Objektorientierte Systemanalyse

Teil 4: Klassendiagramme

IT works.

Agenda

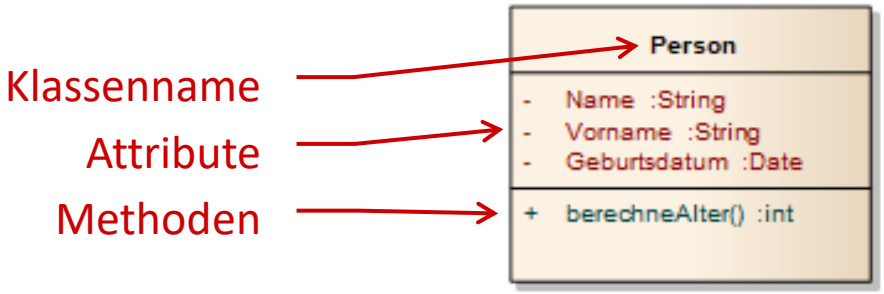
- Einführung
  - Basiskonzepte, UML
  - Fokus in der OOA
- Use Cases
  - Use Case Diagramme
  - Textschablone
- Aktivitätsdiagramme
  - Grundkonzepte
  - Erweiterungen
- **Klassendiagramme**
  - Klassendiagramme
  - Paketdiagramme, Komponentendiagramme
  - Objektmodelle, Szenarien, Zustandsautomaten
  - CRC-Karten
- UI-Design
  - Ergonomie, Usability, Gestaltung
  - Ableitung aus Klassendiagrammen
  - Beschreibung über Zustandsautomaten

Klassendiagramme

- Klassendiagramme sind Bestandteile des statischen Modells. In der Objektorientierten Analyse stellt dieses Modell die fachlichen Datenstrukturen und ihre Zusammenhänge dar. Technische Aspekte werden in der Analyse nicht berücksichtigt. Dies erfolgt erst im Objektorientierten Design.

- Das Klassendiagramm stellt die einzelnen Klassen, eventuelle Vererbungsstrukturen und die Beziehungen zwischen Klassen da.
  - Häufig wird in der Analyse auch von Business-Klassen gesprochen.
  - Modellierte Beziehungen existieren zwischen den einzelnen Instanzen (Objekten) der Klassen
  - Attribute und Methoden der Klassen werden in der Analyse häufig ohne Angabe von Datentypen verwendet. Meist erfordern jedoch die Modellierungs-Tools die Angabe eines Typs oder vergeben einen Default-Typ (z.B. int).

- Klassen-Notation in der UML an einem Beispiel



Klassen-Notation

Sichtbarkeit

Für jede Methode und jedes Attribut kann im Sinne des Information Hiding die Sichtbarkeit nach Außen festgelegt werden. Hierüber wird definiert, ob auf eine Methode oder ein Attribut von einem anderen Objekt aus zugegriffen werden kann. Die Sichtbarkeit (z.B. + für public) wird vor dem Attributs- oder Methodennamen notiert. In der Analyse ist die Sichtbarkeit eines Attributs oder einer Methode jedoch eher von untergeordneter Bedeutung.

Sichtbarkeit		
-	private	: Zugriff nur innerhalb der Klasse möglich.
+	public	: Zugriff von Außen möglich.
#	protected	: Zugriff nur innerhalb der Klasse und deren Subklassen möglich.
~	package local	: Zugriff nur für Klassen des gleichen Pakets.

Attribut- und Methodendeklaration

Für Attribute definiert die UML eine Schreibweise für die Deklaration, welche sich aus dem Namen und dem Datentyp, getrennt durch einen Doppelpunkt, zusammensetzt. Dies ist bei Methoden analog für Methodenname und Ergebnis-Typ sowie die einzelnen Parameter.

Notation für Deklarationen (allg.): Bezeichner:Typ	
- bei Attributen	: Attributname:Typ
- bei Methoden	: Methodenname(Parameter:Typ,...):Ergebnis-Typ

Klassen-Notation (Fortsetzung)

Abstrakte Klassen

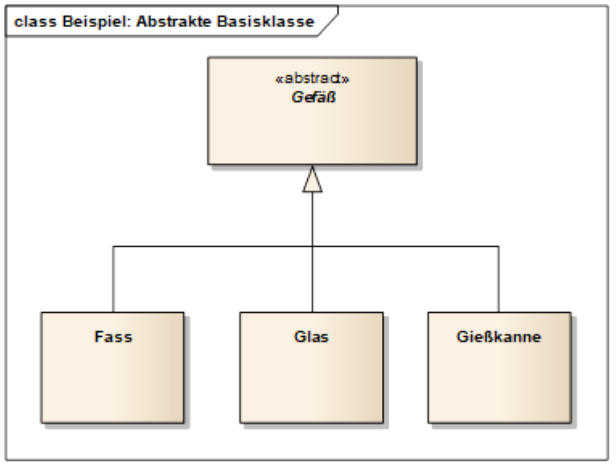
- Eine abstrakte Klasse kann nicht instanziiert werden, d.h. es können keine Objekte erzeugt werden!
- Eine abstrakte Klasse definiert Gemeinsamkeiten einer Gruppe von Unterklassen.
- Eine Klasse ist abstrakt, wenn sie als solche deklariert wird oder mindestens eine abstrakte Methode enthält.

Abstrakte Methoden

Abstrakte Methoden definieren nur die Schnittstelle einer Methode (Signatur), aber nicht deren konkrete Implementierung. Enthält eine Klasse eine abstrakte Methode, dann ist auch die Klasse abstrakt.

Abstrakte Klassen und Methoden werden durch einen **kursiv** geschriebenen Namen gekennzeichnet.

Zur besseren Kennzeichnung werden abstrakte Klassen und Methoden auch häufig mit dem Stereotyp <<abstract>> gekennzeichnet.

