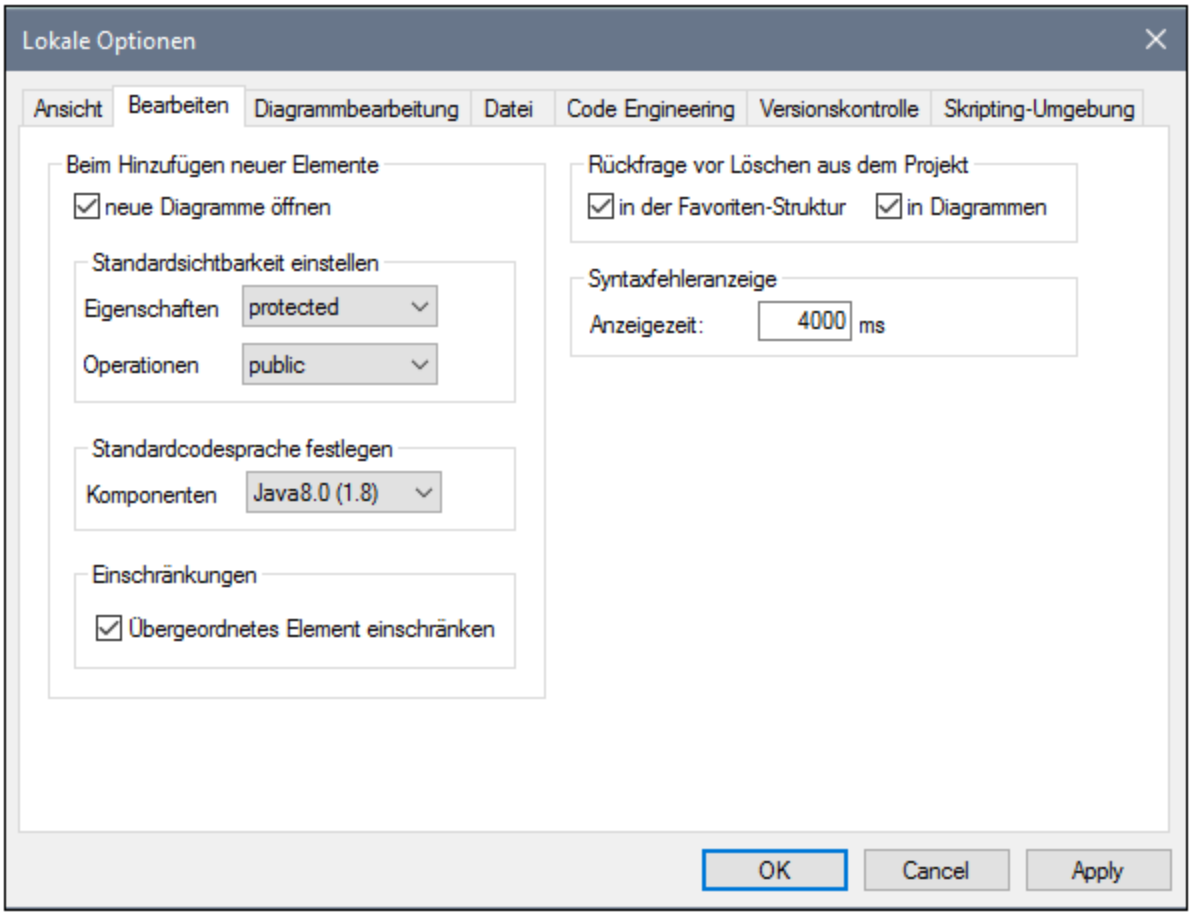
- wo das Programmlogo angezeigt werden soll.
- den Inhalt der Applikationstitelleiste
- die Elementarten, die aufgelistet werden sollen, wenn Sie auf dem Register "Model-Struktur" oder "Favoriten" auf die Kontextmenüoption "In keinem Diagramm verwendete Elemente auflisten" klicken. Sie haben auch die Möglichkeit, Elemente in **inkludierten** Dateien zu ignorieren.
- ob ein in einem Diagramm ausgewähltes Element in der Modell-Struktur automatisch ausgewählt/synchronisiert werden soll.
- die Standardtiefe der hierarchischen Ansicht auf dem Register "Hierarchie" bei Verwendung der Option **Graph. Ansicht anzeigen** .
- Die Einstellungen für automatisches hierarchisches Layout, über Sie für das Hierarchiefenster die Verschachtelungstiefe nach oben und unten festlegen können.
- "Jedes Element nur einmal erweitern". Dadurch kann im selben Bild/Diagramm kann nur eines derselben Classifier-Elemente erweitert werden.
- ob Sie Elemente beim Ziehen mit der Maus in einem Diagramm an Hilfslinien ausrichten möchten.



Auf dem Register **Bearbeiten** können Sie folgende Einstellungen definieren:

- Ob ein neues auf dem Register "Modell-Struktur" erstelltes Diagramm im Hauptfenster automatisch geöffnet werden soll.
- Standardsichtbarkeit beim Hinzufügen neuer Elemente - Eigenschaften oder Operationen.
- die Standard-Codesprache, wenn eine neue Komponente hinzugefügt wird.
- Ob eine neu hinzugefügte Einschränkung auch ihren Besitzer (Owner) einschränken soll.
- Ob Sie gefragt werden möchten, bevor Elemente aus einem Projekt, vom Register **Favoriten** oder aus einem der Diagramme gelöscht werden. Wenn Sie Elemente hier löschen, kann diese Eingabeaufforderung deaktiviert werden. Sie können die Eingabeaufforderung "vor dem Löschen fragen" hier zurücksetzen.
- Die Zeit nach der Syntaxfehler-Popup-Fenster geschlossen werden sollen.

© 2016-2022 Altova GmbH

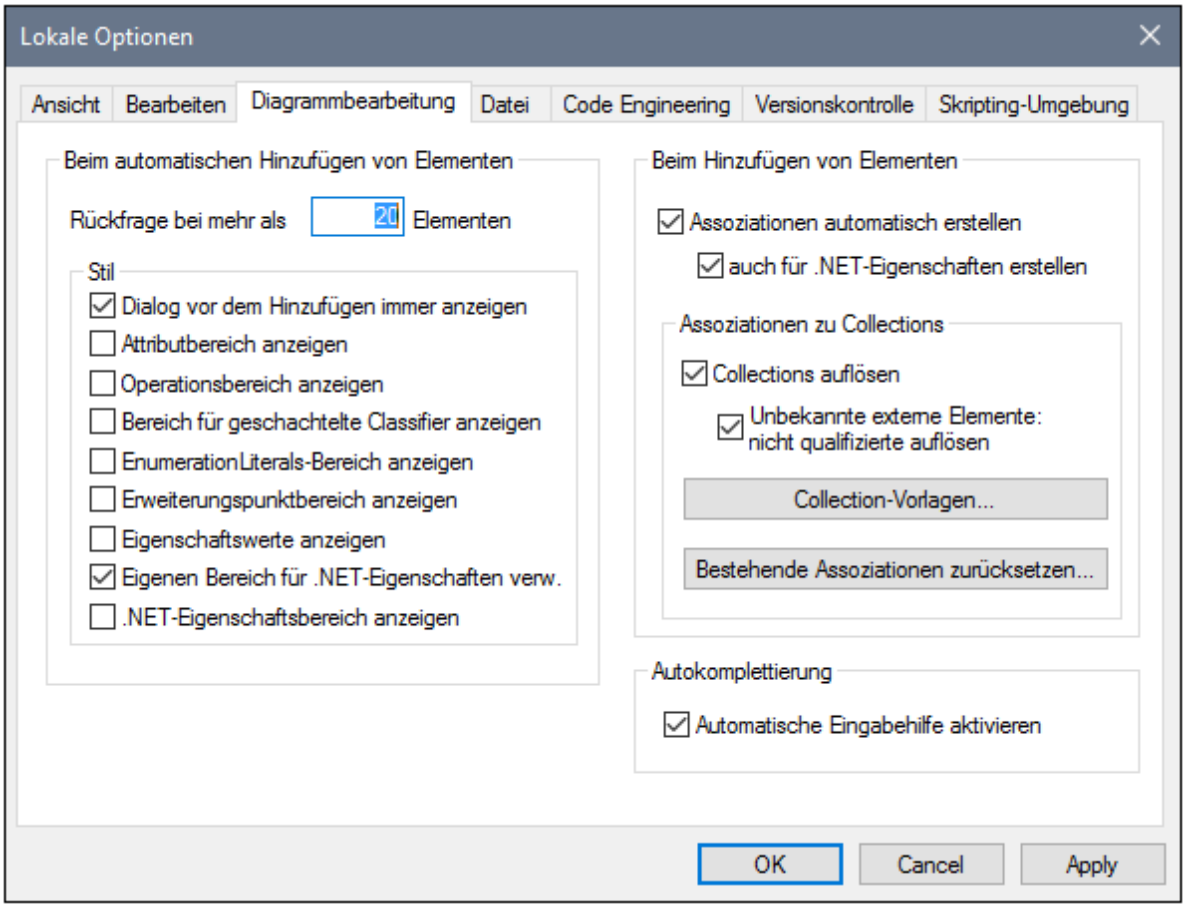


Auf dem Register **Diagrammbearbeitung** können Sie folgende Einstellungen definieren:

- wie viele Objekte automatisch zu einem Diagramm hinzugefügt werden können, bevor eine Eingabeaufforderung erscheint.
- die Anzeige von Stilen, wenn diese automatisch zu einem Diagramm hinzugefügt werden.
- ob automatisch Assoziationen zwischen Modellelementen erstellt werden sollen, wenn Elemente zu einem Diagramm hinzugefügt werden.
- ob die Assoziationen zu Collections aufgelöst werden sollen.
- ob Vorlagen von unbekannten externen Elementen als nicht vollständig qualifiziert aufgelöst werden sollen.
- ob bestehende Collection-Vorlagen verwendet oder neue definiert werden sollen.

  Collection-Vorlagen sollten als voll qualifiziert definiert werden, also a.b.c.List. Wenn die Vorlage diesen Namespace hat, erstellt UModel automatisch eine Collection-Assoziation. Ausnahme: Wenn die Vorlage zum Paket der unbekannten externen Elemente gehört und die Option "Unbekannte externe Elemente: nicht qualifizierte auflösen" aktiviert ist, so wird nur der Vorlagenname berücksichtigt (d.h. List anstelle von a.b.c.List).

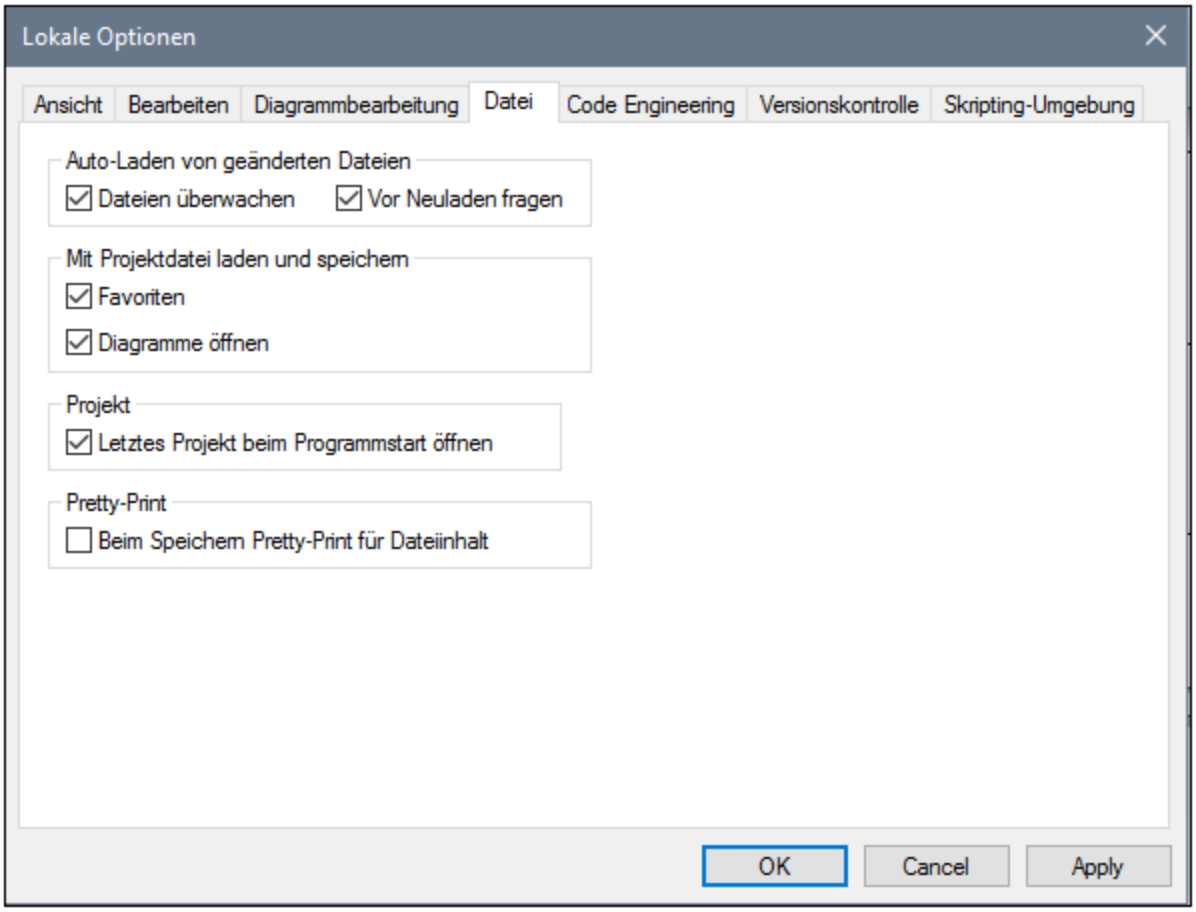
- Ob das Autokomplettierungsfenster bei der Bearbeitung von Attributen oder Operationen im Klassendiagramm verfügbar sein soll.



Auf dem Register **Datei** können Sie folgende Einstellungen definieren:

- welche Aktionen durchgeführt werden sollen, wenn Dateien geändert werden.
- ob der Inhalt des Registers "Favoriten" sowie alle derzeit offenen Diagramme mit dem aktuellen Projekt geladen und gespeichert werden soll.
- ob das zuletzt geöffnete Projekt beim Starten der Applikation automatisch geöffnet werden soll.
- ob die Projektdatei mit CR/LF-Zeilen und Tabulatoreinrückungen im Pretty-Print-Format gespeichert werden soll.

