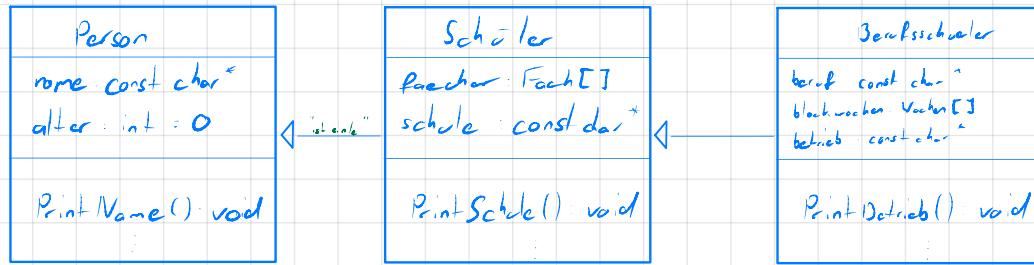


Die Symbolik der UML Modellierung Language

Statische Modelle

Klassen und Objekte stehen nicht isoliert, sondern arbeiten bei der Problemlösung zusammen

Vererbung

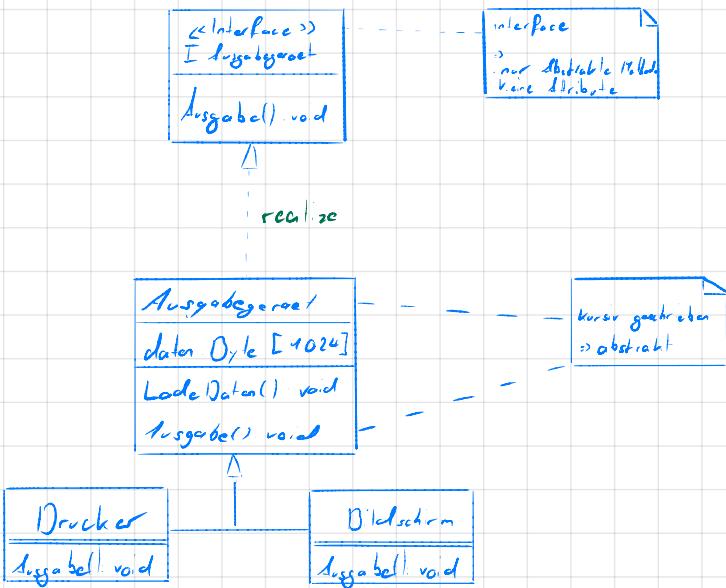


Basisklasse

abgeleitete Klasse

Durch Vererbung entsteht eine "ist ein"-Beziehung
Die abgeleitete Klasse erbt Eigenschaften und Methoden
der Basisklasse.

Abstrakte Klassen



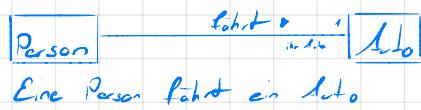
Abstrakte Klassen:

- besitzen mindestens eine abstrakte Methode
- dürfen NICHT instanziiert werden
- dienen als Vorlage zur Vererbung (→ Polymorphie)

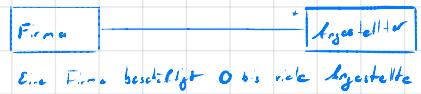
Assoziationen:

Multipizität:

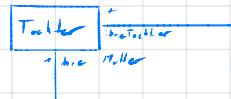
1 zu 1



1 zu n



rekursiv 1 zu 1
1 zu *



Eine Tochter hat eine Mutter, die Sibille Tochter ist

Eine Tochter hat 0 bis viele Tochter

4 ... 8

1 zu 4 bis 8

1 ... *

1 zu 1 bis viele

6, 10, 30

1 zu 6 oder 10 oder 30

n-äre Assoziationen



Ein Hemd passt zu einer Hose und einer Krawatte.

| Dei n-ären Beziehungen besteht eine Assoziation zwischen mehreren Klassen |

Assoziationsklassen



| Assoziationsklassen können Eigenschaften und Methoden definieren |

Aggregation:

Eine Klasse ist Teil einer anderen Klasse ("is part of")

Aggregation



Die Objekte können auch einzeln bestehen

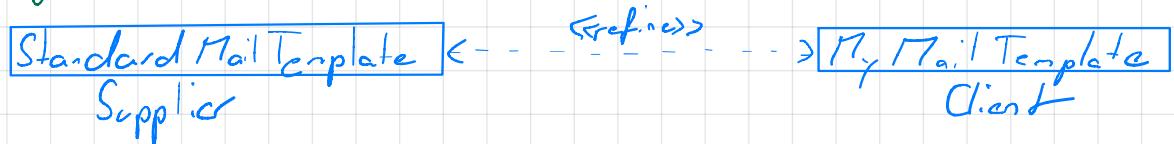
Komposition



Das enthaltene Objekt (Mail) ist vom enthaltenden Objekt (Mailbox) existenzabhängig

