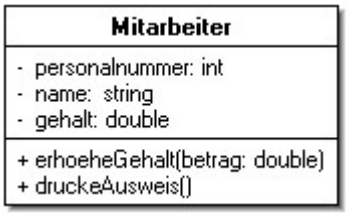
Max-Eyth-Schule	Klassendiagramm in der ObjektOrientierten Analyse	Klasse:
Tag:		Name:

Klassen in UML – das Klassendiagramm

Das Klassendiagramm ist das mit Abstand wichtigste Diagramm der UML. Hier wird die Struktur des Systems modelliert mit Klassen und Assoziationen. Es stellt den Aufbau von Klassen dar und die Zusammenhänge zwischen den Klassen. Es ist die Vorlage für die spätere Implementierung. Je mehr Informationen in diesem Diagramm stecken, desto genauer ist das Software-System modelliert und damit spezifiziert. Je exakter das System spezifiziert ist, desto klarer und einfacher ist die Implementierung.

Eine **Klasse** fasst **Objekt**e mit gleichen Eigenschaften (**Attribut**en) und gleichem Verhalten (**Operation**en, **Methode**n) zusammen. Eine **Klasse** ist eine Schablone oder ein Bauplan für eine Menge gleichartiger Objekte. Ein **Objekt** ist ein konkretes Exemplar (eine **Instanz**) einer **Klasse**. Eine Klasse steht mit anderen Klassen in Beziehung (für Leser mit Datenbankkenntnissen: ähnlich wie im ER-Modell). UML verwendet den Begriff **Assoziation** für Beziehung.

Die Klasse Mitarbeiter im UML-Klassendiagramm



Eine **Klasse** wird als Rechteck dargestellt, das in der Regel in drei Felder aufgeteilt ist. Im oberen Feld steht der **Klassenname**, im zweiten Feld stehen die **Attribute**, im dritten Feld die **Operationen** auch **Methoden** genannt, stets mit einem runden Klammernpaar (), evtl. mit **Parametern** und **Rückgabetypen**.

Der **Klassenname** ist stets ein Substantiv im Singular und wird mit großem Anfangsbuchstaben geschrieben. Beispiele: Mitarbeiter, PKW, Kunde.

Mehrere Methoden mit demselben Namen

Eine Klasse kann mehrere Methoden mit demselben Namen haben, das macht durchaus Sinn. Jedoch müssen sich die beiden Methoden an mindestens einer Stelle unterscheiden, damit der Compiler entscheiden kann, welche Methode nun tatsächlich auszuführen ist.

Gibt es in einer Klasse zwei Methoden mit demselben Namen, so wird diese **Methode überladen (**Engl: **method overloading)**. Der Methodenname zusammen mit den Parametern und dem Rückgabetyp heißt **Signatur** oder auch **Schnittstelle**. Die Signatur einer Methode muss eindeutig sein, es kann keine zweite genau gleiche geben.

Beispiel: Methode `erhoeheGehalt()`, die beiden Methoden heißen gleich, unterscheiden sich aber in ihren Parametern.

Sichtbarkeit – private und public

Innerhalb einer Klasse sind alle Methoden und Attribute für die Methoden sichtbar. Um die **Datenkapselung** (Geheimnisprinzip) umzusetzen und damit die Sichtbarkeit von Attributen, und Methoden nach außen hin, auch im Klassendiagramm auszudrücken, werden "+"- , "#"- und "-"-Zeichen eingefügt, mit der Bedeutung:

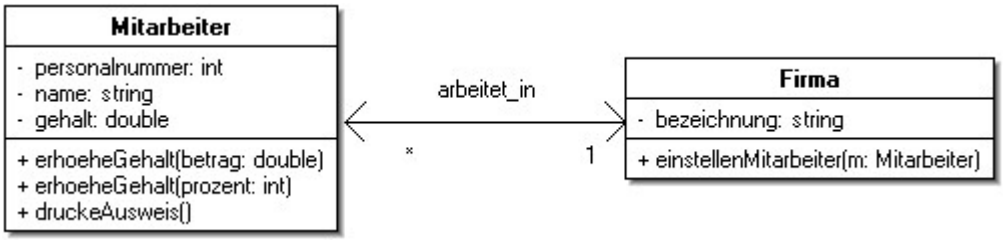
+	**public** / öffentlich	für andere Klassen sichtbar und benutzbar
-	**private** / privat	no public viewing, Element ist für andere Klassen nicht sichtbar
#	**protected** / privat	Elemente verhalten sich wie private-Elemente werden nur bei Ableitung der Klasse anders behandelt (siehe AB Vererbung-Übung Nr. 5)

In der Regel sind Attribute privat und Methoden öffentlich.