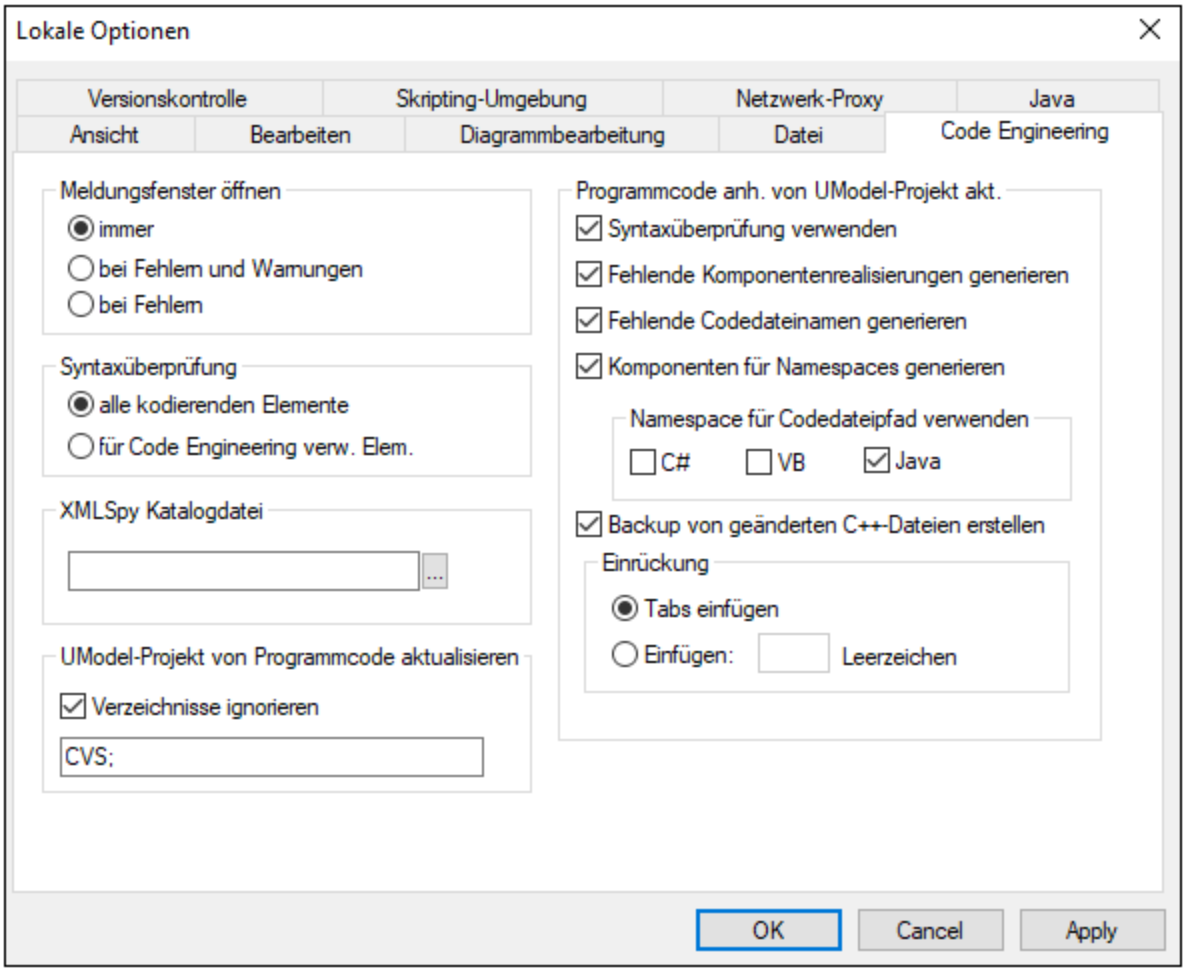
 © 2016-2022 Altova GmbH



Auf dem Register **Code Engineering** können Sie folgende Parameter definieren:

- die Umstände, unter denen das Meldungsfenster angezeigt werden soll.
- ob **alle Code-Elemente**, d.h. jene, die in einer Java-, C# - oder VB-Namespace Root enthalten sind, sowie jene, die einer Java-, C#- oder VB-Komponente zugewiesen sind, überprüft werden sollen oder ob nur **Elemente, die für das Code Engineering verwendet werden**, d.h. Elemente, bei denen das Kontrollkästchen "für Code Engineering verwenden" aktiv ist, überprüft werden sollen.
- ob im zusammenzuführenden Code fehlende Codedateinamen generiert werden sollen.
- Beim Aktualisieren von Programmcode:
  - ob eine Syntaxüberprüfung durchgeführt werden soll.
  - ob fehlende Komponentenrealisierungen automatisch generiert werden sollen.
  - ob im zusammengeführten Code fehlende Codedateinamen generiert werden sollen
  - ob im Codedateipfad Namespaces verwendet werden sollen
- Die im Code verwendete Einrückungsmethode, d.h. Tabulatoren oder beliebig viele Leerzeichen
- welche Verzeichnisse beim Aktualisieren eines UModel-Projekts anhand von Code oder einem Verzeichnis ignoriert werden sollen. Trennen Sie die einzelnen Verzeichnisse durch ein Semikolon "**;**". Untergeordnete Verzeichnisse desselben Namens werden ebenfalls ignoriert.
- den Pfad zur XMLSpy-Datei **RootCatalog.xml**, welche es UModel sowie XMLSpy erlaubt, allgemein verwendete Schemas (sowie Stylesheets und andere Dateien) aus lokalen Benutzerordnern aufzurufen. Dadurch wird die allgemeine Verarbeitungsgeschwindigkeit erhöht und Benutzer können auf diese Art auch offline arbeiten.

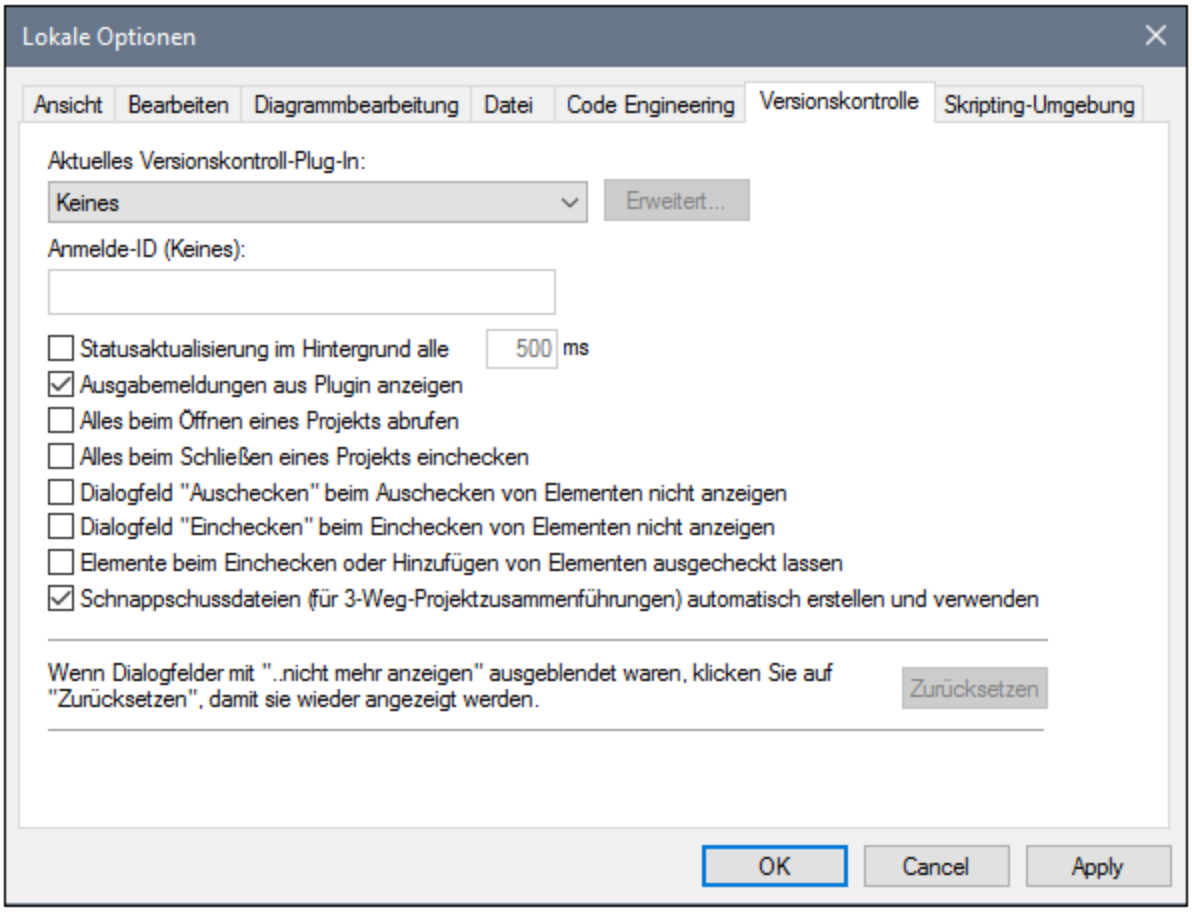
- Außerdem können Sie festlegen, ob eine Sicherungskopie von geänderten C++-Dateien angelegt werden soll.



Auf dem Register **Versionskontrolle** können Sie folgende Einstellungen vornehmen:

- Auswahl des aktuellen Versionskontroll-Plug-in über die Auswahlliste. Über die Schaltfläche "Erweitert" können Sie die genauen Einstellungen für das ausgewählte Versionskontroll-Plug-in definieren. Diese Einstellungen sind unterschiedlich, je nachdem, welches Versionskontroll-Plug-in Sie verwenden.
- Die Anmelde-ID für das Versionskontrollsystem.
- Spezifische Ein- und Auscheck-Einstellungen.
- Die Schaltfläche "Zurücksetzen" steht zur Verfügung, wenn Sie in einem der Dialogfelder die Option "Wenn Dialogfelder mit "..nicht mehr anzeigen" ausgeblendet waren, klicken Sie auf "Zurücksetzen", damit sie wieder angezeigt werden" aktiviert haben. Daraufhin wird diese "nicht mehr anzeigen"-Meldung wieder angezeigt.

 © 2016-2022 Altova GmbH

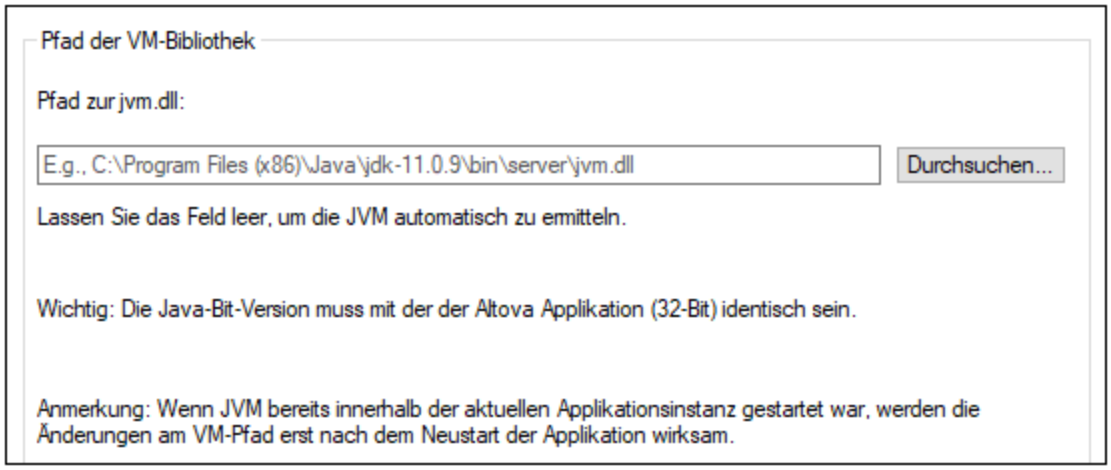


Informationen zu den Einstellungen auf dem Register **Netzwerk-Proxy** finden Sie unter Netzwerk-Proxy-Einstellungen 532. Nähere Informationen zu Java VM-Einstellungen finden Sie unter Java Virtual Machine-Einstellungen 531.

12.6.4.1 Java Virtual Machine-Einstellungen

Im Abschnitt *Java* (*siehe Abbildung unten*) haben Sie die Möglichkeit, den Pfad zu einer Java VM (Virtual Machine) auf Ihrem Dateisystem einzugeben. Beachten Sie, dass dies nicht immer notwendig ist. UModel versucht standardmäßig den Java VM-Pfad automatisch zu ermitteln. Dazu wird zuerst die Windows Registry und anschließend die JAVA_HOME-Umgebungsvariable gelesen. Ein in dieses Dialogfeld eingegebener benutzerdefinierter Pfad hat Vorrang vor allen automatisch ermittelten Java VM-Pfaden.

Wenn Sie eine Java Virtual Machine verwenden, die keinen Installer hat und keine Registry-Einträge erstellt (z.B. OpenJDK von Oracle), müssen Sie eventuell einen benutzerdefinierten Java VM-Pfad angeben. Auch wenn Sie automatisch von UModel ermittelte Java VM-Pfade aus irgendeinem Grund außer Kraft setzen müssen, müssen Sie diesen Pfad eventuell definieren.



Beachten Sie dazu Folgendes:

- Der Java VM-Pfad wird gemeinsam von allen Altova Desktop-Applikationen (nicht aber den Server-Applikationen) verwendet. Wenn Sie den Pfad daher in einer Applikation ändern, gilt dies automatisch auch für alle anderen Altova-Applikationen.
- Der Pfad muss auf die Datei `jvm.dll` im Verzeichnis `\bin\server` oder `\bin\client` (relativ zum Verzeichnis, in dem JDK installiert ist) verweisen.
- Die UModel-Plattform (32-Bit, 64-Bit) muss mit der des JDK identisch sein.
- Nachdem Sie den Java VM-Pfad geändert haben, müssen Sie UModel eventuell neu starten, damit die neuen Einstellungen wirksam werden.

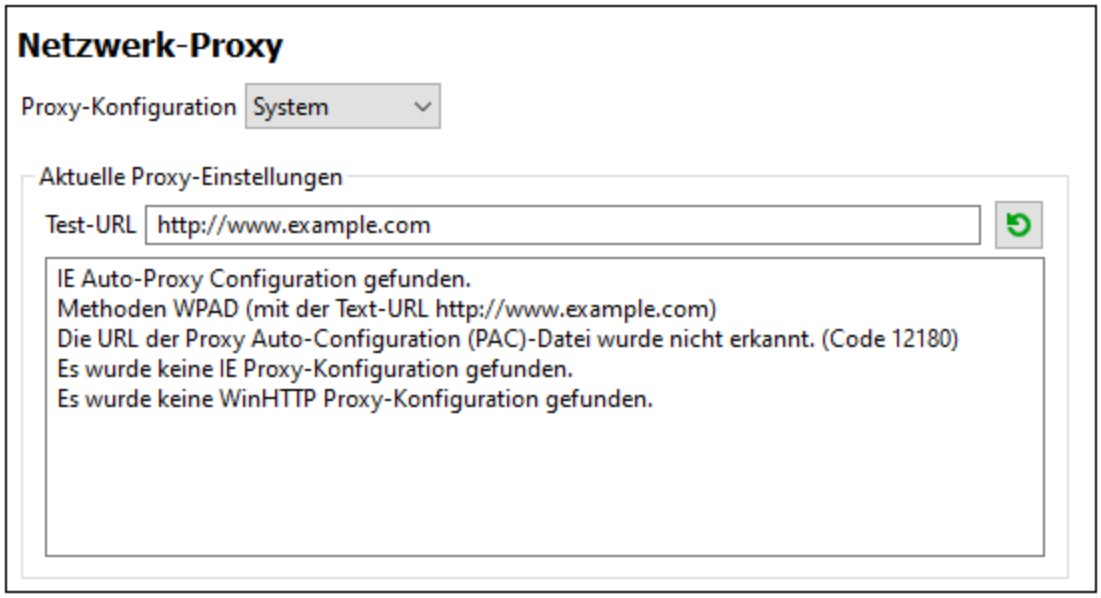
Diese Einstellung hat keine Auswirkung auf die Java-Codegenerierung und den Import von Java-Code. Beachten Sie, dass die für den Import von Java-Binärdateien in UModel verwendeten Java Runtimes separat konfiguriert werden können, siehe Hinzufügen von benutzerdefinierten Java Runtimes 207.

