6 Objekte/Klassen/Assoziationen

In der objektorientierten Programmierung besitzt ein **Objekt** einen bestimmten Zustand und reagiert mit einem definierten Verhalten auf seine Umgebung.

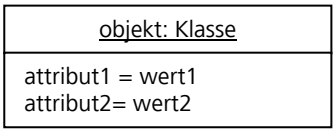
- **Zustand**: umfasst die Werte der Attribute sowie die Beziehungen zu anderen Objekten
- **Verhalten**: wird durch die Operationen beschrieben; eine Änderung oder Abfrage des Zustands ist nur durch Aufruf einer Operation möglich

Attribute

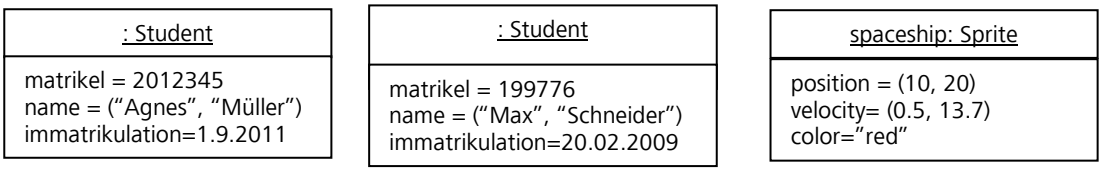
Die **Attribute** beschreiben die Daten, die von Objekten einer Klasse angenommen werden können. Jedes Attribut besitzt einen bestimmten Typ. Alle Objekte einer Klasse besitzen dieselben Attribute, können jedoch unterschiedliche Attributwerte haben. **Attributbezeichner** sind in der Regel kleingeschriebene Substantive.

UML Objektdiagramm

Im UML-Diagramm wird ein Objekt als zweigeteiltes Rechteck dargestellt. Das obere Rechteck enthält den unterstrichenen Klassennamen, optional kann der Objektname vor den Doppelpunkt gestellt werden. Die untere Hälfte enthält die Attribute. Es ist auch möglich, nur den unterstrichenen Klassennamen oder nur den Namen des Objekts anzugeben.

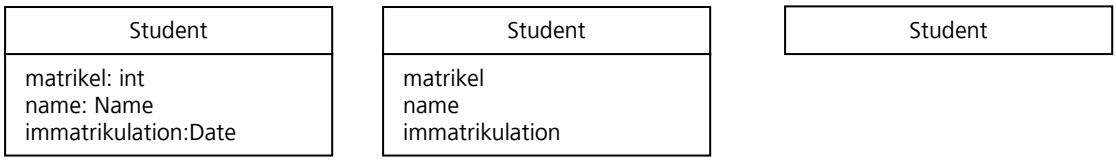


Beispiel für drei Objekte



Klasse

Eine **Klasse** definiert eine Kollektion von Objekten mit gleicher Struktur (Attribute), Verhalten (Operationen) und Beziehungen (Assoziationen, Generalisierungen). Sie besitzt einen Mechanismus um neue Objekte zu erzeugen (object factory).
Das UML-Klassendiagramm ist ein dreigeteiltes Rechteck. Oben steht der Klassenname, in der Mitte die Attribute und im unteren Teil die Operationen. Optional können Attribute und/oder Operationen auch weggelassen werden.
Folgendes Beispiel zeigt unterschiedliche Varianten zur Darstellung einer Klasse Student im Klassendiagramm:

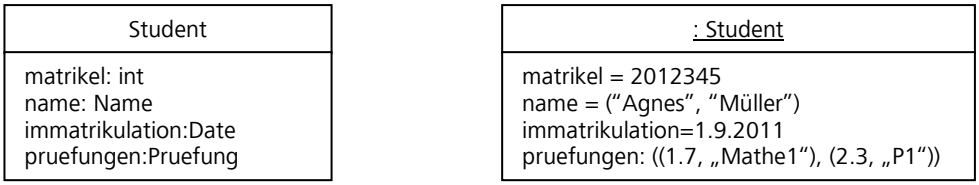


Datentypen: Datentypen von Variablen oder die Rückgabetypen von Methoden werden durch einen Doppelpunkt getrennt hintenan gestellt. Auch bei Methodenargumenten folgt der Datentyp durch Doppelpunkt getrennt dem Argument.