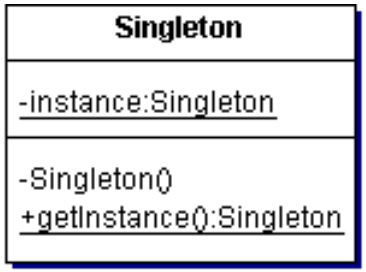


Klassen-Notation (Fortsetzung)

- Klassenattribute
  - Ein Klassenattribut liegt vor, wenn nur ein Attributwert für alle Objekte der Klasse existiert. Klassenattribute sind von der Existenz der Objekte unabhängig.
  - Klassenattribute werden nicht einer Objektinstanz zugeordnet, sondern der Klasse.
  - Sie besitzen über alle Objekte hinweg den gleichen Zustand (= Wert).

- Klassenmethode (-operation)
  - Eine Klassenmethode ist eine Operation, die für eine Klasse statt für ein Objekt der Klasse ausgeführt wird.
  - Der Aufruf einer Klassenmethode ist nicht an ein Objekt gebunden, sondern an den Klassennamen.

Klassenattribute und -methoden werden **unterstrichen**.



Vererbung

- Vererbung definiert eine neue Klasse auf Basis einer existierenden Klasse. Dabei erweitert die neue Klasse die Basisklasse um ihre speziellen Eigenschaften.

- Die neue Klasse „erbt" alle Eigenschaften der Basisklasse
  - die Attribute (Datenstruktur),
  - das Verhalten (Operationen),
  - die Beziehungen der Basisklasse

  und erweitert sie um ihre speziellen Eigenschaften.

- Die neue Klasse kann geerbtes Verhalten durch „Überschreiben" der Operationen neu definieren.

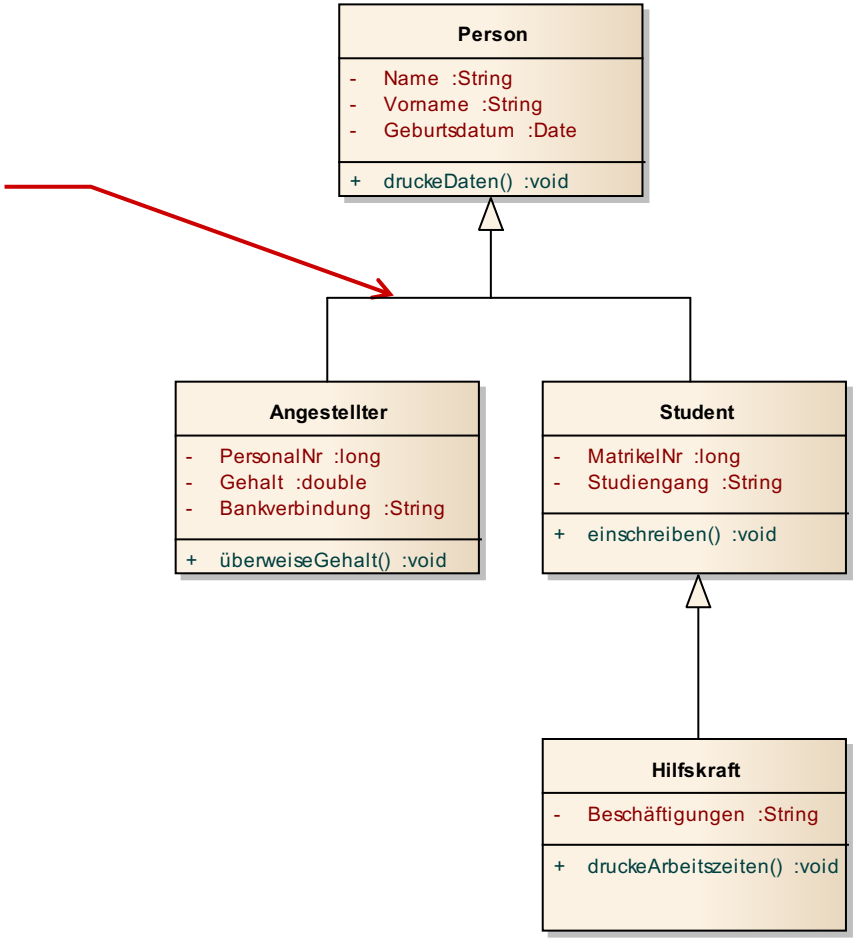
- **Definition Vererbung:**
  Die Vererbung beschreibt die Beziehung zwischen einer allgemeineren Klasse und einer spezialisierten Klasse. Die spezialisierte Klasse erweitert die Liste der Attribute, Operationen und Beziehungen der Basisklasse. Operationen der Basisklasse dürfen redefiniert werden.
  Es entsteht eine Klassenhierarchie oder Vererbungsstruktur.

- Synonyme:
  - Oberklasse = Superklasse = Basisklasse: allgemeinere Klasse (Generalisierung)
  - Unterklasse = Subklasse: spezialisierte Klasse (Spezialisierung)

Vererbung

- Darstellung einer Vererbungshierarchie

Vererbungspfeil zeigt zur Basisklasse (von der Spezialisierung zur Generalisierung)



Übung zu Klassendiagrammen

Aufgabe 1

Gegeben sind die Klassen Windgetriebenes Fahrzeug, Segelboot, Motorgetriebenes Fahrzeug, Wasserfahrzeug und Fahrzeug.

Erstellen Sie ein Klassendiagramm, in dem als Beziehung nur Vererbung verwendet wird und geben Sie pro Klasse zwei Attribute und zwei Methoden an.