12.6.4.2 Netzwerk-Proxy-Einstellungen

Im Abschnitt *Netzwerk-Proxy* können Sie die benutzerdefinierten Proxy-Einstellungen konfigurieren. Diese Einstellungen beeinflussen, wie die Applikation eine Verbindung mit dem Internet herstellt (z.B. zur XML-Validierung). Standardmäßig werden die Proxy-Einstellungen des Systems verwendet, d.h. die Einstellungen funktionieren, ohne dass der Benutzer etwas daran ändern muss. Falls nötig, können Sie jedoch einen anderen Netzwerk-Proxy-Server definieren. Wählen Sie dazu in der Auswahlliste *Proxy-Konfiguration* entweder die Option *Automatisch* oder *Manuell*, um die Einstellungen entsprechend zu konfigurieren.

Anmerkung: Die Nethwerk-Proxy-Einstellungen werden von allen Altova MissionKit-Applikationen gemeinsam verwendet. Wenn Sie daher die Einstellungen in einer Applikation ändern, wirkt sich dies automatisch auf alle anderen Applikationen aus.

 © 2016-2022 Altova GmbH

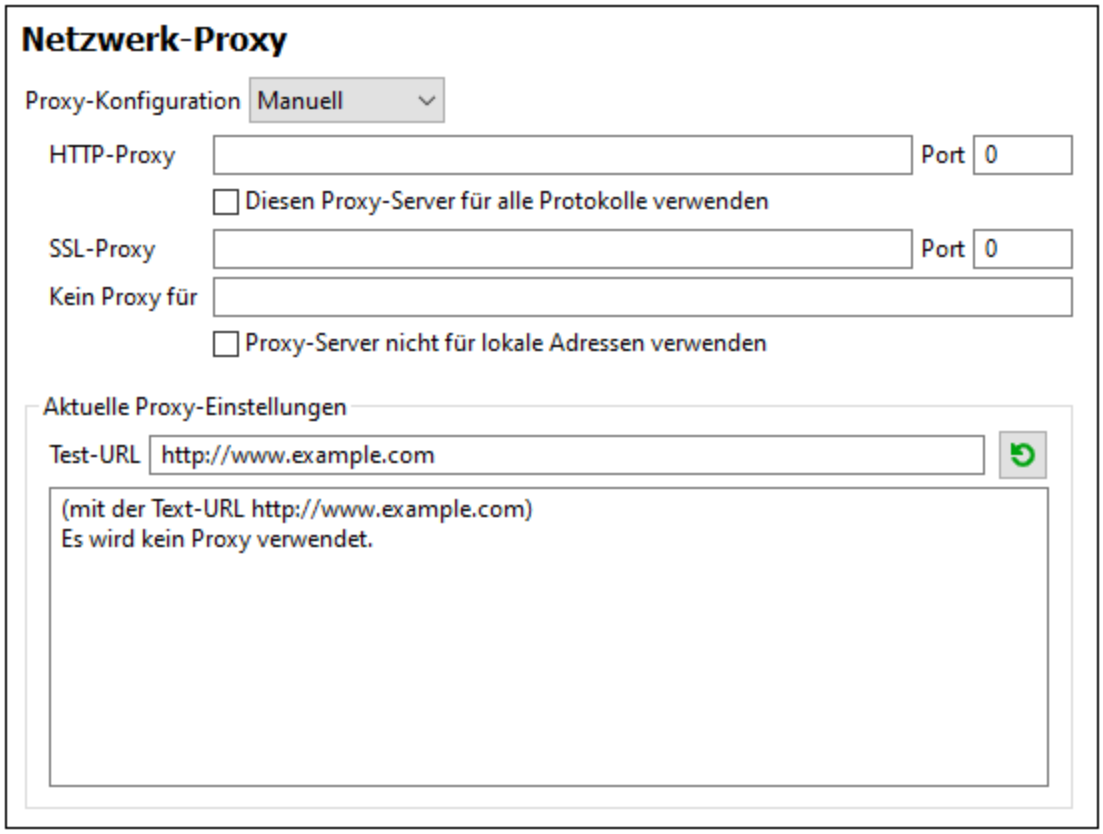


System-Proxy-Einstellungen verwenden
Dadurch werden die über die System-Proxy-Einstellungen konfigurierbaren Internet Explorer (IE)-Einstellungen verwendet. Führt außerdem eine Abfrage der mit **`netsh.exe winhttp`** konfigurierten Einstellungen durch.

Automatische Proxy-Konfiguration
Es stehen die folgenden Optionen zur Verfügung:

- *Einstellungen automatisch ermitteln:* **`Verwendet ein WPAD-Skript`** (http://wpad.LOCALDOMAIN/wpad.dat) über DHCP oder DNS, um die Einrichtung des Proxy-Servers zu konfigurieren.
- *Skript-URL:* Definieren Sie eine HTTP URL zu einem automatischen Proxy-Konfigurationsskript (**`.pac`**), mit dem der Proxy-Server eingerichtet wird.
- *Neu laden:* Setzt die aktuelle automatische Proxy-Konfiguration zurück und lädt sie neu. Dafür ist Windows 8 oder neuer erforderlich. Die Rücksetzung kann bis zu 30 Sekunden dauern.

Manuelle Proxy-Konfiguration
Definieren Sie den vollständig qualifizierten Host-Namen und Port für die Proxy-Server der jeweiligen Protokolle manuell. Im Host-Namen kann ein unterstütztes Schema inkludiert werden (z.B.: **`http://hostname`**). Das Schema muss nicht mit dem entsprechenden Protokoll übereinstimmen, wenn der Proxy-Server das Schema unterstützt.



Es stehen die folgenden Optionen zur Verfügung:

- *HTTP-Proxy:* Verwendet den angegebenen Host-Namen und Port für das HTTP-Protokoll. Wenn *Diesen Proxy-Server für alle Protokolle verwenden* aktiviert ist, wird der angegebene HTTP-Proxy-Server für alle Protokolle verwendet.
- *SSL-Proxy:* Verwendet den angegebenen Host-Namen und Port für das SSL-Protokoll.
- *Kein Proxy für:* eine durch Semikola (`;`) getrennte Liste von voll qualifizierten Host-Namen, Domain-Namen oder IP-Adressen für Hosts, die ohne einen Proxy-Server verwendet werden sollen. IP-Adressen dürfen nicht abgeschnitten werden und IPv6-Adressen müssen innerhalb von eckige Klammern gesetzt werden (z.B.: `[2606:2800:220:1:248:1893:25c8:1946]`). Domain-Namen muss ein Punkt vorangestellt werden (z.B: `.example.com`).
- *Proxy-Server nicht für lokale Adressen verwenden:* Falls dieses Kontrollkästchen aktiviert ist, wird `<local>` zur *Kein Proxy für*-Liste hinzugefügt. Falls diese Option ausgewählt ist, wird für die folgenden Adressen kein Proxy-Server verwendet: (i) `127.0.0.1`, (ii) [::1], (iii) alle Host-Namen, die kein Punktzeichen (`.`) enthalten.

Aktuelle Proxy-Einstellungen
Stellt ein ausführliches Protokoll der Proxy-Ermittlung bereit. Es kann über die Schaltfläche **Aktualisieren** rechts vom Feld *Test-URL* aktualisiert werden (z.B. bei Wechsel zu einer anderen Test-URL oder wenn die Proxy-Einstellungen geändert wurden).

- *Test-URL:* Anhand einer Test-URL kann ermittelt werden, welcher Proxy-Server für diese bestimmte URL verwendet wird. Mit dieser URL erfolgt kein I/O. Dieses Feld darf nicht leer sein, wenn die automatische Proxy-Konfiguration verwendet wird (entweder über *System-Proxy-Einstellungen*

 © 2016-2022 Altova GmbH

verwenden oder *Automatische Proxy-Konfiguration*).

12.7 Fenster

Überlappend

Mit diesem Befehl werden alle offenen Dokumentenfenster so angeordnet, dass sie einander überlappend angezeigt werden.

Horizontal anordnen

Mit diesem Befehl werden alle offenen Dokumentenfenster horizontal nebeneinander angeordnet, sodass alle gleichzeitig sichtbar sind.

Vertikal anordnen

Mit diesem Befehl werden alle offenen Dokumentenfenster vertikal übereinander angeordnet, sodass alle gleichzeitig sichtbar sind.

Symbole anordnen

Damit werden Diagramme, die beliebig und in Symbolform angeordnet sind, entlang des unteren Rands des Diagrammansichtsbereichs positioniert.

Schließen

Schließt das derzeit aktive Diagrammregister.

Alle schließen

Schließt alle derzeit geöffneten Diagrammregister.

Alle bis auf das aktive schließen

Schließt alle Diagrammregister mit Ausnahme des derzeit aktiven.

Vorwärts

Immer wenn Sie den Fokus von einem Diagrammfenster auf ein anderes verlegen oder auf einen Hyperlink klicken, merkt sich UModel dies als Ereignis. Mit diesem Befehl gelangen Sie im Verlauf solcher Ereignisse vorwärts. Der Befehl steht nur dann zur Verfügung, wenn Sie den Befehl **Zurück** bereits verwendet haben (siehe unten).

Zurück

Mit diesem Befehl gelangen Sie zu dem Fenster zurück, das sich zuvor im Fokus befunden hat. Dies kann nützlich sein, wenn Sie gleichzeitig mit mehreren Diagrammfenstern arbeiten oder wenn Sie mittels Hyperlinks navigieren, siehe Erstellen von Hyperlinks zu Elementen 115.

Fensterliste (1,2)

In dieser Liste werden alle derzeit offenen Fenster angezeigt, sodass Sie zwischen ihnen wechseln können. Sie können auch mit Hilfe der Tastaturkürzel **Strg-Tab** oder **Strg F6** zwischen den offenen Fenstern hin- und herwechseln.

© 2016-2022 Altova GmbH

Fenster

Ruft ein Dialogfeld auf, in dem Sie das Layout für mehrere Diagrammfenster gleichzeitig bestimmen bzw. mehrere Fenster gleichzeitig schließen können, siehe auch Diagrammbereich 95.

12.8 Hilfe

Inhaltsverzeichnis

Mit dem Befehl "Inhaltsverzeichnis" wird die Bildschirmhilfe für UModel geöffnet, wobei das Inhaltsverzeichnis auf der linken Seite des Hilfefensters angezeigt wird. Das Inhaltsverzeichnis bietet eine Übersicht über das gesamte Hilfedokument. Wenn Sie auf einen Eintrag im Inhaltsverzeichnis klicken, gelangen Sie zu diesem Thema.

Index

Mit dem Befehl Index wird die Bildschirmhilfe für UModel geöffnet, wobei auf der linken Seite des Hilfefensters der Stichwortindex angezeigt wird. Im Index werden Stichwörter aufgelistet und Sie können durch Doppelklick auf das Stichwort zum jeweiligen Thema navigieren. Wenn ein Stichwort mit mehr als einem Thema verknüpft ist, können Sie das Thema aus einer Liste von Themen auswählen.

Suchen

Mit dem Befehl Suchen wird die Bildschirmhilfe für UModel geöffnet, wobei auf der linken Seite des Hilfefensters das Suchdialogfeld angezeigt wird. Um nach einem Begriff zu suchen, geben Sie ihn in das Eingabefeld ein und drücken Sie die **Eingabetaste** oder klicken Sie auf **Kapitel auflisten**. Das Hilfesystem führt eine Volltextsuche am gesamten Hilfedokument durch und gibt eine Trefferliste zurück. Doppelklicken Sie auf einen Eintrag, um das Thema anzuzeigen.

Software-Aktivierung

Lizenzieren Ihres Produkts
Nachdem Sie Ihre Altova-Software heruntergeladen haben, können Sie sie entweder mit Hilfe eines kostenlosen Evaluierungs-Keycode oder eines käuflich erworbenen permanenten Lizenzkeycode lizenzieren oder aktivieren.

- ***Kostenlose Evaluierungs-Lizenz.*** Wenn Sie die Software zum ersten Mal starten, wird das Dialogfeld **Software-Aktivierung** angezeigt. Es enthält eine Schaltfläche, über die Sie eine kostenlose Evaluierungs-Lizenz anfordern können. Geben Sie in das Dialogfeld Ihren Namen, den Namen Ihrer Firma und Ihre E-Mail-Adresse ein und klicken Sie auf **Anfordern**. Daraufhin wird an die von Ihnen angegebene E-Mail-Adresse eine Lizenz gesendet und sollte in wenigen Minuten eintreffen. Speichern Sie die Lizenzdatei in einem geeigneten Ordner.

  Wenn Sie auf **Anfordern** klicken, wird am unteren Rand des Dialogfelds ein Eingabefeld angezeigt. Über dieses Feld gelangen Sie zum Pfad der Lizenzdatei. Navigieren Sie zur Lizenzdatei oder geben Sie den Pfad dazu ein und klicken Sie auf **OK**. (Sie können im Dialogfeld **Software-Aktivierung** auch auf **Neue Lizenz hochladen** klicken, um das Dialogfeld, in das der Pfad zur Lizenzdatei eingegeben wird, aufzurufen). Die Software wird für einen Zeitraum von 30 Tagen aktiviert.

- ***Permanenter Lizenz-Keycode.*** Über das Dialogfeld **Software-Aktivierung** können Sie einen permanenten Lizenz-Keycode erwerben. Wenn Sie auf diese Schaltfläche klicken, gelangen Sie zum Altova Online Shop, in dem Sie einen permanenten Lizenzschlüssel für Ihr Produkt erwerben können. Ihre Lizenz wird Ihnen in Form einer Lizenzdatei, die Ihre Lizenzdaten enthält, per E-Mail zugesendet.

  Es gibt drei Arten von permanenten Lizenzen: *Einzelplatzlizenzen*, *Parallellizenzen* und *Named User-Lizenzen* (benutzerdefinierte Nutzung). Mit einer Einzelplatzlizenz wird die Software auf

 © 2016-2022 Altova GmbH

einem einzigen Rechner freigeschaltet. Wenn Sie eine Einzelplatzlizenz für *N* Rechner erwerben, gestattet Ihnen die Lizenz, die Software auf bis zu *N* Rechnern zu verwenden. Mit einer Parallellizenz für *N* Parallelbenutzer dürfen *N* Benutzer die Software gleichzeitig ausführen. (Die Software darf auf 10N Rechnern installiert sein.) Mit einer Named User-Lizenz darf ein bestimmter Benutzer die Software auf bis zu 5 verschiedenen Rechnern verwenden. Um Ihre Software zu aktivieren, klicken Sie auf **Neue Lizenz hochladen** und geben Sie im daraufhin angezeigten Dialogfeld den Pfad zur Lizenzdatei ein und klicken Sie auf **OK**.

Anmerkung: Bei Mehrplatzlizenzen wird jeder Benutzer aufgefordert, seinen eigenen Namen einzugeben.

Ihre Lizenz-E-Mail und die verschiedenen Methoden, Ihr Altova-Produkt zu lizenzieren
Die Lizenz-E-Mail, die Sie von Altova erhalten, enthält Ihre Lizenzdatei im Anhang. Die Lizenzdatei hat die Dateierweiterung `.altova_licenses`.