Bei Aggregation und Komposition wird auf die enthaltenen Objekte über das enthaltende Objekt zugriffen

Abhängigkeiten:



|| Abhängigkeiten beschreiben einer "wer kennt/nutzt wen" Beziehung. ||

Die Symbolik der Unified Modelling Language (2)

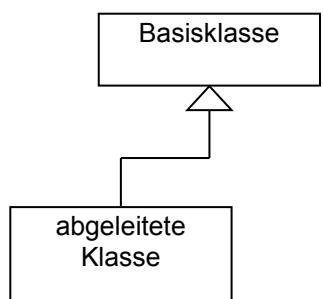
Statische Modelle in der UML

Klassen existieren in objektorientierten Programmen nicht isoliert voneinander. Damit ein Nachrichtenfluss zwischen Objekten stattfinden kann, müssen Klassen und Objekte zueinander in Beziehung stehen. Im objektorientierten Design gibt es für diese statischen Beziehungen verschiedene Arten.

Vererbung

C++ ermöglicht das Erstellen von Vererbungshierarchien zwischen Klassen. Dabei unterscheidet man zwischen Basisklasse und abgeleiteter Klasse. Die abgeleitete Klasse erbt Methoden und Attribute von der Basisklasse. Entsprechend werden Attribute und Methoden nur einmal implementiert (Wiederverwendbarkeit). Die Vererbung kann dabei auch über mehrere Ebenen erfolgen.

Vererbung



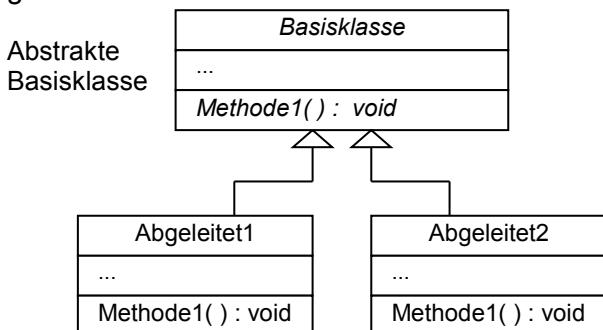
Beim Vererbungskonzept werden Basis- und abgeleitete Klassen gebildet und Beziehungen zwischen den Klassen aufgebaut. Attribute und Operationen der Basisklassen sind auch den abgeleiteten Klassen zugänglich.

Der Diskriminator (Pfeil mit nicht ausgefüllter Pfeilspitze) wird als Unterscheidungsmerkmal für die Hierarchieebenen dargestellt. Bei der Mehrfachvererbung wird die Zahl der Hierarchieebenen vergrößert. Attribute und Methoden

können so über mehrere Ebenen vererbt werden. Durch die Vererbung entsteht eine "ist ein" (is a) Beziehung zwischen Klassen. Die Pfeilrichtung gibt dabei die Richtung der Beziehung an.

Abstrakte Basisklassen

Abstrakte Basisklassen stellen eine Sonderform von Basisklassen dar. Da sie eine oder mehrere rein virtuelle Methoden nur deklarieren, nicht aber definieren, kann von ihnen keine Instanz (kein Objekt) gebildet werden.



Eine abstrakte Basisklasse ist in der UML durch kursive Schreibweise des Klassennamens dargestellt. Zusätzlich kann dem Klassennamen die Bezeichnung <>abstract<> nachgestellt werden.

Da abstrakte Klassen nicht instantiiert werden können, stellen sie Schnittstellen für abgeleitete Klassen dar.

Die eigentliche Implementierung der rein virtuellen Methoden (*kursiv*) findet dann eine (oder mehrere) Ebene(n) tiefer in der Vererbungshierarchie statt.

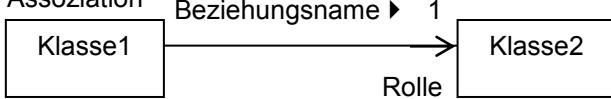
Abgeleitet1 und Abgeleitet2 definieren jeweils ihre eigene Version von Methode(). Über die Vererbung

und abstrakte Basisklassen kann das Konzept der Polymorphie in C++ umgesetzt werden.

Assoziation

Assoziationen beschreiben gemeinsame Beziehungen zwischen Klassen bzw. Objekten. Wenn ein Objekt eine Nachricht an ein weiteres Objekt abschickt, dann besteht zwischen beiden eine Beziehung. Der Rollenname einer Klasse drückt aus, welcher Aspekt der jeweiligen Klasse mit der anderen Klasse verknüpft ist.