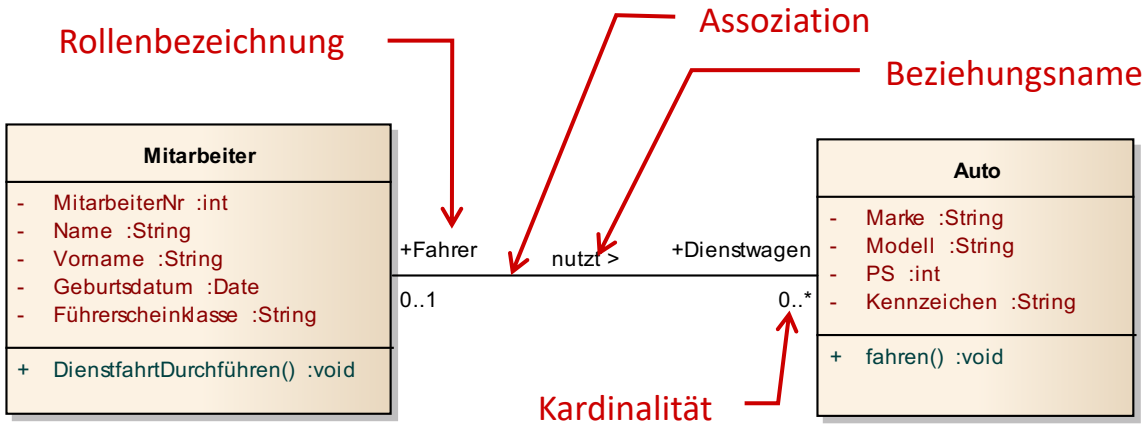
Assoziationen

- Klassen und ihre Objekte existieren nicht isoliert. Assoziationen modellieren Beziehungen zwischen den Objekten einer oder mehrerer Klassen. Beispiele:
  - Beziehungen zwischen Personen, die sich gegenseitig kennen (=Beziehungen zwischen Objekten, die alle zu einer Klasse gehören)
  - Beziehungen zwischen einer Buchung, ihrem Auftraggeber und dessen Adresse oder Beziehungen eines Auftrags, der Auftragspositionen und der Ware je Auftragsposition (= Beziehungen zwischen Objekten unterschiedlicher Klassen)

- Assoziationen sind i.d.R. binär (d.h. sie bestehen zwischen zwei Objekten); es sind aber auch höherwertige Assoziationen möglich und modellierbar.

- Notation der Assoziation
  - Notation im Klassendiagramm als Linie zwischen einer oder zwei Klassen.
  - Eine Assoziation wird zusätzlich näher spezifiziert über:
    - Kardinalität: Auf jeder Seite der Assoziation muss angegeben werden, zu wie vielen Objekten der Klasse die Beziehung besteht.
    - Beziehungsname: Eine Assoziation kann benannt werden, um die Semantik der Beziehung zu beschreiben.
    - Rolle: Auf jeder Seite der Beziehung kann ein sogenannter Rollenname angegeben werden, der die individuelle Sicht auf die Objekte der Klasse beschreibt.

Notation der Assoziation



Ein Mitarbeiter nutzt keinen, einen oder mehrere Dienstwagen.
Ein Auto wird von keinem oder höchstens einem Fahrer gefahren.

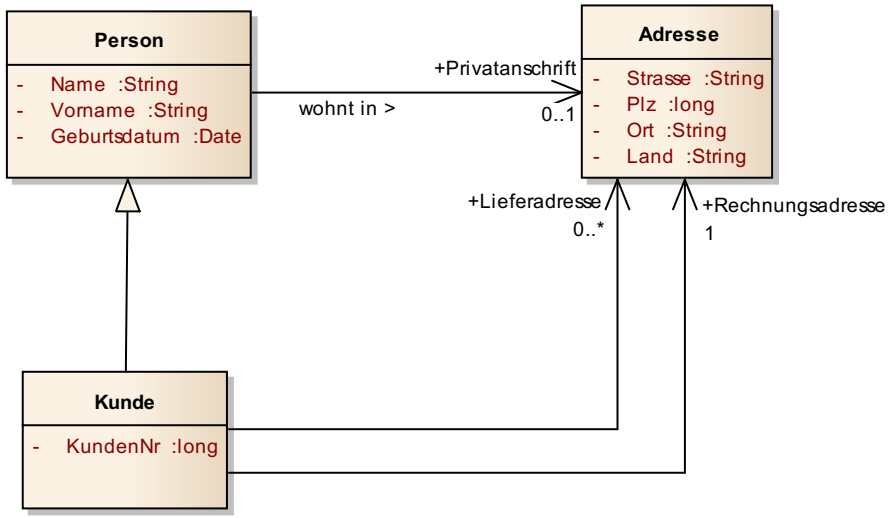
Notation der Kardinalität:

0..1	Kann-Beziehung zu einem anderen Objekt
0..*	Kann-Beziehung zu mehreren Objekten (oder nur *)
1	Muss-Beziehung zu genau einem anderen Objekt
1..*	Muss-Beziehung zu mehreren Objekten

Diese sind die geläufigsten Wertebereiche. Es sind jedoch beliebige Bereiche möglich, z.B. 2..5 oder 0..10.

Gerichtete Assoziationen

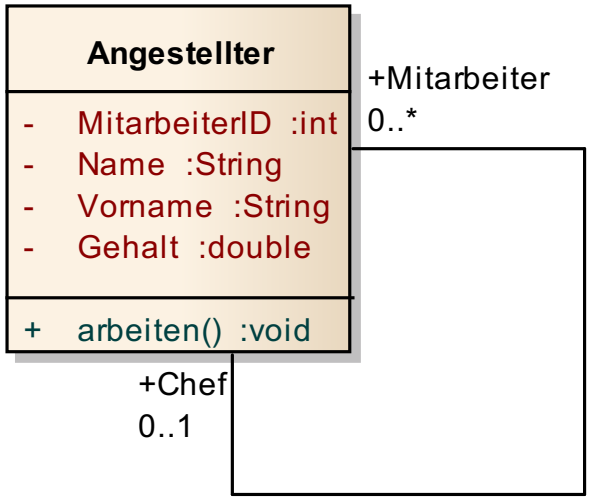
- Gerichtete Assoziationen sind nur in eine Richtung navigierbar. Das bedeutet: Ein Objekt kennt das andere, aber nicht umgekehrt. Und nur diese Richtung wird auch implementiert.



Hinweise:
- In der Praxis sind viele Beziehungen gerichtete Assoziationen.
- Beziehungen sind Eigenschaften einer Klasse und werden vererbt, d.h. in diesem Beispiel erbt die Klasse Kunde die Beziehung der Klasse Person zur Klasse Adresse – sprich: Auch ein Kunde hat eine Privatanschrift.

Reflexive Beziehungen

- Hierunter sind Beziehungen zwischen Objekten derselben Klasse zu verstehen. Beispiele:
  - Verheiratete Personen,
  - Stücklisten-Strukturen,
  - oder wie im modellierten Beispiel, eine Hierarchie von Angestellten.