Sie haben folgende Möglichkeiten, Ihr Altova-Produkt zu aktivieren:

- Speichern Sie die Lizenzdatei (`.altova_licenses`) in einem geeigneten Ordner, doppelklicken Sie auf die Lizenzdatei, geben Sie etwaige erforderliche Informationen in das Dialogfeld ein, das daraufhin angezeigt wird und beenden Sie den Vorgang durch Klicken auf **Lizenzschlüssel anwenden**.
- Speichern Sie die Lizenzdatei (`.altova_licenses`) in einem geeigneten Ordner. Wählen Sie in Ihrem Altova-Produkt den Menübefehl **Hilfe | Software-Aktivierung** und klicken Sie anschließend auf **Neue Lizenz hochladen.** Navigieren Sie zur Lizenzdatei oder geben Sie den Pfad dazu ein und klicken Sie auf **OK**.
- Speichern Sie die Lizenzdatei (`.altova_licenses`) in einem geeigneten Ordner und laden Sie diese von dort aus in den Lizenz-Pool Ihres Altova LicenseServer hoch. Sie können die Lizenz anschließend (i) entweder von Ihrem Altova-Produkt über das Dialogfeld "Software-Aktivierung" abrufen (*siehe unten*) oder (ii) dem Produkt die Lizenz von Altova LicenseServer aus zuweisen. *Nähere Informationen zur Lizenzierung über LicenseServer finden Sie weiter unten in diesem Kapitel.*

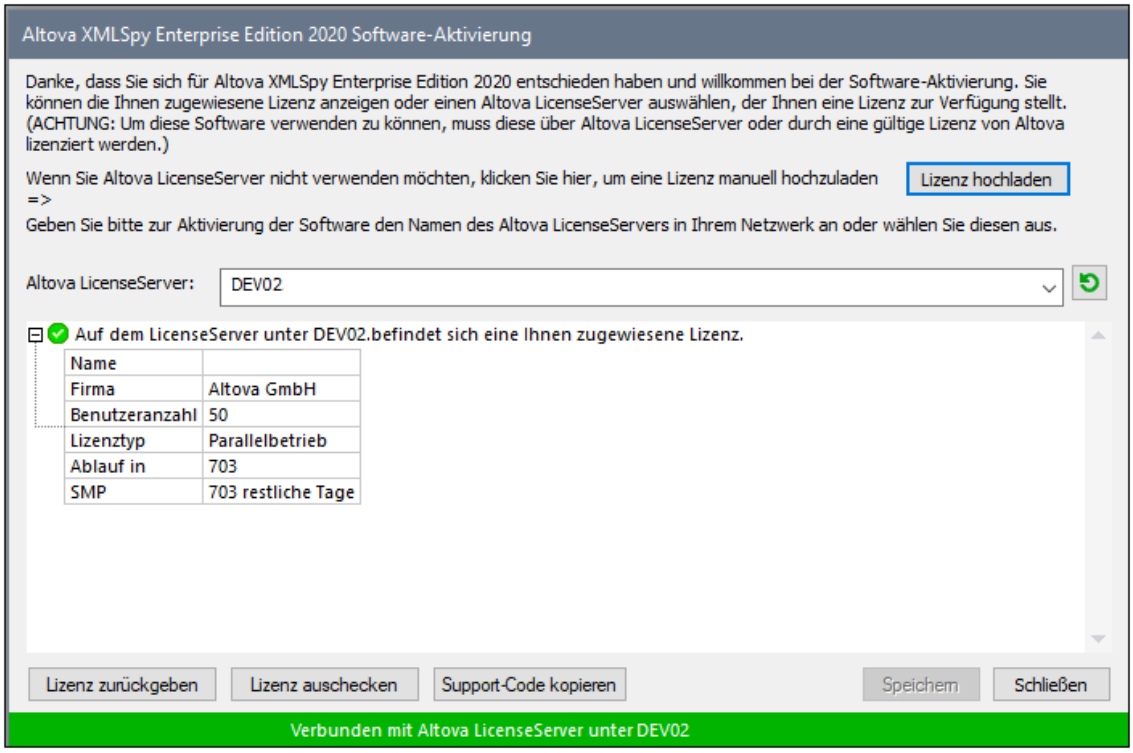
Das Dialogfeld **Software-Aktivierung** (*Abbildung unten*) kann über den Befehl **Hilfe | Software-Aktivierung** aufgerufen werden.

Aktivieren Ihrer Software
Sie können die Software durch Registrieren der Lizenz im Dialogfeld "Software-Aktivierung" oder durch Lizenzierung über Altova LicenseServer (*nähere Informationen siehe unten*) aktivieren.

- *Registrierung der Lizenz im Dialogfeld "Software-Aktivierung".* Klicken Sie im Dialogfeld auf **Neue Lizenz hochladen** und navigieren Sie zur Lizenzdatei. Klicken Sie auf **OK,** um den Pfad zur Lizenzdatei und alle eingegebenen Daten (im Fall einer Mehrplatzlizenz Ihren Namen) zu bestätigen und abschließend auf **Speichern**.

- *Lizenzierung über einen Altova LicenseServer in Ihrem Netzwerk:* Um eine Lizenz über einen Altova LicenseServer in Ihrem Netzwerk abzurufen, (klicken Sie am unteren Rand des Dialogfelds **Software-Aktivierung** auf **Altova LicenseServer verwenden**). Wählen Sie den Rechner aus, auf dem der gewünschte LicenseServer installiert wurde. Beachten Sie, dass die automatische Ermittlung von License Servern durch die Aussendung eines Signals ins LAN erfolgt. Da diese

Aussendung auf ein Subnetz beschränkt ist, muss sich der LicenseServer im selben Subnetz wie der Client-Rechner befinden, damit die Ermittlung von License Servern funktioniert. Falls die automatische Ermittlung nicht funktioniert, geben Sie den Namen des Servers ein. Der Altova LicenseServer muss in seinem Lizenzpool eine Lizenz für Ihre Altova-Produkt haben. Wenn im LicenseServer-Pool eine Lizenz verfügbar ist, wird dies im Dialogfeld **Software-Aktivierung** angezeigt (*siehe Abbildung unten, in der Sie das Dialogfeld in Altova XMLSpy sehen*) und Sie können auf **Speichern** klicken, um die Lizenz abzurufen.



Eine rechnerspezifische Lizenz (Einzelplatzlizenz) kann erst nach Ablauf von sieben Tagen wieder an LicenseServer zurückgegeben werden. Danach können Sie die rechnerspezifische Lizenz durch Klick auf **Lizenz zurückgeben** an den Server zurückgeben, sodass sie von einem anderen Client vom LicenseServer abgerufen werden kann. Ein LicenseServer-Administrator kann die Zuweisung einer abgerufenen Lizenz jedoch über die Web-Benutzeroberfläche von LicenseServer jederzeit aufheben. Beachten Sie, dass eine Rückgabe von Lizenzen nur bei rechnerspezifischen Lizenzen, nicht aber bei Parallellizenzen möglich ist.

Lizenz-Check-Out

Über den Lizenzpool können Sie eine Lizenz für einen Zeitraum von bis zu 30 Tagen auschecken, sodass die Lizenz auf dem lokalen Rechner gespeichert wird. Dadurch können Sie offline arbeiten, was nützlich ist, wenn Sie z.B. in einer Umgebung arbeiten möchten, in der Sie keinen Zugriff auf Ihren Altova LicenseServer haben (z.B. wenn Ihr Altova-Produkt auf einem Laptop installiert ist und Sie gerade unterwegs sind). Solange die Lizenz ausgecheckt ist, zeigt LicenseServer die Lizenz als in Verwendung an. Diese Lizenz kann dann von keinem anderen Rechner verwendet werden. Die Lizenz wird nach Ablauf des Check-Out-Zeitraums automatisch wieder eingecheckt. Alternativ dazu kann eine ausgecheckte Lizenz jederzeit über die Schaltfläche **Einchecken** des Dialogfelds **Software-Aktivierung** wieder eingecheckt werden.

Um eine Lizenz auszuchecken, gehen Sie folgendermaßen vor: (i) Klicken Sie im Dialogfeld

Software-Aktivierung auf **Lizenz auschecken** (*siehe Abbildung oben*); (ii) Wählen Sie im daraufhin angezeigten Dialogfeld **Lizenz-Check-Out** den gewünschten Check-Out-Zeitraum aus und klicken Sie auf **Auschecken**. Daraufhin wird die Lizenz ausgecheckt. Nachdem Sie eine Lizenz ausgecheckt haben, geschehen zwei Dinge: (i) Die Check-Out-Informationen und das Ende des Check-Out-Zeitraums werden im Dialogfeld **Software-Aktivierung** angezeigt; (ii) Die Schaltfläche **Lizenz auschecken** im Dialogfeld ändert sich nun in **Einchecken**. Sie können die Lizenz jederzeit durch Klicken auf **Einchecken** einchecken. Da die Lizenz nach Ablauf des Check-Out-Zeitraums automatisch wieder in den Zustand "Eingecheckt" zurück wechselt, sollte der von Ihnen ausgewählte Zeitraum für das Check-Out den gewünschten Zeitraum, in dem Sie offline arbeiten möchten, entsprechend abdecken.

Wenn eine Lizenz wieder eingecheckt wird, muss diese Lizenz für dieselbe Hauptversion eines Altova-Produkts ausgestellt sein, wie die Lizenz, die ausgecheckt wurde. Stellen Sie daher sicher, dass die Lizenz eingecheckt ist, bevor Sie für Ihr Altova-Produkt ein Upgrade auf die nächste Hauptversion installieren.

Anmerkung: Damit Lizenzen ausgecheckt werden können, muss die Check-Out-Funktion auf dem LicenseServer aktiviert werden. Wenn diese Funktion nicht aktiviert wurde, erhalten Sie eine entsprechende Fehlermeldung, wenn Sie versuchen die Lizenz auszuchecken. Wenden Sie sich in diesem Fall an Ihren LicenseServer-Administrator.

Support-Code kopieren
Klicken Sie auf **Support-Code kopieren,** um Lizenzinformationen in die Zwischenablage zu kopieren. Dies sind die Daten, die Sie bei einer Support-Anfrage über das Online Support-Formular benötigen.

Altova LicenseServer bietet IT-Administratoren einen Echtzeitüberblick über alle Altova-Lizenzen in einem Netzwerk. Dazu werden die Einzelheiten zu jeder Lizenz sowie Client-Zuweisungen und die Verwendung von Lizenzen durch Clients angezeigt. Der Vorteil der Verwendung von LicenseServer liegt in seinen Funktionen zur Verwaltung großer Altova-Lizenzpools. Altova LicenseServer steht kostenlos auf der Altova Website zur Verfügung. Nähere Informationen zu Altova LicenseServer und der Lizenzierung mittels Altova LicenseServer finden Sie in der Dokumentation zu Altova LicenseServer.

Bestellformular

Sobald Sie eine lizenzierte Version des Software-Produkts bestellen möchten, klicken Sie im Dialogfeld **Software-Aktivierung** (*siehe oben*) auf die Schaltfläche **Permanenten Key-Code erwerben...** oder wählen Sie den Befehl **Bestellformular**, um zum sicheren Online-Shop von Altova weitergeleitet zu werden.

Registrierung

Bei Aufruf dieses Befehls wird die Altova-Produktregistrierungsseite auf einem Register Ihres Browsers geöffnet. Durch Registrierung Ihrer Altova-Software stellen Sie sicher, dass Sie immer die neuesten Produktinformationen erhalten.

Auf Updates überprüfen

Überprüft, ob am Altova Server eine neuere Version Ihres Produkts vorhanden ist und zeigt eine entsprechende Meldung an.

Support Center

Der Befehl "Support Center" ist ein Link zum Altova Support Center im Internet. Im Support Center finden Sie Antworten auf häufig gestellt Fragen, Diskussionsforen, in denen Sie Software-Probleme besprechen können und ein Formular, um unsere Mitarbeiter vom technischen Support zu kontaktieren.

Fragen und Antworten im Web

Der Befehl "Fragen und Antworten im Web" ist ein Link zur Altova-Fragen- und Antworten-Datenbank im Internet. Diese Datenbank wird ständig aktualisiert, wenn die Mitarbeiter von Altova von unseren Kunden mit neuen Fragen konfrontiert werden.

Komponenten und Gratistools downloaden

Dieser Befehl ist ein Link zum Komponenten Download Center von Altova im Internet. Von hier können Sie Software-Komponenten verschiedener anderer Anbieter herunterladen, die Sie mit Altova Produkten verwenden können. Dabei handelt es sich um XSLT- und XSL-FO-Prozessoren, Applikationsserverplattformen usw. Die im Komponenten Download Center verfügbare Software ist normalerweise kostenlos.

UModel im Internet

Der Befehl UModel im Internet ist ein Link zur Altova Website im Internet. Hier erfahren Sie mehr über UModel und verwandte Technologien und Produkte auf der Altova Website.

Über UModel

Mit dem Befehl Über UModel wird das Willkommensfenster und die Versionsnummer Ihres Produkts angezeigt. Wenn Sie die 64-Bit-Verson von UModel verwenden, wird dies durch das Suffix (x64) nach dem Applikationsnamen angezeigt. Die 32-Bit-Version hat kein Suffix.

© 2016-2022 Altova GmbH

13 SPL-Referenz

In diesem Abschnitt erhalten Sie eine Übersicht über Spy Programming Language (SPL), die Template-Sprache von Code Generator.

Es wird vorausgesetzt, dass Sie bereits über Programmiererfahrung verfügen und mit Operatoren, Funktionen, Variablen und Klassen sowie den Grundzügen von in SPl häufig verwendeter objektorientierter Programmierung vertraut sind.

Im Ordner \UModelspl finden Sie die von UModel verwendeten Vorlagen**.** Anhand dieser Dateien können Sie Ihre eigenen Vorlagen entwickeln.

Funktionsweise von Code Generator

Die Basis von Code Generator bilden die Vorlagendateien (.spl) und das von UModel bereitgestellte Objektmodell. Die Vorlagendateien enthalten SPL-Anweisungen zum Erstellen von Dateien, Lesen von Informationen aus dem Objektmodell und Ausführen von Berechnungen, die in der Zielprogrammiersprache auch Literalcodefragmente enthalten.

Die Vorlagendatei wird von Code Generator interpretiert und je nach Vorlage wird eine **.java, .cs-**Quellcodedatei oder eine andere Dateiart ausgegeben.

13.1 Grundlegende SPL-Struktur

Eine SPL-Datei enthält Literaltext, der ausgegeben werden soll, der zwischendurch Code Generator-Anweisungen enthält.

Code Generator-Anweisungen sind in eckige Klammern eingeschlossen '**[**' und '**]**'. Innerhalb einer Klammer können mehrere Anweisungen stehen. Zusätzliche Anweisungen müssen durch eine neue Zeile oder einen Doppelpunkt ':' getrennt werden.

Gültige Beispiele sind:

```
[$x = 42
$x = $x + 1]
```

oder

```
[$x = 42: $x = $x + 1]
```

Hinzufügen von Text zu Dateien

Text, der nicht innerhalb von **[** und **]** steht, wird direkt in die Ausgabedatei geschrieben.
Um eckige Klammern als Literale auszugeben, setzen Sie davor als Excape-Zeichen einen umgekehrten Schrägstrich: **\[** und **\]**. Um einen umgekehrten Schrägstrich auszugeben, verwenden Sie \\.

Kommentare

Kommentare innerhalb eines Anweisungsblocks beginnen immer mit einem **'** Zeichen und enden an der nächsten Zeile oder an einem Zeichen, zum Beenden eines Blocks **]**.