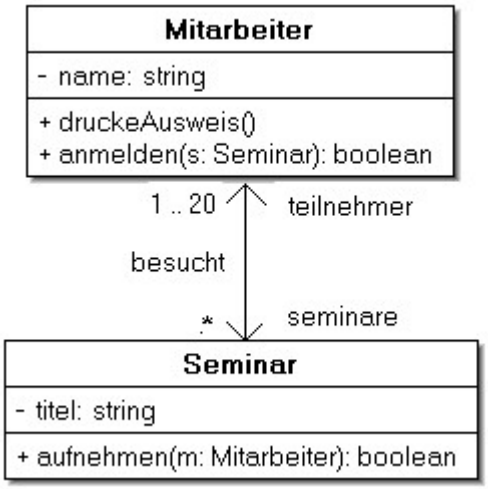
Assoziationen zwischen Klassen

Eine **Assoziation** zwischen 2 Klassen repräsentiert eine Beziehung zwischen den Objekten. Sie sagt zunächst nur, dass ein Objekt einer Klasse ein oder mehrere Objekte einer anderen Klasse kennt. Sie wird als Linie zwischen den beiden Klassen gezeichnet, mit dem Assoziationsnamen beschriftet und an den Enden ggf. mit Pfeilspitzen und der Angabe versehen, wie viele Objekte ein Objekt kennen kann. Diese Angabe zur Häufigkeit an einer Pfeilspitze, im Beispiel 1 oder *, heißt **Multiplizität** (für Leser mit Datenbankkenntnissen: Kardinalität im ER-Modell). Die "1" bedeutet "genau 1", beispielsweise: ein Mitarbeiter arbeitet in genau einer Firma. Der "*" bedeutet "mehrere" und schließt die 0 ein, beispielsweise: für eine Firma können mehrere Mitarbeiter arbeiten. Schließt die Multiplizität die "0" ein, ist es eine **"kann"-Assoziation**, ein Objekt kann eins oder mehrere Objekte der anderen Klasse kennen. Ist die "0" nicht eingeschlossen, ist es eine **"muss"-Assoziation**, ein Objekt kennt mindestens eins oder mehrere Objekte der anderen Klasse.

Beispiel Mitarbeiter und Firma im UML-Klassendiagramm



Ein Mitarbeiter arbeitet in genau einer Firma (1), er kennt seine Firma (Pfeilspitze bei Firma). Für eine Firma können viele Mitarbeiter arbeiten (*), die Firma kennt selbstverständlich ihre Mitarbeiter (Pfeilspitze bei Mitarbeiter). Mögliche **Multiplizität**en und ihre Bedeutung:



Zusätzlich können an den Enden einer Assoziation die **Rollen** der Objekte in Bezug auf die Assoziation angegeben werden. Beispiel: Ein Mitarbeiter kann an mehreren Seminaren teilnehmen, für diese Assoziation ist er `teilnehmer`. Rollen werden später als weitere Attribute in den Klassen implementiert, dabei werden die Rollennamen als Attributnamen verwendet.

Realisierung in C++:

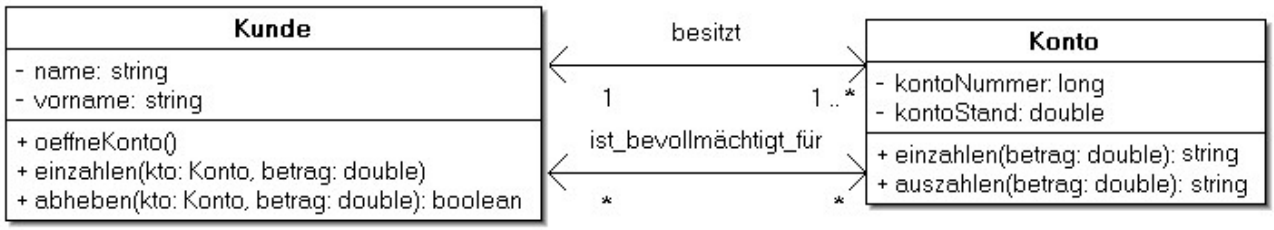
In der Klasse `Mitarbeiter:` `list <Seminar*> seminare;`

In der Klasse `Seminar:` `Mitarbeiter*[20] teilnehmer;`

Weitere Beispiele für Assoziationen

Bank: Ein Kunde besitzt mehrere Konten, ein Konto gehört genau einem Kunden. Ein Kunde kann für mehrere Konten bevollmächtigt sein, für ein Konto kann es mehrere bevollmächtigte Kunden geben.

Zwischen 2 Klassen kann es auch 2 oder mehr Assoziationen geben.



Aufgabe: Realisiere die Assoziationen in C++!