Assoziation



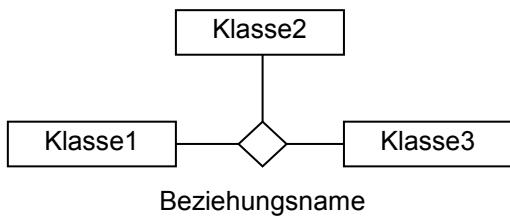
Der Beziehungsname kann mit einem ausgefüllten Pfeil für die Richtung der Beziehung versehen werden.

Die Zahl 1 am Pfeilende bezeichnet die Multiplizität der Beziehung, eine nicht geschlossene Pfeilspitze die

Navigierbarkeit (über Klasse1 kann immer auf Klasse2 zugegriffen werden. Ein Objekt der Klasse1 hält eine Beziehung zu einem Objekt vom Typ Klasse2).

Die Symbolik der Unified Modelling Language (2)

n-äre Assoziationen

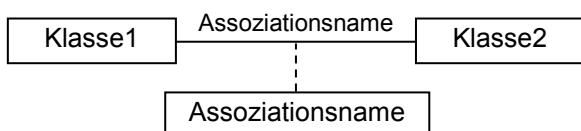


Assoziationen können auch zwischen mehr als nur zwei Klassen bestehen. In diesem Fall spricht man von n-ären Assoziationen (ternär, quarternär,...).

Die Raute in der Mitte symbolisiert die Beziehung. Bei n-ären Assoziationen darf keine Navigierbarkeit eingezeichnet werden, da automatisch alle Verknüpfungen zueinander navigierbar sind.

Assoziationsklassen

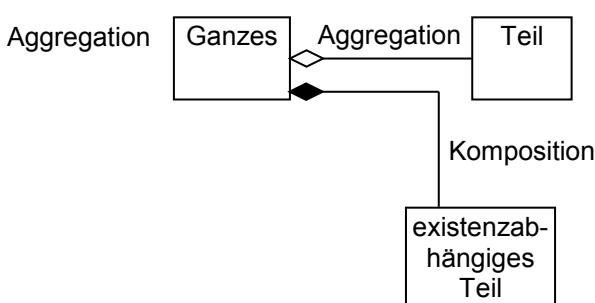
Hat ein Modellelement die Eigenschaften einer Assoziation und einer Klasse, so kann das Element als Assoziationsklasse modelliert werden. Der Name der Assoziation und der Name der Assoziationsklasse sind in diesem Fall gleich. Die Assoziationsklasse wird wie eine normale Klasse mit Eigenschaften und Methoden modelliert. Eine gestrichelte Linie verbindet die assoziative Klasse und die zugehörige Assoziation.



als Assoziationsklasse modelliert werden. Der Name der Assoziation und der Name der Assoziationsklasse sind in diesem Fall gleich. Die Assoziationsklasse wird wie eine normale Klasse mit Eigenschaften und Methoden modelliert. Eine gestrichelte Linie verbindet die assoziative Klasse und die zugehörige Assoziation.

Aggregation

Enger als bei der Assoziation sind Klassen bei der Aggregation miteinander verknüpft. Es besteht eine "ist Teil von" (is part of) Beziehung.



Eine **Aggregation** ist eine Assoziation, die berücksichtigt, dass eine Klasse Teil einer anderen Klasse ist (z. B.: Klasse „Auto“ und Klasse „Rad“). Beide Klassen können auch unabhängig von einander existieren. Die Raute deutet jeweils auf die Gesamtklasse.

Bei einer **Komposition** sind die Klassen von einander existenzabhängig. Verliert ein Objekt der enthaltenden Klasse seine Gültigkeit so wird auch das enthaltene Objekt gelöscht. Das enthaltene Objekt kann nur zu **einem** enthaltenden Objekt gehören (z.B. Baum und Blatt).

Sowohl bei der Aggregation als auch bei der Komposition kann auf die enthaltene Klasse nur über die enthaltende Klasse zugegriffen werden!

Abhängigkeiten

Abhängigkeiten kennzeichnet man durch einen Pfeil mit offener Pfeilspitze und einer gestrichelten Linie. Der Pfeil zeigt vom abhängigen auf das unabhängige Modellelement. Es entsteht eine „wer kennt wen“ Beziehung. Das abhängige Element

(Client) kennt das unabhängige (Supplier), aber nicht umgekehrt.

Abhängigkeiten können mit Hilfe von Stereotypen genauer festgelegt werden. Diese schreibt man zum Abhängigkeitspfeil.

- « use » Client gebraucht Supplier
- « abstraction » unterschiedliche Sicht oder Abstraktionsebene
- « derive » Client berechnet sich aus Supplier
- « refine » verfeinert...
- « trace » bezieht sich auf...

Abhängigkeiten dürfen in der UML nicht nur zwischen Klassen bestehen.