© 2016-2022 Altova GmbH

13.2 Variablen

Jede etwas komplexere SPL-Datei enthält Variablen. Einige Variablen sind vom Code Generator vordefiniert. Sie können jederzeit neue Variablen erstellen, indem Sie diesen Werte zuweisen.

Das **$**-Zeichen wird zur **Deklarierung** oder **Verwendung** einer Variablen verwendet. Vor der Variable muss immer ein **$** stehen. Bei Variablennamen muss die **Groß- und Kleinschreibung beachtet werden.**

Variablentypen:

- Integer - wird auch als Boolescher Wert verwendet, wobei 0 false ist und alle anderen Werte true.
- String
- Objekt - wird von UModel bereitgestellt
- Iterator (siehe foreach 556-Anweisung)

Variablentypen werden durch die erste Zuweisung deklariert.

```
[$x = 0]
```

x ist nun ein Integer.

```
[$x = "teststring"]
```

x wird nun als String behandelt.

Strings

String-Konstanten müssen wie im oben gezeigten Beispiel immer innerhalb von doppelten Anführungszeichen stehen. **\n** und **\t** innerhalb von doppelten Anführungszeichen werden als neue Zeile und Tabulator interpretiert, \" ist ein literales doppeltes Anführungszeichen und \\ ist ein umgekehrter Schrägstrich. String-Konstanten können auch mehrere Zeilen einnehmen.

Bei der Verkettung von Strings wird das **&** Zeichen verwendet.

```
[$BasePath = $outputpath & "/" & $JavaPackageDir]
```

Objekte

Objekte stehen für die Informationen, die im UModel-Projekt enthalten sind. Objekte haben **Eigenschaften**, die über den **.** Operator aufgerufen werden können. In SPL können keine neuen Objekte erstellt werden (sie werden durch den Code Generator vordefiniert, aus der Input-Komponente abgeleitet). Sie können Objekte jedoch Variablen zuweisen.

Beispiel:

```
class [=$class.Name]
```

In diesem Beispiel wird das Wort "class" ausgegeben, gefolgt von einem Leerzeichen und dem Wert der Eigenschaft **Name** des Objekts **$class**.

In der folgenden Tabelle sehen Sie die Beziehung zwischen UML-Elementen und ihren SPL-Entsprechungen sowie eine kurze Beschreibung dazu.

Vordefinierte Variablen

UML-Element	SPL-Eigenschaft	Multiplizität	UML-Attribut / Assoziation	UModel-Attribut / Assoziation	Beschreibung
BehavioralFeature	isAbstract		isAbstract:Boolean		
BehavioralFeature	raisedException	*	raisedException:Type		
BehavioralFeature	ownedParameter	*	ownedParameter:Parameter		
BehavioredClassifier	interfaceRealization	*	interfaceRealization:InterfaceRealization		
Class	ownedOperation	*	ownedOperation:Operation		
Class	nestedClassifier	*	nestedClassifier:Classifier		
Classifier	namespace	*		namespace:Package	packages with code language <<namespace>> set
Classifier	rootNamespace	*		project root namespace:String	VB only - root namespace
Classifier	generalization	*	generalization:Generalization		
Classifier	isAbstract		isAbstract:Boolean		
ClassifierTemplateParameter	constrainingClassifier	*	constrainingClassifier		
Comment	body		body:String		
DataType	ownedAttribute	*	ownedAttribute:Property		
DataType	ownedOperation	*	ownedOperation:Operation		
Element	kind			kind:String	
Element	owner	0..1	owner:Element		

© 2016-2022 Altova GmbH

UML-Element	SPL-Eigenschaft	Multiplizität	UML-Attribut / Assoziation	UModel-Attribut / Assoziation	Beschreibung
Element	appliedStereotype	*		appliedStereotype: StereotypeApplication	applied stereotypes
Element	ownedComment	*	ownedComment:Comment		
ElementImport	importedElement	1	importedElement:PackageableElement		
Enumeration	ownedLiteral	*	ownedLiteral:EnumerationLiteral		
Enumeration	nestedClassifier	*		nestedClassifier::Classifier	
Enumeration	interfaceRealization	*		interfaceRealization:Interface	
EnumerationLiteral	ownedAttribute	*		ownedAttribute:Property	
EnumerationLiteral	ownedOperation	*		ownedOperation:Operation	
EnumerationLiteral	nestedClassifier	*		nestedClassifier:Classifier	
Feature	isStatic		isStatic:Boolean		
Generalization	general	1	general:Classifier		
Interface	ownedAttribute	*	ownedAttribute:Property		
Interface	ownedOperation	*	ownedOperation:Operation		
Interface	nestedClassifier	*	nestedClassifier:Classifier		
InterfaceRealization	contract	1	contract:Interface		
MultiplicityElement	lowerValue	0..1	lowerValue:ValueSpecification		
MultiplicityElement	upperValue	0..1	upperValue:ValueSpecification		
NamedElement	name		name:String		
NamedElement	visibility		visibility:VisibilityKind		
NamedElement	isPublic			isPublic:Boolean	visibility <public>

UML-Element	SPL-Eigenschaft	Multiplizität	UML-Attribut / Assoziation	UModel-Attribut / Assoziation	Beschreibung
NamedElement	isProtected			isProtected:Boolean	visibility <protected>
NamedElement	isPrivate			isPrivate:Boolean	visibility <private>
NamedElement	isPackage			isPackage:Boolean	visibility <package>
NamedElement	namespacePrefix			namespacePrefix:String	XSD only - namespace prefix when exists
NamedElement	parseableName			parseableName:String	CSharp, VB only - name with escaped keywords (@)
Namespace	elementImport	*	elementImport:ElementImport		
Operation	ownedReturnParameter	0..1		ownedReturnParameter:Parameter	parameter with direction return set
Operation	type	0..1		type	type of parameter with direction return set
Operation	ownedOperationParameter	*		ownedOperationParameter:Parameter	all parameters excluding parameter with direction return set
Operation	implementedInterface	1		implementedInterface:Interface	CSharp only - the implemented interface
Operation	ownedOperationImplementations	*		implementedOperation:OperationImplementation	VB only - the implemented interfaces/operations
OperationImplementation	implementedOperationOwner	1		implementedOperationOwner:Interface	interface implemented by the operation
OperationImplementation	implementedOperationName			name:String	name of the implemented operation
OperationImplementation	implementedOperationParseableName			parseableName:String	name of the implemented operation with escaped keywords
Package	namespace	*		namespace:Package	packages with code language <<namespace>> set

 © 2016-2022 Altova GmbH

UML-Element	SPL-Eigenschaft	Multiplizität	UML-Attribut / Assoziation	UModel-Attribut / Assoziation	Beschreibung
PackageableElement	owningPackage	0..1		owningPackage	set if owner is a package
PackageableElement	owningNamespacePackage	0..1		owningNamespacePackage:Package	owning package with code language <<namespace>> set
Parameter	direction		direction:ParameterDirectionKind		
Parameter	isIn			isIn:Boolean	direction <in>
Parameter	isInOut			isInOut:Boolean	direction <inout>
Parameter	isOut			isOut:Boolean	direction <out>
Parameter	isReturn			isReturn:Boolean	direction <return>
Parameter	isVarArgList			isVarArgList:Boolean	true if parameter is a variable argument list
Parameter	defaultValue	0..1	defaultValue:ValueSpecification		
Property	defaultValue	0..1	defaultValue:ValueSpecification		
RedefinableElement	isLeaf		isLeaf:Boolean		
Slot	name			name:String	name of the defining feature
Slot	values	*	value:ValueSpecification		
Slot	value			value:String	value of the first value specification
StereotypeApplication	name			name:String	name of applied stereotype
StereotypeApplication	taggedValue	*		taggedValue:Slot	first slot of the instance specification
StructuralFeature	isReadOnly		isReadOnly		
StructuredClassifier	ownedAttribute	*	ownedAttribute:Property		
TemplateBinding	signature	1	signature:TemplateSignature		

UML-Element	SPL-Eigenschaft	Multiplizität	UML-Attribut / Assoziation	UModel-Attribut / Assoziation	Beschreibung
TemplateBinding	parameterSubstitution	*	parameterSubstitution:TemplateParameterSubstitution		
TemplateParameter	paramDefault			paramDefault:String	template parameter default value
TemplateParameter	ownedParameteredElement	1	ownedParameteredElement:ParameterableElement		
TemplateParameterSubstitution	parameterSubstitution			parameterSubstitution:String	Java only - code wildcard handling
TemplateParameterSubstitution	parameterDimensionCount			parameterDimensionCount:Integer	code dimension count of the actual parameter
TemplateParameterSubstitution	actual	1	OwnedActual:ParameterableElement		
TemplateParameterSubstitution	formal	1	formal:TemplateParameter		
TemplateSignature	template	1	template:TemplateableElement		
TemplateSignature	ownedParameter	*	ownedParameter:TemplateParameter		
TemplateableElement	isTemplate			isTemplate:Boolean	true if template signature set
TemplateableElement	ownedTemplateSignature	0..1	ownedTemplateSignature:TemplateSignature		
TemplateableElement	templateBinding	*	templateBinding:TemplateBinding		
Type	typeName	*		typeName:PackageableElement	qualified code type names
TypedElement	type	0..1	type:Type		
TypedElement	postTypeModifier			postTypeModifier:String	postfix code modifiers
ValueSpecification	value			value:String	string value of the value specification

Hinzufügen eines Präfixes zu Attributen einer Klasse bei der Codegenerierung

Eventuell müssen Sie allen neuen Attributen in Ihrem Projekt die Zeichen "m_" voranstellen.

Alle neuen Kodierungselemente werden unter Verwendung der SPL-Vorlagen geschrieben:

© 2016-2022 Altova GmbH

Wenn Sie einen Blick in UModelSPL\C#[Java]**Default\Attribute.spl** werfen, können Sie die Schreibweise des Namens ändern, z.B. können Sie

```
write $Property.name
```

durch

```
write "m_" & $Property.name
```

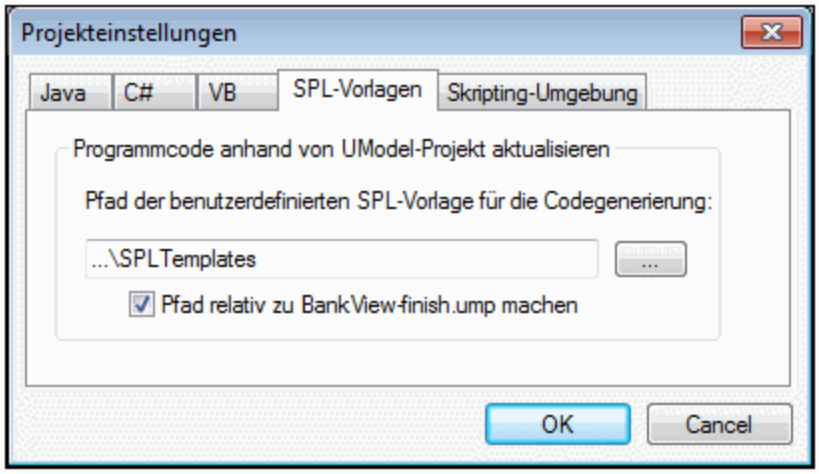
ersetzen.

Es wird unbedingt empfohlen, Ihr Modell sofort nach der Codegenerierung anhand des Codes aktualisieren, um sicherzustellen, dass Code und Modell synchron sind.

Bitte beachten Sie:
Kopieren Sie, wie bereits erwähnt, die SPL-Vorlagen ein Verzeichnis höher hinein (d.h. oberhalb des **Standardverzeichnisses** für **UModelSPL\C#**) bevor Sie sie ändern. Damit stellen Sie sicher, dass sie bei Installation einer neuen Version von UModel nicht überschrieben werden. Stellen Sie bitte sicher, dass Sie im Dialogfeld "Synchronisierungseinstellungen" auf dem Register **Code von Modell** das Kontrollkästchen "Benutzerdefinierte setzt Standardvorlage außer Kraft" aktiviert haben.

SPL-Vorlagen

SPL-Vorlagen können über die Menüoption **Projekt | Projekteinstellungen** (siehe Abbildung unten) pro UModel-Projekt definiert werden. Auch relative Pfade werden unterstützt. Vorlagen, die im angegebenen Verzeichnis nicht gefunden werden, werden im lokalen Standard-Verzeichnis gesucht.



Globale Objekte

$Options	ein Objekt, das globale Optionen enthält:
	generateComments:bool doc-Kommentare generieren (true/false)

$Indent	ein String zum Einrücken von generiertem Code und zur Anzeige der aktuellen Verschachtelungsebene.
$IndentStep	ein String zum Einrücken von generiertem Code und zur Darstellung einer Verschachtelungsebene.
$NamespacePrefix	nur XSD – das Target Namespace Präfix, falls vorhanden

Routinen zur Behandlung von Strings

```
integer Compare(s)
```

Der Rückgabewert gibt die lexikographische Beziehung des String zu s an (unter Berücksichtigung der Groß- und Kleinschreibung):

<0:	der String ist kleiner als s
0:	der String ist identisch mit s
>0:	der String ist größer als s

```
integer CompareNoCase(s)
```

Der Rückgabewert gibt die lexikographische Beziehung des String zu s an (ohne Berücksichtigung der Groß- und Kleinschreibung):

<0:	der String ist kleiner als s
0:	der String ist identisch mit s
>0:	der String ist größer als s

```
integer Find(s)
```

Durchsucht den String nach der ersten Instanz eines Substring s.
Gibt den nullbasierten Index des ersten Zeichens von s oder -1 zurück, wenn s nicht gefunden wird.

```
string Left(n)
```

Gibt die ersten n Zeichen des String zurück.

```
integer Length()
```

Gibt die Länge des String zurück.

© 2016-2022 Altova GmbH

```
string MakeUpper()
```

Gibt einen String zurück, der in Großbuchstaben konvertiert wurde.

```
string MakeUpper(n)
```

Gibt einen String zurück, dessen erste n Zeichen in Großbuchstaben konvertiert wurden.

```
string MakeLower()
```

Gibt einen String zurück, der in Kleinbuchstaben konvertiert wurde.

```
string MakeLower(n))
```

Gibt einen String zurück, dessen erste n Zeichen in Kleinbuchstaben konvertiert wurden.

```
string Mid(n)
```

Gibt einen String zurück, der mit der nullbasierten Indexposition n beginnt

```
string Mid(n,m)
```

Gibt einen String zurück, der mit der nullbasierten Indexposition n beginnt und die Länge m hat

```
string RemoveLeft(s)
```

Gibt einen String zurück, der den Substring s ausnimmt, wenn Left(s.Length()) dem Substring s entspricht.

```
string RemoveLeftNoCase(s)
```

Gibt einen String zurück, der den Substring s ausnimmt, wenn Left(s.Length()) dem Substring s entspricht (Groß- und Kleinschreibung wird ignoriert).

```
string RemoveRight(s)
```

Gibt einen String zurück, der den Substring s ausnimmt, wenn Right(s.Length() dem Substring s entspricht.

```
string RemoveRightNoCase(s)
```

Gibt einen String zurück, der den Substring s ausnimmt, wenn Right(s.Length() dem Substring s entspricht (Groß- und Kleinschreibung wird ignoriert).

```
string Repeat(s,n)
```

Gibt einen String zurück, der den Substring s n mal enthält.

```
string Right(n)
```

Gibt die letzten n Zeichen des String zurück.

