UML Einführung

Die Unified Modeling Language (UML) ist eine objektorientierte Notation zur Spezifikation und Modellierung von Softwaresystemen. UML ist sehr mächtig und enthält eine grosse Anzahl von Diagrammen die sich in folgende Typen unterteilen lassen:

- Anwendungsfalldiagramme
- Klassendiagramme
- Verhaltensdiagramme
    - Aktivitätsdiagramme
    - Kollaborationsdiagramme
    - Sequenzdiagramme
    - Zustandsdiagramme
- Implementierungsdiagramme
    - Komponentendiagramme
    - Verteilungsdiagramme

Da im Softwarepraktikum der Schwerpunkt auf Klassen- und Zustandsdiagrammen liegt, werden in dieser kurzen Einführung nur die wichtigsten Aspekte dieser beiden Typen betrachtet. Beachten Sie dazu auch das Tutorial „A Short Introduction to the UML“ [Intr03].

Klassendiagramme

Klassendiagramme beschreiben die statische Struktur eines Models, d.h. die einzelnen Klassen und ihre Zusammenhänge. Ein Klassendiagramm setzt sich dabei aus folgenden Elementen zusammen:

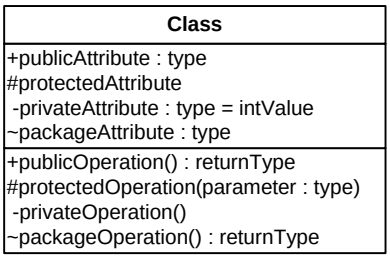
Klassen

Eine Klasse definiert die Struktur (Attribute und Operationen), das Verhalten und die Beziehungen einer Menge von Objekten. Klassen werden als Rechtecke dargestellt, die sich aus den Rubriken Klassenname, Attribute und Operationen zusammensetzen. Die einzelnen Rubriken werden dabei durch eine horizontale Linie voneinander getrennt.

Attribute werden mit ihrem Namen aufgeführt und können zusätzliche Angaben zu ihrem Typ und einem Initialwert enthalten. Die äussere Sichtbarkeit von Attributen wird durch +, #, -, ~ für public-, protected-, private- und package-Sichtbarkeit angegeben. Klassenattribute, die nicht zu einem einzelnen Objekt gehören, sondern ein Attribut einer Klasse sind, werden unterstrichen.

Operationen werden ebenfalls durch ihren Namen und optional durch ihre Attribute, deren Typ und eventuell vorhandene Initialwerte beschrieben. Die Sichtbarkeit wird wie bei Attributen mit

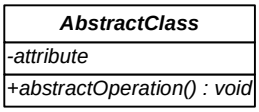
Hilfe von +, #, -, ~ angegeben und Klassenoperationen werden, analog zu Klassenattributen, unterstrichen.



Abstrakte Klassen

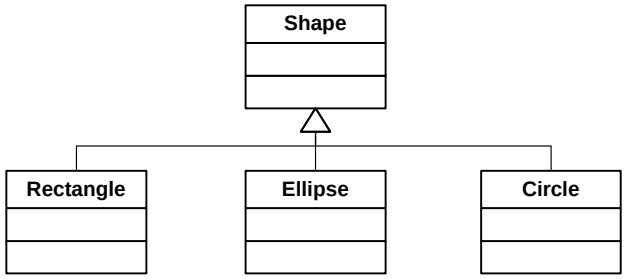
Von einer abstrakten Klasse können keine Exemplare erzeugt werden. Eine abstrakte Klasse ist unvollständig und bildet die Basis für von ihr abgeleitete Klassen. Abstrakte Klassen werden wie normale Klasse dargestellt, der Klassenname wird jedoch kursiv gesetzt.

Abstrakte Operationen haben nur eine Signatur (Name, Parameter, Rückgabewert) und keine Implementierung und können nur in abstrakten Klassen auftreten. Abstrakte Operationen werden kursiv dargestellt.



Generalisierungen, Spezialisierungen

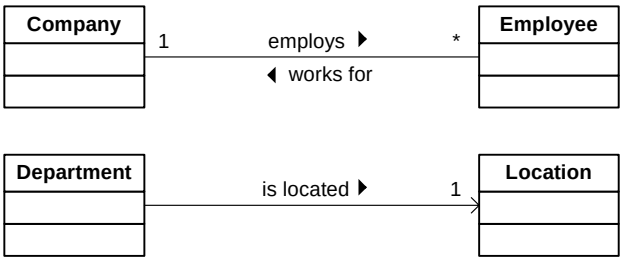
Vererbung als Beziehung zwischen einer Ober- und einer Unterklasse ermöglicht das Zugänglichmachen von Attributen und Operationen der Oberklasse für die Unterklasse. Eine Vererbungsbeziehung wird als Linie vom spezifischen zum generellen Element mit einem nicht ausgefüllten Pfeil am Ende dargestellt.