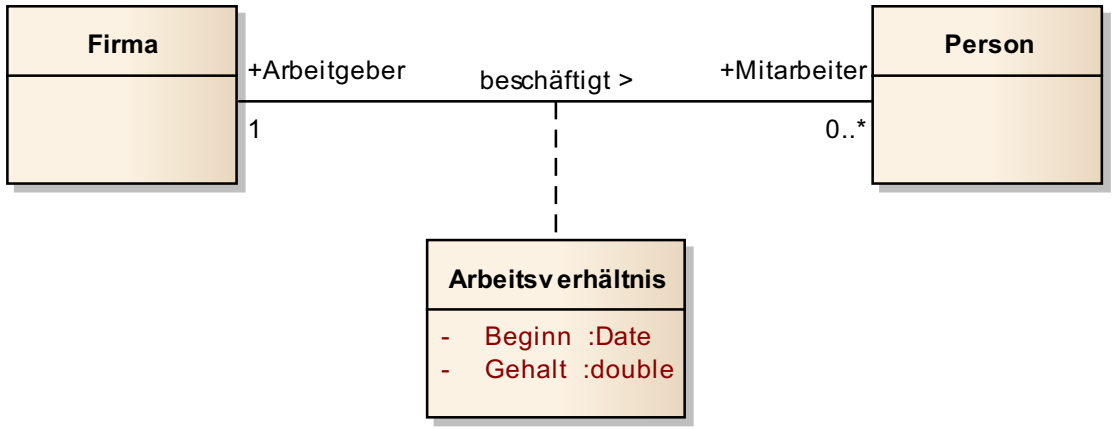


Attributierte Assoziationen

- Assoziationen können selbst Eigenschaften einer Klasse besitzen, d.h. selbst durch
  - Attribute und Operationen
  - Beziehungen zu anderen Klassen

  charakterisiert werden.

- Die UML-Notation dokumentiert dies durch eine Klasse, welche mit der Assoziation durch eine gestrichelte Linie verbunden ist.

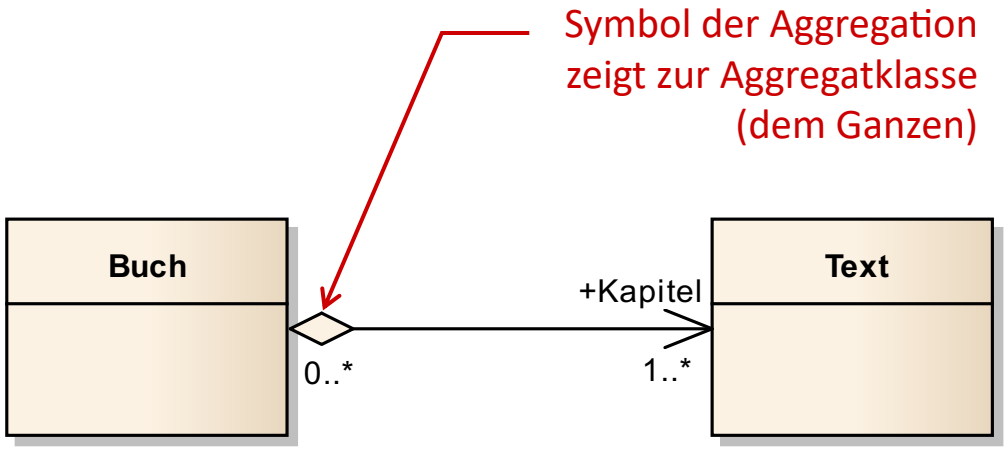


Attributierte Assoziationen müssen im Design immer aufgelöst werden.

Aggregation

- Eine Aggregation ist eine besondere Form der Assoziation.
  - Die beteiligten Klassen sind nicht gleichrangig, sondern haben zueinander eine Ganzes-Teile-Beziehung.
  - Die Teil-Objekte einer Aggregation sind allein existenzfähig.
  - Man spricht auch von einer shared-aggregation, d.h. ein Teil-Objekt kann auch mehreren Aggregatklassen zugeordnet werden.

- UML-Notation: Ergänzung der Assoziation um eine leere Raute auf der Seite des Aggregats.

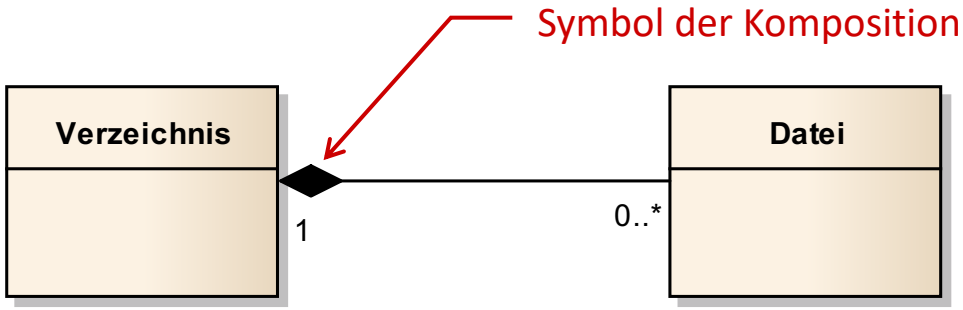


Komposition

- Eine Komposition ist eine strengere Form der Aggregation, bei der die Teile allein nicht existenzfähig sind, d.h. das Löschen des Ganzen bewirkt auch ein Löschen der Teile.
  - Jedes Objekt der Teilklasse kann zu einem Zeitpunkt nur einem Objekt der Aggregatklasse zugeordnet sein (unshared aggregation, strong ownership). Das Teil darf jedoch auch einem anderen Ganzen zugeordnet werden.
  - Die dynamische Semantik des Ganzen gilt auch für seine Teile (Bsp.: Wird das Ganze kopiert, werden auch die Teile kopiert).

- UML-Notation: Ergänzung der Assoziation um eine gefüllte Raute auf der Seite des Aggregats.