13.3 Operatoren

Operatoren in SPL funktionieren wie in den meisten anderen Programmiersprachen.

Liste von SPL-Operatoren absteigend nach Vorrangigkeit gereiht:

.	Objekteigenschaft aufrufen
( )	Gruppierung eines Ausdrucks
true	Boolesche Konstante "true"
false	Boolesche Konstante "false"
&	String-Verkettung
-	Zeichen für negative Zahl
not	Logische Negation
*	Multiplizieren
/	Dividieren
%	Modulo
+	Addieren
-	Subtrahieren
<=	Kleiner oder gleich
<	Kleiner als
>=	Größer oder gleich
>	Größer als
=	Gleich
<>	Nicht gleich
and	Logische Konjunktion (mit short circuit-Auswertung)
or	Logische Disjunktion (mit short circuit-Auswertung)
=	Zuweisung

© 2016-2022 Altova GmbH

13.4 Bedingungen

SPL erlaubt die Verwendung von Standard-"if"-Anweisungen. Die Syntax lautet wie folgt:

```
if condition
        statements
else
        statements
endif
```

oder ohne else:

```
if condition
        statements
endif
```

Beachten Sie bitte, dass für die Bedingung keine runden Klammern verwendet werden!

Wenn in allen anderen Programmiersprachen werden Bedingungen mit logischen und Vergleichsoperatoren[554] konstruiert.

Beispiel:

```
[if $namespace.ContainsPublicClasses and $namespace.Prefix <> ""]
        whatever you want ['inserts whatever you want, in the resulting file]
[endif]
```

Switch

SPL enthält auch eine Multiple Choice-Anweisung.

Syntax:

```
switch $variable
        case X:
                statements
        case Y:
        case Z:
                statements
        default:
                statements
endswitch
```

Die Case-Bezeichnungen müssen Konstanten oder Variablen sein.

Die switch-Anweisung in SPL fällt nicht durch die Cases (wie in C), daher wird auch keine "break"-Anweisung benötigt.

13.5 Collections und foreach

Collections und Iteratoren

Eine Collection enthält mehrere Objekte - wie ein gewöhnliches Array. Iteratoren lösen das Problem des Speicherns und Inkrementierens von Array Indizes beim Aufrufen von Objekten.

Syntax:

```
foreach iterator in collection
       statements
next
```

Beispiel:

```
[foreach $class in $classes
       if not $class.IsInternal
              ]      class [=$class.Name];
[      endif
next]
```

Beispiel 2:

```
[foreach $i in 1 To 3
      Write "// Step " & $i & "\n"
      ' Do some work
next]
```

Foreach iteriert durch alle Datenelemente in $classes und führt für die Anweisung den Code der auf die Anweisung folgt, bis zur **nächsten** Anweisung aus.

In jeder Iteration wird **$class** dem nächsten Klassenobjekt zugewiesen. Sie arbeiten einfach mit dem Klassenobjekt, anstelle classes[i]->class->Name(), zu verwenden, wie das in C++ der Fall wäre.

Alle Collection-Iteratoren haben die folgenden zusätzlichen Eigenschaften:

Index	der aktuelle Index beginnend mit 0
IsFirst	true, wenn das aktuelle Objekt das erste der Collection ist (Index ist 0)
IsLast	true, wenn das aktuelle Objekt das letzte einer Collection ist

Beispiel:

```
[foreach $enum in $facet.Enumeration
       if not $enum.IsFirst
```

 © 2016-2022 Altova GmbH

```
                ], [
        endif
        ]"[=$enum.Value]"[
next]
```

Routinen zur Bearbeitung von Collections

collection **SortByName**(bAscending)

gibt eine Collection zurück, deren Elemente in aufsteigender oder absteigender Reihenfolge (unter Berücksichtigung der Groß- und Kleinschreibung) nach Namen sortiert sind .

collection **SortByNameNoCase**(bAscending)

gibt eine Collection zurück, deren Elemente in aufsteigender oder absteigender Reihenfolge (ohne Berücksichtigung der Groß- und Kleinschreibung) nach Namen sortiert sind .

Beispiel:

```
$SortedNestedClassifier = $Class.nestedClassifier.SortByNameNoCase( true )
```

collection **SortByKind**(bAscending)

gibt eine Collection zurück, deren Elemente in aufsteigender oder absteigender Reihenfolge nach kind-Namen (z.B. “Class”, “Interface”,…) sortiert sind.

collection **SortByKindAndName**(bAscendingKind, bAscendingName)

gibt eine Collection zurück, deren Elemente in aufsteigender oder absteigender Reihenfolge nach "kind" (z.B. “Class”, “Interface”,…) sortiert sind, wobei berücksichtigt wird, ob die "kinds" sortiert in aufsteigender oder absteigender Reihenfolge (unter Berücksichtigung der Groß- und Kleinschreibung) im Namen identisch sind.

collection **SortByKindAndNameNoCase**(bAscending)

gibt eine Collection zurück, deren Elemente in aufsteigender oder absteigender Reihenfolge nach "kind" (z.B. “Class”, “Interface”,…) sortiert sind, wobei berücksichtigt wird, ob die "kinds" sortiert in aufsteigender oder absteigender Reihenfolge (ohne Berücksichtigung der Groß- und Kleinschreibung) im Namen identisch sind.

13.6 Subroutinen

Code Generator unterstützt nun Subroutinen in der Form von Prozeduren oder Funktionen.

Funktionen:

- Übergabe von Werten nach Wert und nach Referenz
- Lokale/globale Parameter (lokal innerhalb von Subroutinen)
- Lokale Variablen
- Rekursiver Aufruf (Subroutinen können sich selbst aufrufen)

13.6.1 Deklaration einer Subroutine

Subroutinen

Syntax-Beispiel:

```
Sub SimpleSub()
       ... lines of code
 EndSub
```

- **Sub** ist das Schlüsselwort, das die Prozedur notiert.
- **SimpleSub** ist der Name, der der Subroutine zugewiesen wurde.
- Runde **Klammern** können eine Parameterliste enthalten.
- Der Code-Block einer Subroutine beginnt direkt nach der geschlossenen Klammer nach dem Parameter.
- **EndSub** notiert das Ende des Code-Blocks.

Bitte beachten Sie:

Eine rekursive oder kaskadierende Subroutine-**Deklaration** ist nicht zulässig, d.h. eine Subroutine darf keine weiteren Subroutinen enthalten.

Parameter

Parameter können auch durch Prozeduren unter Verwendung der folgenden Syntax übergeben werden:

- Alle Parameter müssen Variablen sein
- Variablen muss das **$**-Zeichen vorangestellt werden
- Lokale Variablen werden in einer Subroutine definiert
- Globale Variablen werden explizit außerhalb von Subroutinen deklariert
- Mehrere Parameter werden innerhalb runder Klammern durch Kommas "**,**" voneinander getrennt
- Parameter können Werte übergeben

Parameter - Übergabe von Werten

Parameter können auf zwei Arten übergeben werden: nach Wert oder nach Referenz, wobei die Schlüsselwörter ByVal bzw. ByRef verwendet werden.

© 2016-2022 Altova GmbH

Syntax:

```
' define sub CompleteSub()
[Sub CompleteSub( $param, ByVal $paramByValue, ByRef $paramByRef )
] ...
```

- **ByVal** definiert, dass der Parameter nach Wert übergeben wird. Beachten Sie, dass die meisten Objekte nur nach Referenz übergeben werden können.
- **ByRef** definiert, dass der Parameter nach Referenz übergeben wird. Dies ist die Standardeinstellung, wenn weder ByVal noch ByRef definiert ist.

Funktionsrückgabewerte

Für die Rückgabe eines Werts von einer Subroutine verwenden Sie ein **Return**-Statement. Eine solche Funktion kann von innerhalb eines Ausdrucks aufgerufen werden.

Beispiel:

```
' define a function
[Sub MakeQualifiedName( ByVal $namespacePrefix, ByVal $localName )
if $namespacePrefix = ""
  return $localName
else
  return $namespacePrefix & ":" & $localName
endif
EndSub
]
```

13.6.2 Subroutinenaufruf

Verwenden Sie zum Aufrufen einer Subroutine **call**, gefolgt vom Namen der Prozedur und ggf. den Parametern.

```
Call SimpleSub()
```

oder

```
Call CompleteSub( "FirstParameter", $ParamByValue, $ParamByRef )
```

Funktionsaufruf

Um eine Funktion (jede Subroutine, die ein **Return**-Statement enthält) aufzurufen, verwenden Sie einfach deren Namen in einem Ausdruck. Verwenden Sie zum Aufrufen von Funktionen nicht das **Call**-Statement.
Beispiel:

```
$QName = MakeQualifiedName($namespace, "entry")
```

14 Lizenzinformationen

Dieser Anhang enthält die folgenden Informationen:

- Informationen über den Vertrieb dieses Software-Produkts
- Informationen zur Software-Aktivierung und Lizenzüberwachung
- die Lizenzvereinbarung zu diesem Software-Produkt

Lesen Sie die Informationen bitte sorgfältig - sie sind rechtlich bindend, da Sie sich bei der Installation dieses Software-Produkts damit einverstanden erklärt haben.

Den Inhalt aller Altova-Lizenzenvereinbarungen finden Sie auf der Altova Website unter Rechtliches.

© 2016-2022 Altova GmbH

14.1 Electronic Software Distribution

Dieses Produkt ist über EDS (Electronic Software Distribution), also auf elektronischem Weg erhältlich, eine Methode, die die folgenden einzigartigen Vorteile bietet:

- Sie können die Software kostenlos 30 Tage lang testen, bevor Sie sich zu einem Kauf entscheiden. (*Anmerkung: Die Lizenz für MobileTogether Designer ist kostenlos*.)
- Wenn Sie sich entschieden haben, die Software zu kaufen, können Sie Ihre Bestellung online auf der Altova Website tätigen. Sie erhalten dann innerhalb weniger Minuten ein vollständig lizenziertes Produkt.
- Sie erhalten immer die neueste Version unserer Software
- Die Software enthält ein umfassendes Hilfesystem, das Sie von der Benutzeroberfläche der Appikation aus aufrufen können. Die neueste Version des Benutzerhandbuchs steht auf unserer Website www.altova.com (i) im HTML-Format zum Aufrufen online und (ii) im PDF-Format zum Download und Ausdrucken zur Verfügung.

30-Tage-Evaluierungszeitraum

Nachdem Sie dieses Software-Produkt heruntergeladen haben, können Sie es 30 Tage lang kostenlos testen. Während dieses Zeitraums werden Sie nach etwa 20 Tagen in regelmäßigen Abständen daran erinnert, dass die Software noch nicht lizenziert wurde. Diese Erinnerungsmeldung wird allerdings nur einmal, nämlich bei jedem Start des Programms, angezeigt. Wenn Sie das Programm nach Ablauf des 30-tägigen Evaluierungszeitraums weiterhin verwenden möchten, müssen Sie eine Produktlizenz erwerben, die Sie in Form einer Lizenzdatei mit einem Keycode erhalten. Laden Sie die Lizenzdatei über das Dialogfeld "Software-Aktivierung" Ihres Produkts hoch, um das Produkt freizuschalten.

Sie können Ihre Produktlizenz über https://shop.altova.com/ erwerben.

Weitergabe der Software an andere Mitarbeiter in Ihrem Unternehmen zu Testzwecken

Wenn Sie die Evaluierungsversion der Software auch anderen Personen in Ihrem Unternehmen über das Netzwerk zur Verfügung stellen möchten oder wenn Sie sie auf einem PC installieren möchten, der nicht mit dem Internet verbunden ist, dürfen Sie nur das Installationsprogramm weitergeben, vorausgesetzt es wurde nicht modifiziert. Jeder, der das von Ihnen zur Verfügung gestellte Installationsprogramm aufruft, muss einen eigenen Evaluierungs-Keycode für 30 Tage anfordern. Nach Ablauf des Testzeitraums, muss eine Lizenz erworben werden, damit das Produkt weiter verwendet werden kann.

14.2 Software-Aktivierung und Lizenzüberwachung

Im Rahmen der Aktivierung der Software durch Altova, verwendet die Software unter Umständen Ihr internes Netzwerk und Ihre Internetverbindung, um die Lizenzdaten während der Installation, Registrierung, der Verwendung oder der Aktualisierung an einen von Altova betriebenen Lizenzserver zu übertragen und die Authentizität der Lizenzdaten zu überprüfen, damit Altova-Software nicht ohne Lizenz oder auf unzulässige Art und Weise verwendet werden kann und um den Kundenservice gleichzeitig zu verbessern. Bei der Aktivierung werden zwischen Ihrem Computer und dem Altova-Lizenzserver für die Lizenzierung erforderliche Daten wie Informationen über Betriebssystem, IP-Adresse, Datum/Uhrzeit, Software-Version und Computername sowie andere Informationen ausgetauscht.

Ihr Altova-Produkt verfügt über ein integriertes Lizenzüberwachungsmodul, das ebenfalls dazu beiträgt, unbeabsichtigte Verletzungen der Lizenzvereinbarung zu vermeiden. Ihr Produkt kann entweder mit einer Einzelplatzlizenz oder einer Mehrfachlizenz erworben werden. Je nach Lizenz stellt das Lizenzüberwachungsmodul sicher, dass nicht mehr als die lizenzierte Anzahl an Benutzern die Applikation gleichzeitig verwendet.

Bei dieser Lizenzüberwachungsmethode wird Ihr LAN-Netzwerk verwendet, um die Kommunikation zwischen Instanzen der Applikation, die auf verschiedenen Computern laufen, zu überwachen.

Einzelplatzlizenz

Beim Start der Applikation wird im Rahmen der Lizenzüberprüfung ein kurzes Broadcast-Datagramm abgesendet, um andere Instanzen des Produkts, die auf anderen Computern im selben Netzwerk laufen, zu finden. Wenn keine Antwort einlangt, wird ein Port geöffnet, der Informationen von anderen Instanzen der Applikation empfangen kann.

Mehrplatzlizenz

Wenn Sie im selben LAN mehrere Instanzen der Applikation verwenden, kommunizieren diese beim Start kurz miteinander, um Keycode-Informationen auszutauschen, damit Sie sicher sein können, dass nicht mehr als die lizenzierte Anzahl an Lizenzen gleichzeitig in Verwendung ist. Dieselbe Lizenzüberwachungstechnologie wird auch bei Unix und vielen anderen Datenbankentwicklungstools verwendet. Sie gestattet Benutzern den Erwerb von Parallellizenzen für mehrere Benutzer zu vernünftigen Preisen.

Wir sind außerdem bestrebt, nur wenige, kleine Netzwerkpakete zu versenden, um Ihr Netzwerk nicht zu überlasten. Die von Ihrem Altova Produkt verwendeten TCP/IP Ports (2799) sind offiziell bei IANA registriert, (*nähere Informationen siehe IANA Service Name Registry*) und unser Lizenzüberwachungsmodul basiert auf einer bewährten und erprobten Technologie.

Wenn Sie eine Firewall verwenden, werden Sie unter Umständen feststellen, dass die Computer, auf denen Altova-Produkte laufen, über Port 2799 miteinander kommunizieren. Sie können diesen Netzwerkverkehr zwischen verschiedenen Gruppen in Ihrem Unternehmen natürlich blockieren, solange Sie mit anderen Mitteln sicherstellen können, dass Ihre Lizenzvereinbarung eingehalten wird.