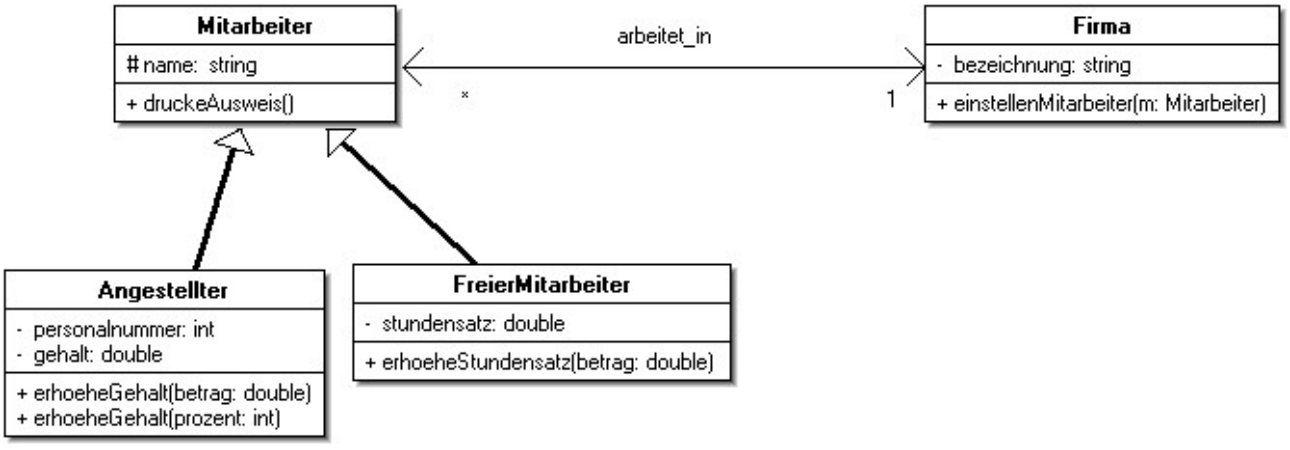
Ein Angestellter kann mehrere Mitarbeiter haben und einen Chef. Dies ist ein Beispiel für eine **reflexive Assoziation**, eine Assoziation zwischen einer Klasse und sich selbst.

Aufgabe: Realisiere die Assoziation in C++!

Eine spezielle Assoziation zwischen zwei Klassen – die Vererbung

Bei der Vererbung gibt es eine Superklasse und eine Subklasse, die Subklasse "**erbt"** alle Attribute, Methoden und Assoziationen von der Superklasse. Ein Objekt der Subklasse "**ist ein**" Objekt der Superklasse, aber mit zusätzlichen Attributen und/oder zusätzlichen Methoden. Das UML-Symbol für Vererbung ist der Pfeil mit der Dreiecksspitze.

Beispiel: Ein Angestellter ist ein Mitarbeiter, ein freier Mitarbeiter ist ein Mitarbeiter, beide erben von Mitarbeiter.



Ein Angestellter hat einen Namen, obwohl dieses Attribut nicht in der Klasse Angestellter definiert ist. Er bzw. sie hat auch die Methode `druckeAusweis()`, er bzw. sie arbeitet in genau einer Firma. Ein Angestellter hat zusätzlich eine `personalnummer` und ein `gehalt` sowie die Methode `erhoeheGehalt(betrag: double)`.

`Angestellter` und `FreierMitarbeiter` sind Spezialisierungen von Mitarbeiter; Mitarbeiter ist eine Generalisierung der beiden anderen Klassen.

Eine solche Vererbungshierarchie kann in mehreren Ebenen fortgesetzt werden, im Beispiel könnte ein `Angestellter` in die Klassen `BefristetAngestellter` und `UnbefristetAngestellter` spezialisiert werden.

Sichtbarkeit – protected

In der Regel sollen alle Attribute und Methoden vererbt werden und somit für die Subklassen sichtbar sein. Dafür gibt es eine weitere Art von Sichtbarkeit: **protected** mit dem Zeichen #.

#	**protected** / geschützt	Für Subklassen sichtbar, aber nicht für andere Klassen außerhalb der Vererbungshierarchie

Eine weitere spezielle Assoziation zwischen zwei Klassen – die Ganzes-Teile Assoziation mit der Komposition und der Aggregation

Bei einer Ganzes-Teile-Assoziation gibt es ein Ganzes, das aus mehreren Teilen besteht. Die Art wie die Teile zum Ganzen zusammengefügt sind, kann variieren. Bei einer **Komposition** ist die Lebensdauer der Teile an die Lebensdauer des Ganzen gebunden, wird das Ganze gelöscht, werden automatisch die Teile mitgelöscht. Ein Teil-Objekt gehört zu genau einem Komposit-Objekt. Das UML-Symbol ist die ausgefüllte Raute. Bei einer **Aggregation** ist die Lebensdauer der Teile nicht an die des Ganzen gebunden, wird das Ganze gelöscht, bleiben die Teile durchaus weiter bestehen. Teile können zu mehreren Aggregat-Objekten gehören. Das UML-Symbol ist die leere Raute.

Beispiel: Eine Firma besteht aus mehreren Abteilungen, zwischen der Firma und der Abteilung besteht eine Komposition. Wird die Firma liquidiert, gibt es auch keine Abteilungen mehr. Eine Firma besteht aus mehreren Gebäuden, zwischen der Firma und den Gebäuden besteht eine Aggregation. Wird die Firma liquidiert, bleiben die Gebäude weiterhin bestehen.