- Problem: Die Frage nach Aggregation oder Komposition kann nicht immer eindeutig beantwortet werden. Häufig liegt jedoch eine Komposition vor.

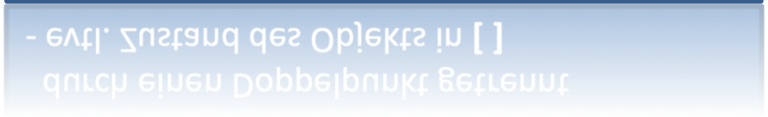


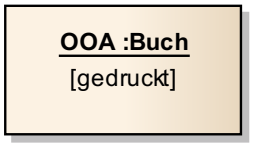
Objektdiagramm

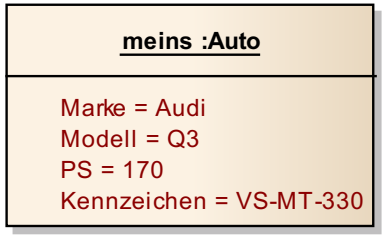
- Das Objektdiagramm stellt Objekte und ihre Verbindungen untereinander dar.
- Objektdiagramme modellieren einen Ausschnitt des Systems zu einem bestimmten Zeitpunkt.
- Objekte können einen Namen besitzen oder es können anonyme Objekte sein.

Notation der Objekte:

- konkret mit Objektname und Klassenname
- anonymes Objekt, d.h. lediglich Klassenname
- Objekt- bzw. Klassenname **unterstrichen** und durch einen Doppelpunkt getrennt
- evtl. Zustand des Objekts in **[ ]**