Anmerkung zu Zertifikaten

Ihre Altova Applikation kontaktiert den Altova Lizenzierungsserver über HTTPS (`link.altova.com`). Für diese Kommunikation verwendet Altova ein registriertes SSL-Zertifikat. Wenn dieses Zertifikat ersetzt wird (z.B. von Ihrer IT-Abteilung oder einer externen Agentur), werden Sie von Ihrer Altova Applikation gewarnt, dass die Verbindung nicht sicher ist. Sie könnten Ihre Altova Applikation mit dem Ersetzungszertifikat starten. Dies

 © 2016-2022 Altova GmbH

würde jedoch auf Ihr eigenes Risiko geschehen. Wenn Sie eine Warnung sehen, dass die *Verbindung nicht sicher* ist, überprüfen Sie den Ursprung des Zertifikats und wenden Sie sich an Ihr IT-Team (die in der Lage sein sollten, zu entscheiden, ob das Abfangen und die Ersetzung des Altova-Zertifkats fortgesetzt werden soll).

Wenn Ihr Unternehmen sein eigenes Zertifikat verwenden muss (z.B. um die Kommunikation zu und von Client-Rechnern zu überwachen), empfehlen wir Ihnen, Altova LicenseServer, die kostenlose Lizenzverwaltungssoftware von Altova in Ihrem Netzwerk zu installieren. Client-Rechner verwenden mit dieser Konfiguration weiterhin die Zertifikate Ihres Unternehmens, während der Altova LicenseServer für die Kommunikation mit Altova das Altova-Zertifikat das Altova-Zertifikat verwenden kann.

14.3 Altova Endbenutzer-Lizenzvereinbarung

- Die Altova-Endbenutzer-Lizenzvereinbarung kann unter https://www.altova.com/de/legal/eula eingesehen werden.
- Die Altova-Datenschutzbestimmungen finden Sie unter https://www.altova.com/de/privacy.

© 2016-2022 Altova GmbH

Index

.

.NET 5,
als UModel-Profil, 164
Typen aus Binärdateien importieren, 97
Unterstützung, 11
.NET Core, 11
Assemblys importieren, 211
.NET Framework,
Assemblys importieren, 211
UModel Projekt inkludieren, 164

A

Abgeleitete,
Klasse, 34
Abhängigkeit,
inkludieren, 16
Verwendung, 48
Abhängigkeiten,
anzeigen, 88
Abstrakte,
Klasse, 25
Akteur,
benutzerdefiniert, 16
Aktiviere Versionskontrolle, 464
Aktivierungsfeld,
Ausführungsspezifikation, 359
Aktivität, 320
Diagramm zu Transition hinzufügen, 320
Diagrammelemente, 308
Operation hinzufügen, 320
Verzweigung / Merge erstellen, 306
zum Zustand hinzufügen, 320
Aktivitätsdiagramm, 302
Elemente einfügen, 303
Symbole, 488
Aktualisieren,
Projektdatei, 221
Aktuellste Version holen, 465
Alle,
erweitern/reduzieren, 394
Anbieter wechseln,
Versionskontrolle, 480
Anpassen, 515
Befehle der Symbolleiste/des Menüs, 516
Ansicht, 514
zu mehreren Instanzen eines Elements, 394
Antwort,
Nachricht automatisch generieren, 365
Anwendungsfall,
hinzufügen, 16
Anzeigen,
Eigenschaft als Assziation, 288
Regionsnamen ausblenden, 328
Arbeitsablauf,
Projekt, 152
Arbeitsverzeichnis,
Versionskontrolle, 461
Artefakt,
Manifest, 54
zu Node hinzufügen, 54
Assoziation,
aggregrieren/zusammensetzen, 25
als Beziehung, 133
anzeigen, 137
Eigenschaften ändern, 137
erstellen, 133, 137
Objektverknüpfungen, 40
reflexive Assoziationen, 137
typisierte Eigenschaft anzeigen, 288
Use Case, 16
zwischen Klassen, 25
Assoziationen,
anzeigen, 88
Assoziationsqualifier,
erstellen, 137
Attribut,
Autokomplettierungsfenster, 525
ein-/ausblenden, 394
Farbe, 400
Aufruf,
Nachricht, 365
Aufrufoperation,
einfügen, 303
Aufrufverhalten,
einfügen, 303
Ausblenden,

Ausblenden,
einblenden - Slot, 394
Auschecken, 468
Auschecken rückgängig, 470
Ausführungsspezifikation,
Lebenslinie, 359
Ausgliedern,
aus Versionskontrolle, 474
Auslösendes Ereignis,
Zeitverlaufsdiagramm, 390
Ausnahme,
Ausnahmeereignis hinzufügen, 394
Ausnahmeereignis,
hinzufügen, 394
Ausrichten,
beim Ziehen an Hilfslinien ausrichten, 525
Elemente beim Ziehen, 16
Ausrichtungshilfslinien, 16
Außer Kraft setzen,
SPL-Standardvorlagen, 221
Austrittspunkt,
zu Unterautomat hinzufügen, 328
Autokomplettierung,
Fenster zum Bearbeiten von Klassen, 525
Funktion, 25
Autokomplettierung von Datentypen,
auslösen, 130
deaktiviere, 130
Automatisch generieren,
Antwortnachricht, 365

B

Ball and socket,
Interface notation, 394
Basis,
Klasse, 34
Basisklasse,
überschreiben, 394
Batch-Modus,
Projekte erstellen, 102
Projekte laden, 102
Projekte speichern, 102
Bearbeiten (Menü),
Befehle, 507
Bearbeitungen rückgängig machen, 470
Befehle,
zur Symbolleiste/zum Menü hinzufügen, 516
Befehlszeile,
Binärtypen importieren, 97
Code und Modell synchronisieren, 97
Programmcode generieren, 97
Projekte erstellen, 102
Projekte laden, 102
Projekte speichern, 102
Quellcode importieren, 97
Referenz, 97
Benutzerdefinierte,
SPL-Vorlagen, 221
Benutzerdefinierter,
Akteur, 16
Bereich,
einzelne/mehrere erweitern, 394
Beziehungen,
Abhängigkeit, 133
Aggregation, 133
anzeigen, 136
Assoziation, 107, 133
Generalisierung, 107, 133
Komposition, 133
Realisierung, 133
Stil ändern, 134
Bilder,
als Elementhintergrund verwenden, 119
Binärdateien,
ins Modell importieren, 206

C

C#,
Attribute importieren, 208
automatisch implementierte Eigenschaften, 177
Binärdateien importieren, 206, 211
Code generieren, 170, 177
Codegenerierungsoptionen, 176
Optionen für den Code-Import, 196
Quellcode importieren, 194
C++,
Quellcode importieren, 194
Call-Nachricht,
gehe zu Operation, 365
Classifier,

© 2016-2022 Altova GmbH

Classifier,
einschränken, 286
neu, 222
umbenennen, 222
Code, 224
Code zu Sequenzdiagramm hinzufügen, 382
Java-Code und Klassendateinamen, 224
Refactoring, 224
Sequenzdiagramm generieren, 371
SPL, 543
Standard, 525
Synchronisierung, 221
von Sequenzdiagramm generieren, 378
Code Engineering,
Assoziatonen auflösen, 140
Code zu Modell, 69
Fehler, 92
Informationsmeldungen, 92
Komponentenrealisierungen generieren, 222
Modell zu Code, 59
Projektdatei in neuen Ordner verschieben, 152
Tutorial-Beispiele, 14
Warnungen, 92
Collection-Assoziation,
erstellen, 140
in Collection-Vorlagen auflösen, 140
Voraussetzungen, 140
Combined Fragment, 361
Copyright-Informationen, 560
CR/LF,
für ump-Datei beim Speichern, 152

D

Datei,
Projektdateien zusammenführen, 280
ump, 152
von URL öffnen, 505
Datei (Menü),
Befehle, 505
Datei abrufen,
Versionskontrolle, 465
Datei holen,
Versionskontrolle, 465
Deaktiviere Versionskontrolle, 464
Deployment,
Diagramm, 54
Symbole, 493
Deployment-Diagramm, 413
Diagram,
- Use Case-Diagramm, 347
Diagramm, 414, 525
- Aktivitätsdiagramm, 302
- Interaktionsübersichtsdiagramm, 351
- Klassendiagramm, 394
Aktivität zu Transition hinzufügen, 320
als png speichern, 505
Code von Sequenzdiagramm generieren, 378
Code zu Sequenzdiagramm hinzufügen, 382
Deployment-Diagramm, 413
Elemente aus inkludierten Dateien ignoieren, 525
Elemente einfügen, 107
Kommunikationsdiagramm, 347
Komponentendiagramm, 413
Kompositionsstrukturdiagramm, 410
mehrere Klasseninstanzen, 394
nicht verwendete Elemente suchen, 113
Objektdiagramm, 414
offene Diagramme mit Projekt speichern, 525
Paketabhängigkeiten erstellen, 414
Paktediagramm, 414
Schnellbildlauf, 90
Sequenzdiagramm, 356
Stile, 87
Symbole, 487
Symbolreferenz, 80
Übersicht anzeigen, 90
XML-Schema, 435
Zeitverlaufsdiagramm, 385
zu Favoriten hinzufügen, 85
Zusätzliches - XML-Schema, 435
Zustandsdiagramm, 319
Diagramme, 301
an Fenstergröße anpassen, 132
aus Projekt löschen, 126
Aussehen ändern, 127
erstellen, 94, 122
generieren, 123
Größe ändern, 127
in einem Projekt anzeigen, 84
öffnen, 125
Strukturdiagramme, 394
über das Fenster "Hierarchie" generieren, 88
vergrößern/verkleinern, 132

© 2016-2022 Altova GmbH

Diagramme, 301
Verhaltensdiagramme, 302
Diagramm-Struktur (Fenster), 84
Diagrammtyp,
identifizieren, 94
Dokumentation,
anhand von UML-Projekt generieren, 290
aus Quellcode importieren, 118
Quellcode mit Dokumenation generieren, 118
zu Elementen hinzufügen, 118
Dokumentation (Fenster), 91
Download Versionskontroll-Projekt, 461
Druckvorschau,
Optionen, 505
Durchsuchen,
Diagramme, 111
Elemente, 111
Text, 111

E

Eigenschaft,
als Lebenslinie typisiert, 359
Farbe, 400
typisierte - anzeigen, 288
wiederverwenden, 34
Eigenschaften,
hinzufügen, 25
Versionskontrolle, 479
Eigenschaften (Fenster),
benutzerdefinierte Eigenschaften hinzufügen, 86
Eigenschaftswerte,
als Enumerationen, 424, 427
anzeigen oder ausblenden, 149
Beispiel, 427
Beispiele, 146
Definition, 146
erstellen, 147, 424
Ein-/ausblenden,
Attribute, Operationen, 394
Einblenden,
ausblenden - Slot, 394
Einchecken, 469
Einfügen, 303
Aktion (Aufrufoperation), 303
Aktion (Aufrufverhalten), 303
einfachen Zustand, 320
Interaktionsübersichtselemente, 352
Kompositionsstrukturelemente, 411
Paketdiagrammelemente, 416
Zeitverlaufsdiagrammelemente, 386
Einführung, 10
Einschränken,
Classifier, 286
Einstelllungen,
Synchronisierung, 221
Einstellungen,
Versionskontrolle, 525
Eintrittspunkt,
zu Unterautomat hinzufügen, 328
Element,
Stile, 87
zu Favoriten hinzufügen, 85
Elemente,
aus Diagramm löschen, 110
aus Include-Dateien ignorieren, 525
aus Projekt löschen, 110
Aussehen ändern, 119
automatisches Layout, 128
benutzerdefinierte Bilder anwenden auf, 119
dokumentieren, 91, 118
Eigenschaften ändern, 86
einschränken, 114
ersetzen, 111
Größe anpassen, 128
Hyperlinks erstellen, 115
in einem Diagramm ausrichten, 128
in einem Diagramm suchen, 113
kopieren, 109
suchen, 111
umbenennen, 109
verschieben, 109
zu einem Diagramm hinzufügen, 107
zum Modell hinzufügen, 80, 106
Zustandsdiagramm einfügen, 320
Elementimport,
anzeigen, 88
Endbenutzer-Lizenzvereinbarung, 560, 564
Enthältbeziehung,
in einem Diagramm zeichnen, 143
Erstellen,
getter / setter Methoden, 394
Erweitern,
alle Klassenbereiche, 394

© 2016-2022 Altova GmbH

Evaluierungszeitraum,
für Altova-Software-Produkte, 561
von Altova Software-Produkten, 560
Exportieren,
UModel-Projekte in XMI, 454
Externe Applikationen,
von UModel aus öffnen, 518
Extras, 515
Optionen, 525
Extras (Menü),
benutzerdefinierte Befehle hinzufügen, 518

F

Farbe,
Syntaxfärbung - aktivieren /deaktivieren, 400
Favoriten (Fenster),
entfernen aus, 85
hinzufügen zu, 85
Fehler,
beim Code Engineering, 92
Fenster,
Standardeinstellungen wiederherstellen, 515
Forward Engineering, 59
Freigeben,
aus Versionskontrolle, 475

G

Gate,
Sequenzdiagramm, 364
Gehe zu,
Lebenslinie, 359
Generalisieren,
spezialisieren, 34
Generalisierung,
als Beziehung, 107, 133
erstellen, 133
Generalisierungen,
anzeigen, 88
Generated documentation,
options, 294
Generics,
Unterstützung für Java und C#, 286
Generieren,
Antwortnachricht automatisch generieren, 365
Komponentenrealisierungen automatisch generieren, 222
Sequenzdiagramm von Kommunikationsdiagramm, 348
UML-Projektdokumentation, 290
Geschwindigkeit,
Verbesserung, 169
Get,
getter / setter Methoden, 394

H

Hierarchie (Fenster), 88
Hierarchiediagramm,
in der Dokumentation angezeige Ebenen, 290
Hilfe (Menü),
Befehle, 538
Hilfslinie,
beim Ziehen mit der Maus, 525
Hinzufügen, 471
Diagramm zu Paket, 16
neues Projekt, 152
Paket zu Projekt, 16
Projekt in Versionskontrolle, 471
Versionskontrolle, 471
Hyperlinks,
im Dokumentationstext, 118

I

Ignorieren, 525
Elemente in Liste, 525
Verzeichnisse, 525
Importieren,
XMI in UModel, 454
Inkludieren,
.NET Framework, 164
Abhängigkeit, 16
UModel Projekt, 164
Instanz,
Diagramm, 40
mehrere Klassen, Anzeige, 394
Objekt, 40
Intelligentes,

Intelligentes,
Autokomplettieren, 25
Interaction Use, 364
Interaktinosoperand, 361
Interaktionsoperand,
mehrzeiliger, 361
Interaktionsoperator,
definieren, 361
Interaktionsübersicht,
Elemente einfügen, 352
Interaktionsübersichtsdiagramm, 351
Symbole, 494
Interface,
ball and socket, 394

J

Java,
Annotationen importieren, 208
Binärdateien importieren, 214
Code generieren, 170, 182
Code und Klassendateinamen, 224
Codegenerierungsoptionen, 176
Optionen für den Code-Import, 196
Quellcode importieren, 194