Assoziationen

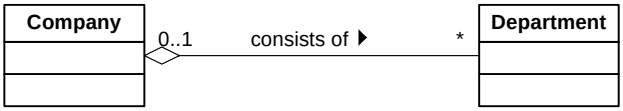
Durch Assoziationen werden Beziehungen zwischen Klassen beschrieben. Eine Assoziation ist normalerweise eine Beziehung zwischen unterschiedlichen Klassen. Sie kann aber auch reflexiv sein, d.h. eine Klasse hat eine Beziehung zu sich selbst. Assoziationen sind notwendig, damit Objekte (die Exemplare der Klassen) miteinander kommunizieren können. Assoziationen werden durch Linien zwischen den beteiligten Klassen dargestellt. Jede Beziehung sollte mit einem Namen versehen werden, der beschreibt warum diese Beziehung besteht. An den jeweiligen Enden sollte die Kardinalität als einzelne Zahl oder als Wertebereich angegeben werden. Die

Kardinalität der Assoziation gibt an, mit wie vielen Objekten der gegenüberliegenden Klasse ein Objekt assoziiert werden kann.
Eine gerichtete Assoziation ist eine Assoziation, bei der von einer der beteiligten Klasse zur anderen navigiert werden kann, jedoch nicht umgekehrt.



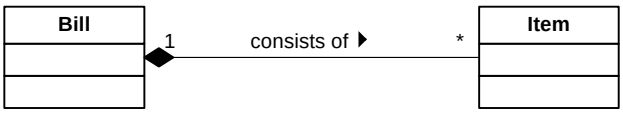
Aggregation

Eine Aggregation ist eine Assoziation, bei der die beteiligten Klassen keine gleichwertige Beziehung führen, sondern eine Ganzes-Teile-Hierarchie darstellen. In einer Aggregation nimmt das Ganze (Aggregat) stellvertretend Aufgaben für seine Teile wahr. So kann das Aggregat z.B. Operationen haben, die keine unmittelbare Veränderung in ihm zur Folge haben, sondern die Nachricht an seine Teile weiterleiten. Instanzen der Teile können jedoch auch unabhängig vom Aggregat existieren.
Eine Aggregation wird wie eine Assoziation durch eine Linie dargestellt und zusätzlich auf der Seite des Aggregats mit einer kleinen Raute versehen.



Komposition

Eine strengere Form einer Aggregation stellt eine Komposition dar, indem die Teile vom Ganzen existenzabhängig sind. Die Lebenszeit der Teile ist dabei von der Lebenszeit des Ganzen (Aggregat) abhängig, d.h. sie werden zusammen oder nach dem Aggregat erzeugt und sie werden vor oder mit dem Aggregat zerstört.
Zur Darstellung einer Komposition dient ebenfalls eine Linie mit einer kleinen Raute auf der Seite des Aggregats, im Gegensatz zur Aggregation wird die Raute jedoch ausgefüllt.



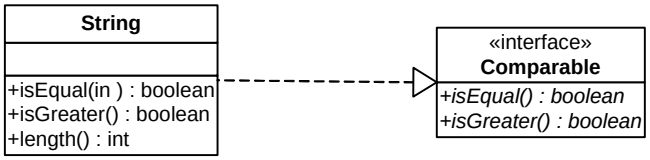
Schnittstellen (Interfaces)

Interfaces beschreiben einen ausgewählten Teil des extern sichtbaren Verhaltens von Modellelementen (hauptsächlich Klassen) und enthalten Signaturen von Methoden, welche Klassen, die diese Schnittstelle bereitstellen wollen, implementieren müssen. Konsequenterweise haben Interfaces keine Attribute. Interfaces werden mit dem Schlüsselwort *interface*

gekennzeichnet. Da alle Methoden eines Interfaces zwingend abstrakt sind, müssen sie nicht als abstrakt gekennzeichnet werden.