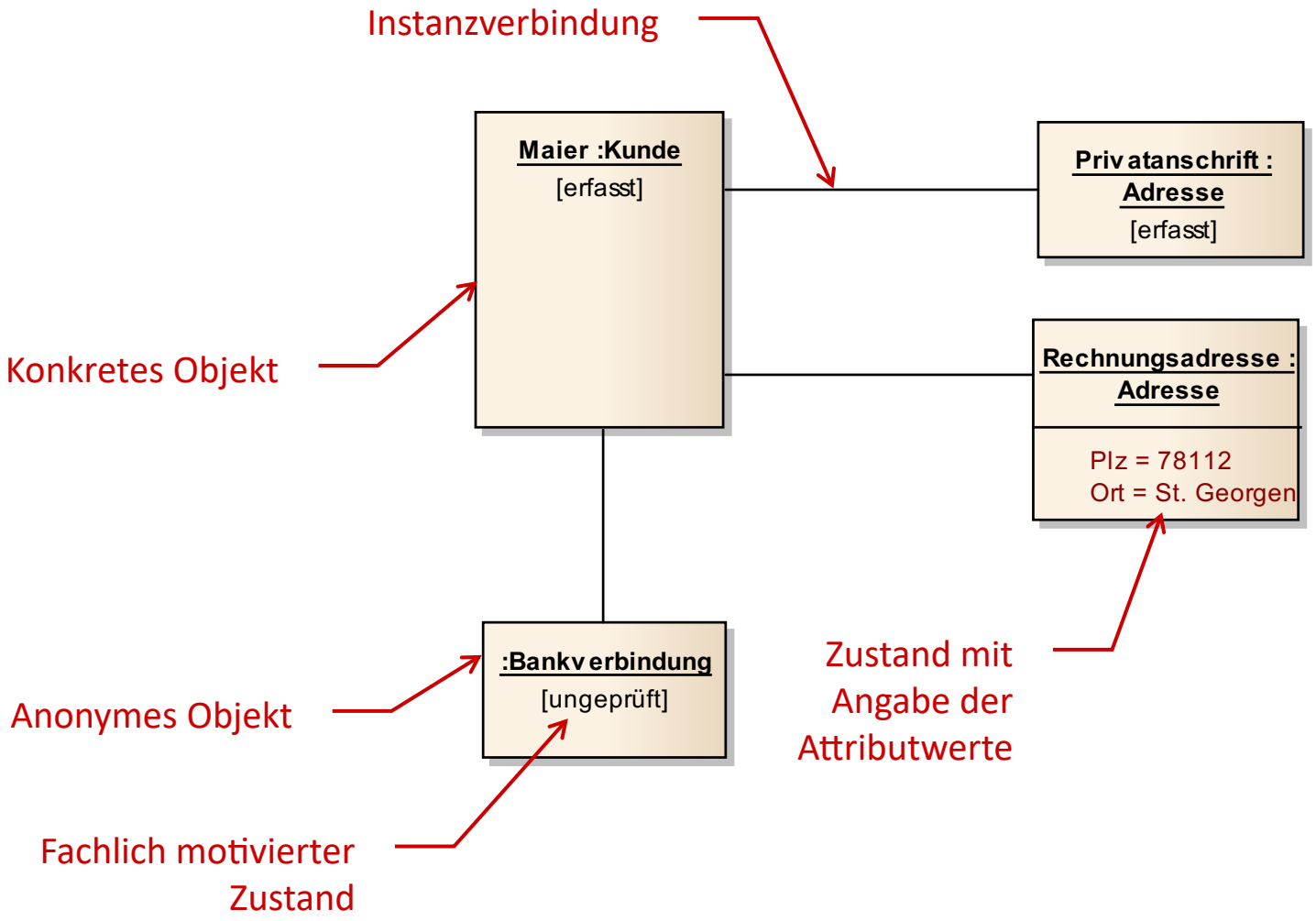






Objektdiagramm

- Beispiel

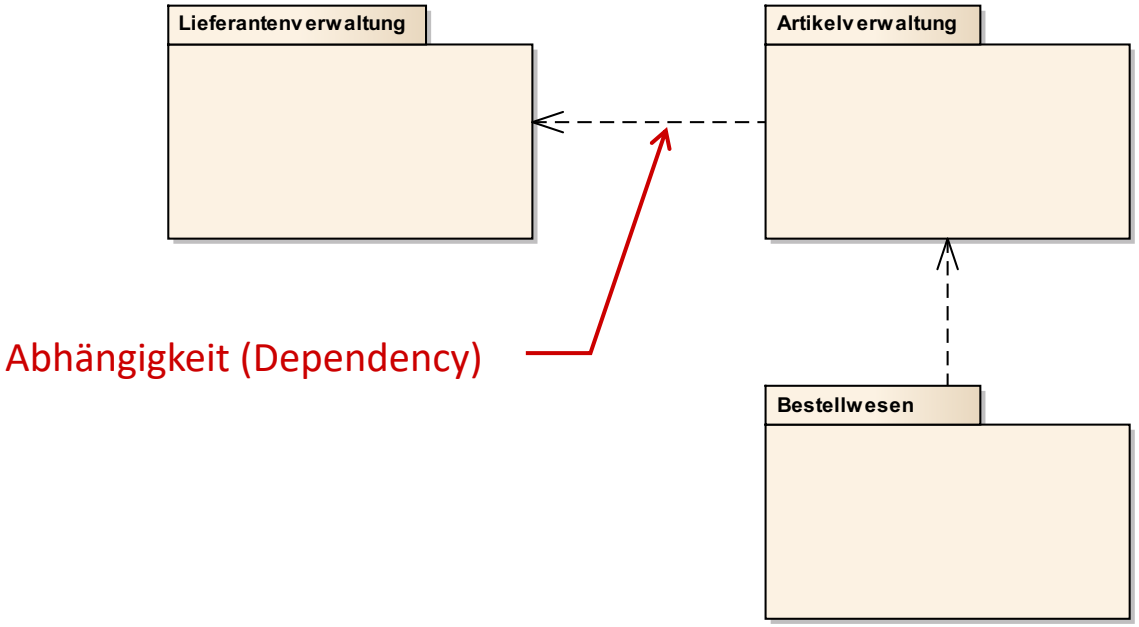


Paketdiagramm

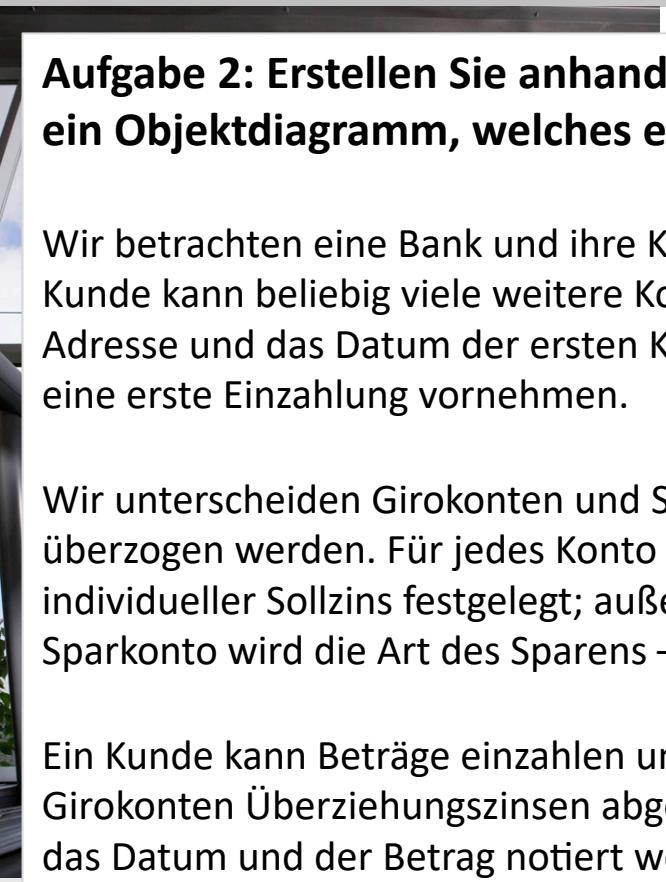
- Pakete dienen der logischen Gruppierung von Klassen
  - Gruppierungsmerkmale sind in der Analysephase des Projekts fachlich motiviert und gruppieren die Klassen nach Teilgebieten des Problembereichs (z.B. Bestellwesen, Artikelverwaltung, Partner)
  - In der Designphase des Projekts wird häufig die Architektur / Schichtenzuordnung oder andere technische Einflüsse die Bildung von Paketen beeinflussen (z.B. Client, Server, Gui, Controller, Modell)
- Pakete können selbst wieder weitere Pakete beinhalten.
- Klassen sind immer eindeutig einem Paket zugeordnet.
- Abhängigkeiten zwischen den Paketen sind sorgfältig zu beobachten. Da Klassen i.d.R. nicht isoliert existieren, sondern mit anderen Klassen in Beziehung stehen, ergeben sich zwangläufig Abhängigkeiten zwischen den Paketen, die diese Klassen enthalten.

Paketdiagramm

- Paketdiagramme zeigen die Paketstruktur
- Paketdiagramme zeigen die Abhängigkeiten zwischen Paketen, d.h. welche Auswirkung hat eine Änderung an der Definition / dem Inhalt eines Pakets?
- Paketdiagramme werden i.d.R. nicht als eigenständige Diagrammform betrachtet. Pakete werden als eigene Elemente im Klassendiagramm notiert.



Übungen zu Klassendiagrammen



Aufgabe 2: Erstellen Sie anhand der folgenden Problemstellung ein Klassendiagramm sowie ein Objektdiagramm, welches eine beispielhafte Ausprägung des Klassendiagramms zeigt.

Wir betrachten eine Bank und ihre Kunden. Eine Person wird Kunde, wenn sie ein Konto eröffnet. Ein Kunde kann beliebig viele weitere Konten eröffnen. Für jeden neuen Kunden werden dessen Name, Adresse und das Datum der ersten Kontoeröffnung erfasst. Bei der Kontoeröffnung muss der Kunde gleich eine erste Einzahlung vornehmen.

Wir unterscheiden Girokonten und Sparkonten. Girokonten dürfen bis zu einem bestimmten Betrag überzogen werden. Für jedes Konto wird ein individueller Habenzins, für Girokonten auch ein individueller Sollzins festgelegt; außerdem besitzt jedes Konto eine eindeutige Kontonummer. Für jedes Sparkonto wird die Art des Sparens – z.B. Festgeld – gespeichert.