K

Katalog,
Datei - XMLSpy Katalogdatei, 525
Klasse, 34
abgeleitete, 34
abstrakte und konkrete, 25
Assoziationen, 25
Autokomplettierungsfenster aktivieren, 525
ball and socket interface, 394
Basis, 34
Basisklasse überschreiben, 394
Diagramme, 25
Eigenschaften hinzufügen, 25
erweitern, Bereiche reduzieren, 394
in Komponentendiagramm, 48
mehrere Instanzen in einem Diagramm, 394
Namensänderungen - Synchronisierung, 224
Operation - überschreiben, 394
Operationen hinzufügen, 25
Symbole, 489
Synchronisierung, 221
Syntaxfärbung, 400
Klassendiagramm, 394
neues hinzufügen, 25
Klassennamen ändern,
Auswirkung auf Codedateinamen, 224
Knoten,
Stile, 87
Kollaboration,
Kompositionsstrukturdiagramm, 411
Kommunikationsdiagramm, 347
Symbole, 490
von Sequenzdiagramm generieren, 348
Kompatibilität,
Projekte aktualisieren, 221
Komponente,
Diagramm, 48
Klasse einfügen, 48
Realisierung, 48
Symbole, 492
Komponentenansicht,
als Paket, 109
Komponentendiagramm, 413
Komponentenrealisierung,
automatische Generierung, 222
Komposition,
Assoziation - erstellen, 25
Kompositionsstruktur,
Elemente einfügen, 411
Kompositionsstrukturdiagramm, 410
Symbole, 491
Konkrete,
Klasse, 25

L

Layout (Menü),
Befehle, 512
Lebenslinie, 359
allgemeiner Wert, 386
Attribute, 359
Eigenschaft vom Typ Lebenslinie, 359
Gehe zu, 359

© 2016-2022 Altova GmbH

Linie,
orthogonal, 48
Linien,
benutzerdefinierte, 134
formatieren, 40
gerade, 134
Hilfslinien, 525
orthogonale, 134
Stil ändern, 134
verschieben, 134
Links,
in generated documentation, 294
Lizenz, 564
Informationen, 560
Lizenzüberwachung,
in Altova-Produkten, 562
Lokales Projekt, 461
Löschen, 516
Befehl aus der Symbolleiste löschen, 516
Symbol aus der Symbolleiste, 516
Symbolleiste, 517

M

Mail,
Projekt senden, 505
Manifest,
Artekfakt, 54
Mehrzeilig,
Use Case, 16
Mehrzeiliger,
Akteurtext, 16
Interaktionsoperand, 361
Meldungen (Fenster),
Referenz, 92
Menü,
Ansicht, 514
Befehl hinzufügen/löschen, 516
Extras, 515
Merge,
in Aktivität erstellen, 306
Methode,
Ausnahmeereignis hinzufügen, 394
Methoden,
getter / setter, 394
Modell,
Elemente hinzufügen, 80, 106
Klassennamen ändern - Auswirkung in Java, 224
Modellieren,
Performance verbessern, 169
Modell-Struktur (Fenster),
Einträge erweitern oder reduzieren, 80
Modellelemente anzeigen oder ausblenden, 80
Modellelemente sortieren, 80
Projekt anzeigen, 80
Symbolreferenz, 80

N

Nachricht, 365
Aufruf, 365
einfügen, 365
gehe zu Operation, 365
nummerieren, 365
Objekt erstellen, 365
Pfeile, 365
verschieben, 365
Zeitverlaufsdiagramm, 392
Nachrichtenpfeile verschieben, 365
Name,
Regionsnamen - ein- /ausblenden, 328
Native Oberfläche ausführen, 480
Neu,
Classifier, 222
Neue Zeile,
in Lebenslinie, 348
Interaktionsoperand, 361
Node,
Artefakt hinzufügen, 54
hinzufügen, 54
Nummerieren,
Nachrichten, 365

O

Objekt,
Diagramm, 40
Nachricht erstellen, 365
Symbole, 495
Verknüpfungen - Assoziationen, 40

© 2016-2022 Altova GmbH

Objektdiagramm, 414
Öffnen,
Projekt-Versionskontrolle, 461
OpenJDK,
Binärdateien importieren, 207
Operand,
Interaktion, 361
Operation,
Autokomplettierungsfenster, 525
automatisch zu Aktivität hinzufügen, 320
ein-/ausblenden, 394
Farbe, 400
gehe zu von Call-Nachricht, 365
überschreiben, 394
Vorlage, 288
wiederverwenden, 34
Operation automatisch hinzufügen, 320
Operationen,
hinzufügen, 25
Operator,
Interaktion, 361
Optionen, 525
Extras, 525
Versionskontrolle, 525
Options,
when generating documentation, 294
Ordner,
in Versionskontrolle abrufen, 466
Ordner abrufen,
Versionskontrolle, 466
Orthogonale,
Linie, 48
Orthogonaler,
Zustand, 328

P

Paket,
Standardpakete, 80
Symbolreferenz, 80
Paketdiagramm, 414
Abhängigkeiten erstellen, 414
Elemente einfügen, 416
Symbole, 496
Paketimport,
anzeigen, 88
Paketmerge, 416
Paketzusammenführung,
anzeigen, 88
Pakteimport, 416
Parameter,
Vorlage, 288
Per E-Mail senden,
Projekt, 505
Performance,
Verbesserung, 169
Pfad,
Projekt verschieben, 152
Projektordner ändern, 152
SPL-Vorlagenpfad, 545
Pretty Print,
in exportierten XMI-Dateien, 454
Projekt mit Pretty Print speichern, 152
Profile,
auf ein Paket anwenden, 160, 422
Definition, 421
erstellen, 422
vordefinierte, 422
Projekt, 525
3-Weg-Zusammenführung, 281, 283
anzeigen, 80
Arbeitsablauf, 152
Datei - aktualisieren, 221
Dokumentation generieren, 290
entfernen aus Versionskontrolle, 474
erstellen, 152
Hinzufügen in Versionskontrolle, 471
in Module aufteilen, 160
in Unterprojekte aufteilen, 160
letzten beim Starten öffnen, 525
Modellelemente hinzufügen oder entfernen, 80
offene Diagramme speichern, 525
Paket einfügen, 152
per E-Mail senden, 505
speichern - Pretty Print, 152
Standardcode, 525
Stile, 87
UModel Projekt inkludieren, 164
verschieben, 152
zusammenführen, 280
Projekt (Menü),
Befehle, 509
Projekt öffnen,
Versionskontrolle, 461

© 2016-2022 Altova GmbH

Projektsyntax,
überprüfen, 92
Projektzusammenführung,
3-Weg, 281
Provider,
auswählen, 461

R

Raster,
Elemente an Hilfslinien ausrichten, 16
Hilfslinien, 525
Realisierung,
Komponente, 48
Komponentenrealisierungen generieren, 222
Rechtliches, 560
Rechtschreibung,
prüfen, 91
Reduzieren,
Klassenbereiche, 394
Refactoring,
Klassennamen - Synchronisierung, 224
Referenz, 504
Region,
zu zusammengesetztem Zustand hinzufügen, 328
Regionsname,
ein- / ausblenden, 328
Reverse Engineering, 69
Root,
als Paket, 109
Katalog - XMLSpy, 525
Paket/Klasse synchronisieren, 221

S

SC,
Syntaxfärbung, 400
Schaltflächen,
Sichtbarkeitsschaltflächen, 394
Schnittstelle,
implementieren, 394
Sequenz,
Symbole, 499
Sequenzdiagramm, 356
Code generieren, 378
Code hinzufügen, 382
Combined Fragment, 361
Elemente einfügen, 357
Gate, 364
Interaction Use, 364
Lebenslinie, 359
Nachrichten, 365
von Kommunikationsdiagarmm generieren, 348
Zustandsinvariante, 365
Sequenzdiagramme,
anhand von Getter / Setter generieren, 377
anhand von Quellcode generieren, 371
mehrere generieren, 377
Set,
getter / setter Methoden, 394
Shortcuts,
löschen, 521
zuweisen, 521
Sichtbarkeit,
Schaltflächen - auswählen, 394
Signatur,
Vorlage, 286, 287
Slot,
ein-/ausblenden, 394
Software-Produktlizenz, 564
Speichern,
Diagramm als Bild, 505
Spezialisieren,
generalisieren, 34
SPL, 543
benutzerdefinierte, 221
Codeblöcke, 544
foreach, 556
SPL-Vorlagen,
Vorlagenpfad, 545
Standard,
Projektcode, 525
SPL-Vorlagen, 221
Starten,
mit letzten Projekt, 525
Status aktualisieren,
Versionskontrolle, 480
Stereotype,
auf Elemente anwenden, 147, 427
Beispiel, 427
Beispiele, 145, 421
benutzerdefinierte Stile hinzufügen, 432

© 2016-2022 Altova GmbH

Stereotype,
benutzerdefinierte Symbole hinzufügen, 432
Definition, 145
erstellen, 424, 427
zum Fenster "Eigenschaften" hinzufügen, 86
Stile,
auf Diagramme anwenden, 127
auf Elemente anwenden, 119
auf Linien anwenden, 134
kaskadierend, 127
kaskadierende, 119, 134
Vorrangigkeit, 119, 127, 134
Stile (Fenster), 87
Strukturdiagramme,
Diagramme, 394
StyleVision,
generierte Dokumentation anpassen mit, 290, 299
Suchen,
Diagramme, 111
Elemente, 111
Text, 111
Symbol, 493
Aktivitätsdiagramm, 488
Deployment, 493
groß anzeigen, 524
Klasse, 489
Komponente, 492
Objekt, 495
Sequenz, 499
Verteilung, 493
zur Symbolleiste/zum Menü hinzufügen, 516
Symbole,
Interaktionsübersichtsdiagramm, 494
Kommunikationsdiagramm, 490
Kompositionsstrukturdiagramm, 491
Paketdiagramm, 496
Use Case, 502
XML-Schemadiagramm, 503
Zeitverlaufsdiagramm, 501
Zustandsdiagramm, 500
Symbolleiste,
aktivieren/deaktivieren, 517
Befehl hinzufügen, 516
große Symbole anzeigen, 524
neue erstellen, 517
Standardeinstellungen wiederherstellen, 515
Symbolleiste & Menübefehle zurücksetzen, 517
Synchronisieren,
mit neuem Ordner, 152
Root/Paket/Klasse, 221
Synchronisierung, 224
Änderung von Klassennamen, 224
Einstellungen, 221
Klasse und Codedateinamen, 224
Syntaxfärbung, 400

T

Tastaturkürzel,
in Tooltipp anzeigen, 524
löschen, 521
zuweisen, 521
Tick-Symbol,
Zeitverlaufsdiagramm, 389
Tooltipp,
anzeigen, 524
Tastaturkürzel anzeigen, 524
Transition, 320
Aktivitätsdiagramm hinzufügen, 320
Trigger definieren, 320
zwischen Zuständen definieren, 320
Trigger,
Transitions-Trigger definieren, 320
Tutorial,
Beispieldateien, 14
Typ,
Eigenschaft - anzeigen, 288
Typisieren,
Eigenschaft - als Lebenslinie, 359

U

Überschreiben,
Klassenoperationen, 394
Übersicht (Fenster),
scrollen, 90
Umbenennen,
Classifier, 222
UML,
Diagramme, 301
Sichtbarkeitsschaltflächen, 394
Variablen, 545

© 2016-2022 Altova GmbH

UML,
Vorlagen, 286
UModel,
Einführung, 11
Hauptfunktionen, 11
UModel Diagrammsymbole, 487
UModel Einführung, 10
UModel-Projekte,
öffnen, speichern, erstellen, 15
Ump,
Dateierweiterung, 152
Projektverzeichnis ändern, 152
Unterautomatenzustand,
neuen Eintrtts-/Austrittspunkt hinzufügen, 328
Unterprojekt,
anhand eines Hauptprojekts erstellen, 160
wieder in das Hauptprojekt integrieren, 160
Unterschiede anzeigen, 478
URL,
Datei öffnen von, 505
Use Case,
Assoziation, 16
Bereiche, 16
hinzufügen, 16
mehrzeilig, 16
Symbole, 502
Use Case diagram, 347

V

Variablen,
UML, 545
VB.NET,
Binärdateien importieren, 206
Code generieren, 170
Codegenerierungsoptionen, 176
Optionen für den Code-Import, 196
Quellcode importieren, 194
Verbessern,
Performance, 169
Verhaltensdiagramme, 302
Verlauf,
anzeigen, 476
Verlauf anzeigen, 476
Verschieben,
Projekt, 152
Versionsdateien vergleichen, 478
Versionskontrolle,
aktivieren /deaktivieren, 464
aktuellste Version holen, 465
auschecken, 468
Auschecken rückgängig, 470
ausgliedern, 474
Befehle, 461
Datei abrufen, 465
Eigenschaften, 479
Einchecken, 469
Hinzufügen in Versionskontrolle, 471
installieren eines Versionskontroll-Plug-in, 456
native Oberfläche ausführen, 480
Optionen / Einstellungen, 525
Projekt öffnen, 461
Status aktualisieren, 480
Unterschiede anzeigen, 478
Verlauf anzeigen, 476
Versionskontrollsystem wechseln, 480
Verteilung,
Symbole, 493
Vertrieb,
von Altova Software-Produkten, 560
von Altova-Software-Produkten, 561
Verwendung,
Abhängigkeit, 48
Vorlage, 288
Verzeichnis,
beim Zusammenführen igrnoieren, 525
Projektverzeichnis ändern, 152
Verzweigung,
in Aktivität erstellen, 306
Vorlage,
Signatur, 286, 287
Verwendung, 288
Vorlagen,
benutzerdefinierte SPL, 221
Operation/Parameter, 288
SPL-Vorlagen, 545

W

Warnungen,
beim Code Engineering, 92
Wertverlaufslinie,

Wertverlaufslinie,
Zeitverlaufsdiagramm, 386
Wiederherstellen,
Symbolleisten und Fenster, 515

X

XMI,
importieren und exportieren, 454
XML-Schema,
anhand von Modell generieren, 444
Diagramme, 435
Diagramme erstellen, 441
ins Modell importieren, 436
modellieren, 441, 444
Namespace deklarieren, 441
XML-Schemadiagramm,
Symbole, 503

Z

Zeilenumbruch,
im Akteurtext, 16
Zeitbedingung,
Zeitverlaufsdiagramm, 392
Zeitdauerbedingung,
Zeitverlaufsdiagramm, 391
Zeitlinie,
Zustandsänderungen definieren, 386
Zeitverlaufsdiagramm, 385, 386
auslösendes Ereignis, 390
Elemente einfügen, 386
Lebenslinie, 386
Nachricht, 392
Symbole, 501
Tick-Symbol, 389
wechseln zwischen Typen, 386
Wertverlaufslinie, 386
Zeitbedingung, 392
Zeitdauerbedingung, 391
Zeitlinie, 386
Zurücksetzen,
Symbolleiste & Menübefehle, 517
Zusammenführen,
Projekte, 280
Verzeichnis ignorieren, 525
Zusammenführung,
3-Weg-Projektzusammenführung, 281, 283
Zusammengesetzter Zustand,
Region hinzufügen, 328
Zustand, 320
Aktivität hinzufügen, 320
einfachen einfügen, 320
orthogonaler, 328
Transition definieren, 320
Unterautomatenzustand, 328
zusammengesetzter, 328
Zustandsänderungen,
auf einer Zeitlinie definieren, 386
Zustandsdiagramm,
Diagrammelemente, 340
Elemente einfügen, 320
Symbole, 500
zusammengesetzte Zustände - Regionen, 328
Zustände, Aktivitäten, Transitionen, 320
Zustandsdiragramm, 319
Zustandsinvariante, 365

© 2016-2022 Altova GmbH