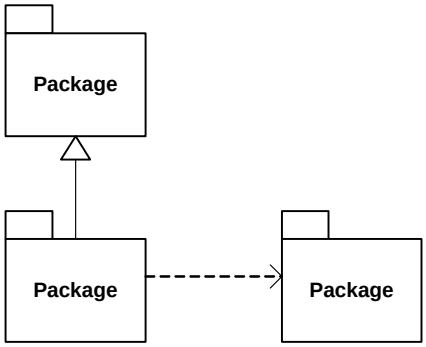
Realisierungsbeziehung

Zwischen einem Interface und der Klasse, die dieses Interface implementiert, besteht eine Realisierungsbeziehung, die durch eine gestrichelte Linie mit einem nicht ausgefüllten Pfeil dargestellt wird.



Pakete (Packages)

Pakete sind Ansammlungen von Modellelementen beliebigen Typs, mit denen das Gesamtmodell in leichter überschaubare Teile aufgeteilt wird. Innerhalb eines Pakets muss der Name eines Elements eindeutig sein. Pakete werden in Form von Aktenregistern dargestellt und können auch ineinander geschachtelt werden.
Abhängigkeiten zwischen Paketen werden durch gestrichelte Pfeile notiert, die in Richtung des unabhängigen Pakets zeigen. Abhängigkeitsbeziehungen zwischen Paketen können auch Generalisierungsbeziehungen sein.

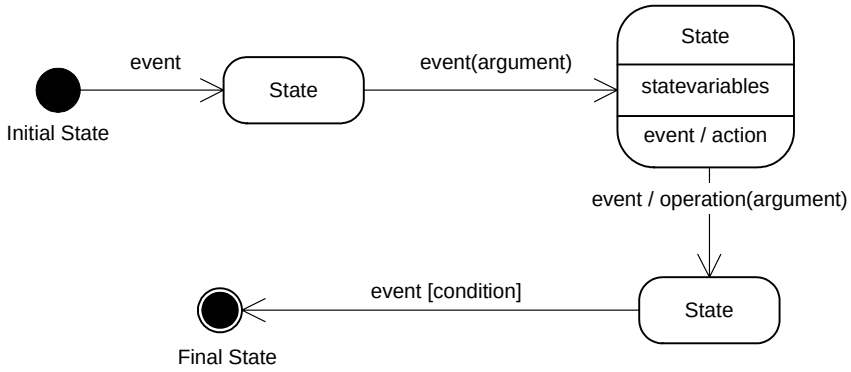


Statecharts

Ein Statechart stellt eine Menge von Zuständen dar, die ein Objekt während seiner Lebenszeit einnehmen kann und gibt an, aufgrund welcher Ereignisse ein Zustandsübergang eintritt. Statecharts bestehen aus einer Menge von Zuständen und Zustandsübergängen sowie einem Anfangszustand und einem oder mehreren Endzuständen.

Ein Zustand gehört zu genau einer Klasse und stellt eine Zusammenfassung von möglichen Attributwerten dar, die die Objekte dieser Klasse einnehmen können.
Jeder Zustand kann entweder hierarchisch oder parallel in weitere Diagramme zerlegt werden. Zustände werden durch Rechtecke mit abgerundeten Ecken dargestellt und können in bis zu drei Bereiche unterteilt werden. Der oberste Bereich enthält den Namen des Zustandes, im mittleren

Bereich können Zustandsvariablen aufgelistet werden und der letzte Bereich kann eine Liste von internen Ereignissen, Bedingungen und aus ihnen resultierende Operationen enthalten.
Zu einem Startzustand kann kein Zustandsübergang stattfinden, und von einem Endzustand führt keine Transition mehr weg. Startzustände werden als kleine ausgefüllte Kreise dargestellt, Endzustände durch nicht ausgefüllte Kreise, in denen ein konzentrisch ausgefüllter Kreis liegt. Zustandsübergänge werden durch einen Pfeil dargestellt und mit den Ereignissen (events) beschrieben, die den Übergang auslösen. Ereignisse können mit einer Bedingung (condition) und einer auszuführenden Aktion (action) versehen werden. Zusätzlich können einem Ereignis Argumente übergeben werden.



Referenzen

[Rumb99]	J. Rumbaugh, I. Jacobson, G. Booch: *The Unified Modeling Language Reference Manual*. Addison-Wesley. 1999
[Intr03]	*A Short Introduction to the UML*
[Toge03]	TogetherSoft: *Practical UML – A Hands-On Introduction for Developers*. http://www.togethersoft.com/services/practical_guides/umlonlinecourse/