Ein Kunde kann Beträge einzahlen und abheben. Des Weiteren werden Zinsen gutgeschrieben und bei Girokonten Überziehungszinsen abgebucht. Um die Zinsen zu berechnen, muss für jede Kontobewegung das Datum und der Betrag notiert werden. Die Gutschrift bzw. Abbuchung der Zinsen erfolgt bei den Sparkonten jährlich und bei den Girokonten quartalsweise.

Ein Kunde kann jedes seiner Konten wieder auflösen. Bei der Auflösung des letzten Kontos hört er auf, Kunde zu sein.

Tipps zur Modellerstellung: Klassen

- Alle Gegenstände, Personen, etc. die im Zusammenhang zum Problembereich stehen, sind potentielle Kandidaten für Klassen.
- Aus den Anforderungen müssen die wichtigen Klassen (Businessklassen) identifiziert werden. Diese Businessklassen sollen für alle am Entwicklungsprozess beteiligten verständlich sein (Kunde, Entwickler, Manager, etc.).
- Mögliche Kandidaten für Klassen:
  - Personen bzw. ihre Rollen, Orte, sichtbare / greifbare Dinge (Auto, Buch)
  - Schnittstellen (Drucker, Scanner)
  - Organisationen (Öffentliche Einrichtungen, Filialen)
  - Ereignisse (Reservierung, Buchung, Abflug)
  - Beziehungen zwischen Objekten (Vertrag)
  - Informationen über Aktionen (Bestellung, Überweisung)
  - Formulare (Umfragebögen, Bewerbungsbögen)

Tipps zur Modellerstellung: Attribute

- Hier ist zunächst abzugrenzen, ob die identifizierte Eigenschaft einer Klasse überhaupt systemrelevant ist.

- Anhaltspunkte für die Einordnung von Attributen
  - Zuordnung eines Attributes zu einer Klasse, wenn die Informationen des Attributs semantisch zur Klasse gehört.
  - Zuordnung eines Attributes zu einer Beziehung, wenn es ein Merkmal einer Beziehung beschreibt (beim Design muss überlegt werden, wie dieses Attribut in das Designmodell einbezogen werden kann)
  - Attributnamen sollen selbsterklärend gewählt werden.

Tipps zur Modellerstellung: Methoden

- Für jedes Ereignis (von außen oder von innerhalb des Systems) muss eine Operation vorhanden sein, die auf dieses Ereignis reagiert.

- Zuordnung von Operationen zu Klassen
  - Assoziationen zwischen Klassen sind vorhanden, weil Objekte dieser Klassen Botschaften austauschen. Welche Klasse muss welchen Dienst erbringen?
  - Zu welcher Klasse gehören die Attribute, die von der Operation betroffen sind?

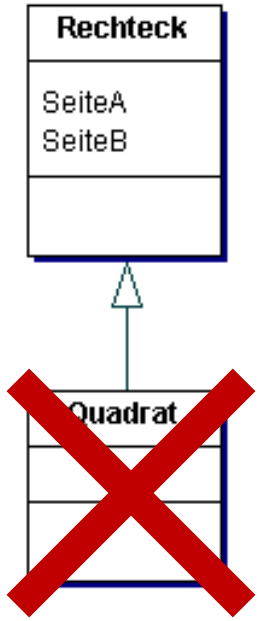
Tipps zur Modellerstellung: Beziehungen zwischen Klassen

- Assoziationen können durch die in der Problembeschreibung verwendeten Verben gefunden werden, die die Beziehung von Gegenständen, Personen, etc. im Problemraum ausdrücken.

- Anhaltspunkte für das Finden der Art und Eigenschaften von Assoziationen
  - Müssen die Objekte zweier Klassen miteinander kommunizieren?
  - Hängen Datenstrukturen voneinander ab? (Lieferant eines Warenartikels, Auftraggeber einer Bestellung)
  - Liegt eine Muss- oder eine Kann-Beziehung vor (Auswirkung auf die Kardinalität)?
  - Sind die Beteiligten gleichrangig (Assoziation oder Aggregation)? Im Zweifelsfall Assoziation wählen.
  - Erfolgt der Zugriff auf einzelne Teile ausschließlich über das Ganze (Aggregation oder Komposition)?
  - Wird eine Operation auf Teilobjekte weitergegeben, z.B. kopiere Seite, kopiere Zeilen (Aggregation oder Komposition)?

Tipps zur Modellerstellung: Klassenhierarchien

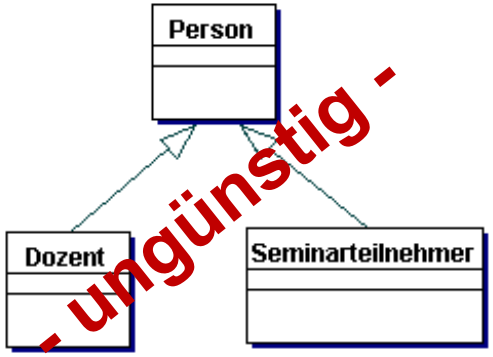
- Eine Generalisierung und Spezialisierung erfolgt nicht immer allein zur Optimierung und Redundanzvermeidung, sondern vor allem durch die angenommene Semantik.
- Besteht von einer Klasse B eine Ist-Ein-Beziehung zu einer Klasse A, so ist dies ein Indiz für eine Generalisierung (A als Oberklasse und B als Unterklasse).
- Klasse B besitzt zusätzliche Attribute und / oder Operationen.

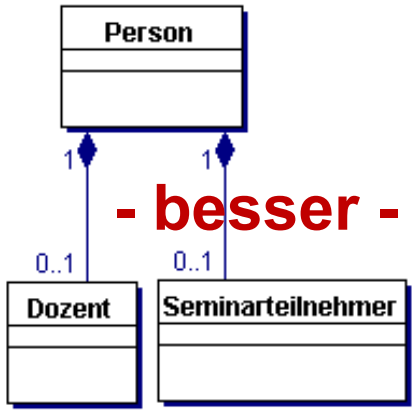


Hinweis:
Es ist immer zu prüfen, ob die Ist-Ein-Beziehung bereits so stark ist, dass eigentlich keine neue Klasse mehr eingeführt werden muss (Beispiel Rechteck und Quadrat).