Datenstrukturen: bei Arrays, Listen, oder andere Datenstrukturen wird der Datentyp der Elemente angegeben und folgendes Symbol hinten angestellt: [*]. Beispielsweise wird ein Array von Point Objekten wie folgt im UML-Diagramm dargestellt: points: Point[*]
Der **Datentyp** kann optional hinter einen Doppelpunkt angehängt werden. Das kann sein:

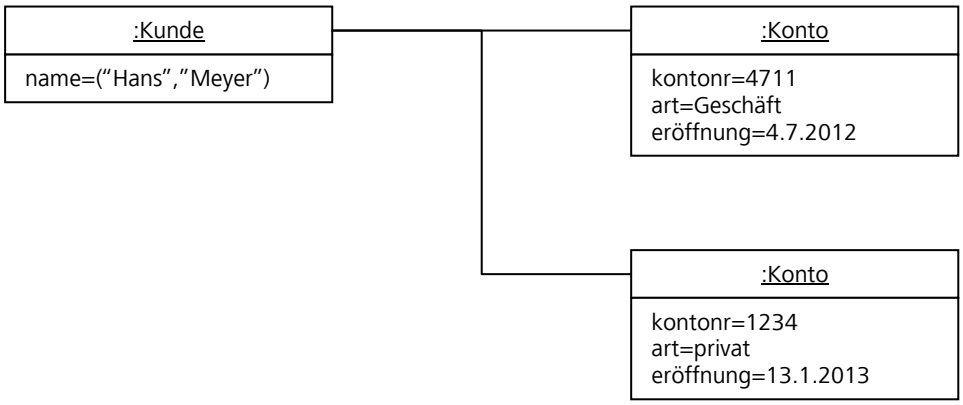
- Datentyp (float, int, BigInteger, String)
- Aufzählungstyp (enum in Java/C)
- Klasse

Ein Attribut kann eine **Multiplizität** haben, die in eckigen Klammern angegeben wird. Zum Beispiel könnten die erfolgreich abgelegten Prüfungen eines Student-Objekts in einem Attribut prüfungen definiert sein.



Assoziationen

Zwischen den Objekten von Klassen können Objektbeziehungen existieren. Eine **Assoziation** beschreibt alle gleichartigen Objektbeziehungen zwischen Klassen.
Beispiel*: Hans Meyer eröffnet am 4.7.2012 ein Geschäftskonto Nr. 4711 und am 13.1.2013 ein privates Konto Nr. 1234.*



Im Klassendiagramm wird eine Assoziation als Linie zwischen den Klassen dargestellt. Sie bedeutet zunächst, dass sich die beiden Klassen kennen.

Multiplizität

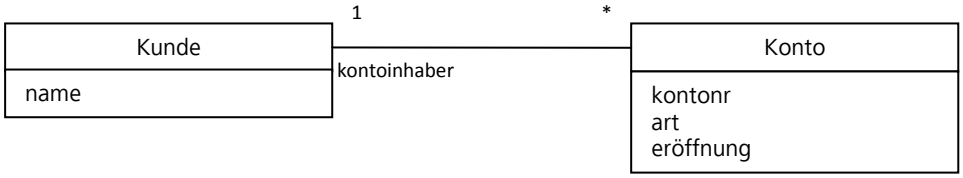
An den Enden der Assoziation wird mit der **Multiplizität** angegeben, wie viele Objekte ein bestimmtes Objekt kennen kann. Eine Assoziation mit einer Multiplizität 1 oder größer heißt **Muss-Assoziation.** Eine Assoziation mit Multiplizität 0 oder größer heißt **Kann-Assoziation.**
Im obigen Beispiel:

- Kann-Assoziation von Kunde zu Konto (*): jeder Kunde kann mehrere Konten besitzen
- Muss-Assoziation von Konto zu Kunde (1): jedes Konto gehört genau einem Kunden

Aufgabe: Wie ändert sich die Assoziation von Kunde zu Konto, wenn sie die Multiplizität 1…* erhält?

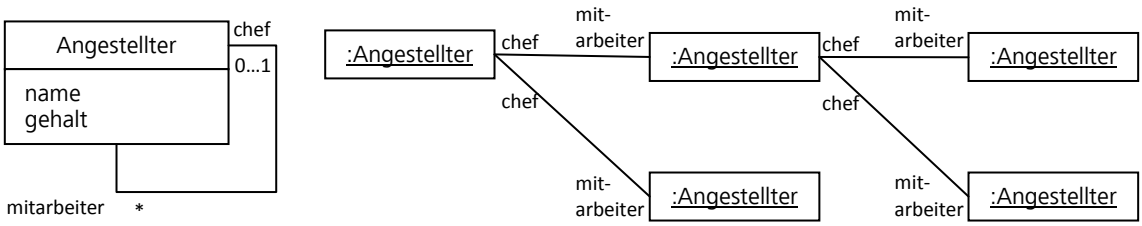
Assoziationsname

Eine Assoziation kann benannt werden. Der Name beschreibt im Allgemeinen eine Richtung der Assoziation, die Richtung kann mit einem schwarzen Dreieck angezeigt werden.
Weiter kann die **Rolle** einer Klasse in der Assoziation benannt werden. Sie wird an das Ende der Linie eingetragen, in der Nähe der Klasse, die beschrieben wird.



Aufgabe: a) Ein Konto soll zusätzlich zum Kontoinhaber beliebig viele Kontoberechtigte haben. Ergänzen Sie das UML-Diagramm. Zeichnen Sie für diesen Fall ein Objektdiagramm.
b) Erstellen Sie mit Eclipse ein compilierbares Projekt, das dieses UML-Diagramm realisiert.

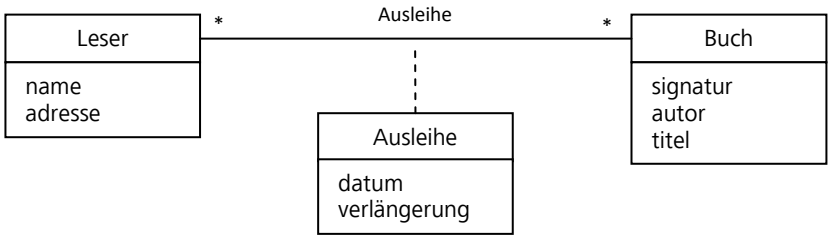
Eine **reflexive Assoziation** ist gegeben, wenn Objekte einer Klasse Beziehungen zu Objekten der gleichen Klasse haben, wie bei einer Liste oder einer Baumstruktur von Objekten. Folgendes Beispiel zeigt die Assoziation chef-mitarbeiter als Klassen- und Objektdiagramm.



Rollenname und Assoziationsname müssen angegeben werden, wenn es mehr als eine Assoziation zwischen zwei Klassen gibt, oder bei reflexiven Assoziationen.

Assoziationsklasse

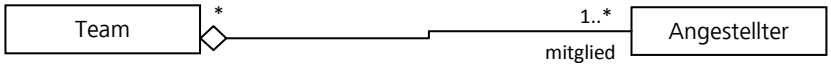
Eine Assoziation kann zusätzlich die Eigenschaften einer Klasse (Attribute, Operationen) haben. Dies wird im UML-Diagramm als Klasse dargestellt, die über eine gestrichelte Linie mit der Assoziation verbunden ist. Ein Beispiel ist die Assoziationsklasse Ausleihe, die die entsprechende Assoziation zwischen den Klassen Buch und Leser beschreibt.



Aufgabe: Wie würden Sie das in Java implementieren? Skizzieren Sie den Quelltext.

Aggregation

Eine **Aggregation** ist eine Assoziation bei der zwischen den Objekten der beteiligten Klassen eine Rangordnung besteht, der Art „ist Teil von“ oder „besteht aus“. Die Objekte einer Assoziation bilden einen gerichteten azyklischen Graphen: wenn A Teil von B ist, kann B nicht Teil von A sein.
Zum Beispiel können Objekt der Klasse Angestellter einem (oder auch mehreren) Teams angehören:



Eine **Komposition** ist eine Aggregation, bei der gilt:

- ein Objekt kann nur Komponente eines einzigen Objekts der Aggregat-Klasse sein, d.h. deren Multiplizität darf nicht größer als 1 sein
- das Ganze ist verantwortlich für das Erzeugen und Löschen der Teile
- wird das Ganze gelöscht, werden automatisch die Teile gelöscht

Zum Beispiel werden in Java Swing Objekte der Klasse Component (oder Unterklassen) zur Klasse JFrame hinzugefügt.