Tipps zur Modellerstellung: Klassenhierarchien

Hinweis:
Es ist stets zu prüfen, ob die Spezialisierung nur durch die Aussage „ist ein“ zu beschreiben ist. Spezialisierungen, welche besser durch die Aussage „ist eine Rolle, in der die Basisklasse auftritt“ zu beschreiben sind, sollten nicht durch Vererbung, sondern besser durch Assoziation bzw. Komposition modelliert werden.



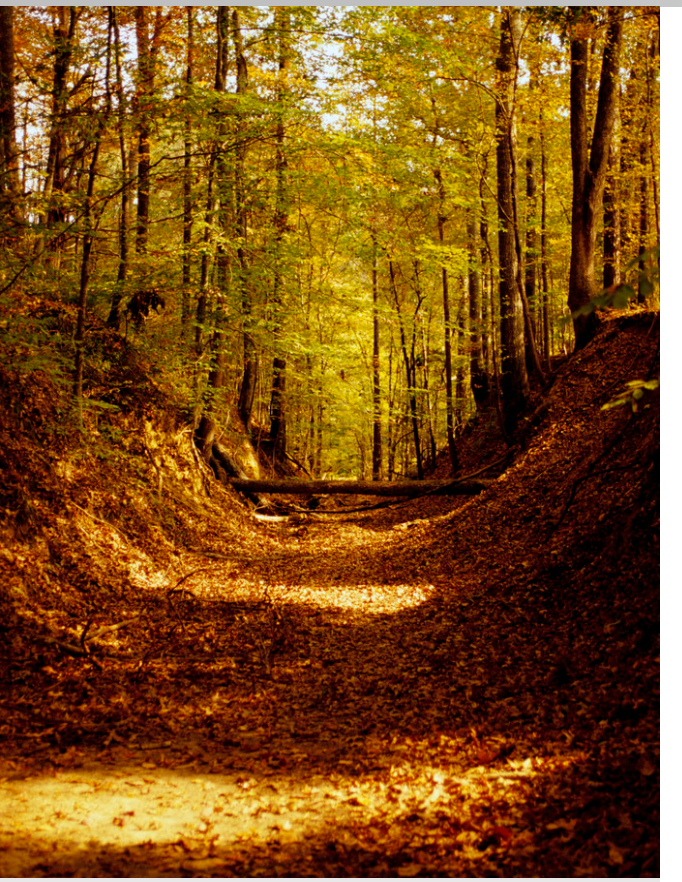


Tipps zur Modellerstellung: Klassenhierarchien

Kriterien für den Einsatz der Vererbung

- Die Spezialisierung kann mit der Formulierung "ist ein" beschrieben werden. Es ist keine Formulierung der Art "x tritt in der Rolle y auf" oder "x benutzt/besitzt y" möglich.
- Ein Objekt der Klasse muss niemals in ein Objekt einer anderen Klasse innerhalb der Vererbungshierarchie umgewandelt werden.
- Es werden die Eigenschaften der Basisklasse hauptsächlich erweitert und nicht nur überschrieben, ignoriert oder nicht benutzt (null-gesetzt). Oder anders formuliert: die Spezialisierung benötigt alle Eigenschaften der Basisklasse und definiert nur zusätzliche Attribute und Methoden.
- Es handelt sich innerhalb des Problembereichs um Klassen zur Abbildung von Rollen (oder Strategien), Transaktions-informationen oder Informationen zu Gegenständen bzw. Sachen.
- Utility-Klassen, d.h. Klassen mit nützlicher Funktionalität, die wiederverwendet werden kann, sollten niemals abgeleitet werden.

Checkliste: Schritte zum Statischen Modell



(1) Klassen identifizieren
Identifizieren Sie für jede Klasse nur so viele Attribute und Operationen, wie für das Problemverständnis und das einwandfreie Identifizieren der Klasse notwendig sind.

(2) Assoziationen identifizieren
Tragen Sie zunächst nur die reinen Verbindungen ein, d.h. machen Sie noch keine genaueren Angaben zu Kardinalität oder Art der Assoziation

(3) Attribute identifizieren
Identifizieren Sie alle Attribute des Fachkonzepts.

(4) Vererbungsstrukturen identifizieren
Erstellen Sie aufgrund der identifizierten Attribute Vererbungsstrukturen. Beachten Sie, dass Vererbung stets der Erweiterung dienen und nicht überschreiben/löschen soll.

(5) Assoziationen vervollständigen
Treffen Sie die endgültige Festlegung, ob eine „normale" Assoziation, Aggregation oder Komposition vorliegt und spezifizieren Sie die Beziehungen vollständig mit Kardinalität, Rollenname, evtl. Beziehungsname und Constraints.

(6) Attribute spezifizieren
Erstellen Sie für alle identifizierten Attribute eine vollständige Spezifikation

Seminaranmeldung

Als Teilnehmer zu den genannten Seminaren wird angemeldet:

Name, Vorname: _____, _____

Seminar-Nr.	Seminar-Bezeichnung	vom – bis
_____	_____	_____
_____	_____	_____
_____	_____	_____

Anmeldebestätigung/Rechnung senden an:

Name, Vorname: _____

Firma: _____

Adresse: _____

Datum, Unterschrift: _____, _____

Aufgabe 3:
Analysieren Sie das nebenstehende Formular und leiten Sie daraus Klassen sowie deren Attribute ab. Stellen Sie diese Klassen mit ihren Beziehungen in einem Klassen-diagramm dar.

Vielen Dank für Ihre Aufmerksamkeit. Haben Sie noch Fragen?

IT works.

Kontakt

Dr. Nikolaus Mairon

klaus@mairon-online.de
Tel. +49 (0) 160 96678776
Skype-ID: klaus@mairon-online.de

msg nexinsure ag
Geschäftsstelle St. Georgen
Leopoldstr. 1
78112 St. Georgen

nikolaus.mairon@msg.group
Tel. +49 (0) 89 96101-3004
Mobil +49 (0) 171 9716462