Fachhochschule
Dortmund
University of Applied Sciences and Arts

Softwaretechnik 1(A)

UML-Statisches Modell

Autorin: Prof. Dr. Sabine Sachweh

UML (Unified Modeling Language)

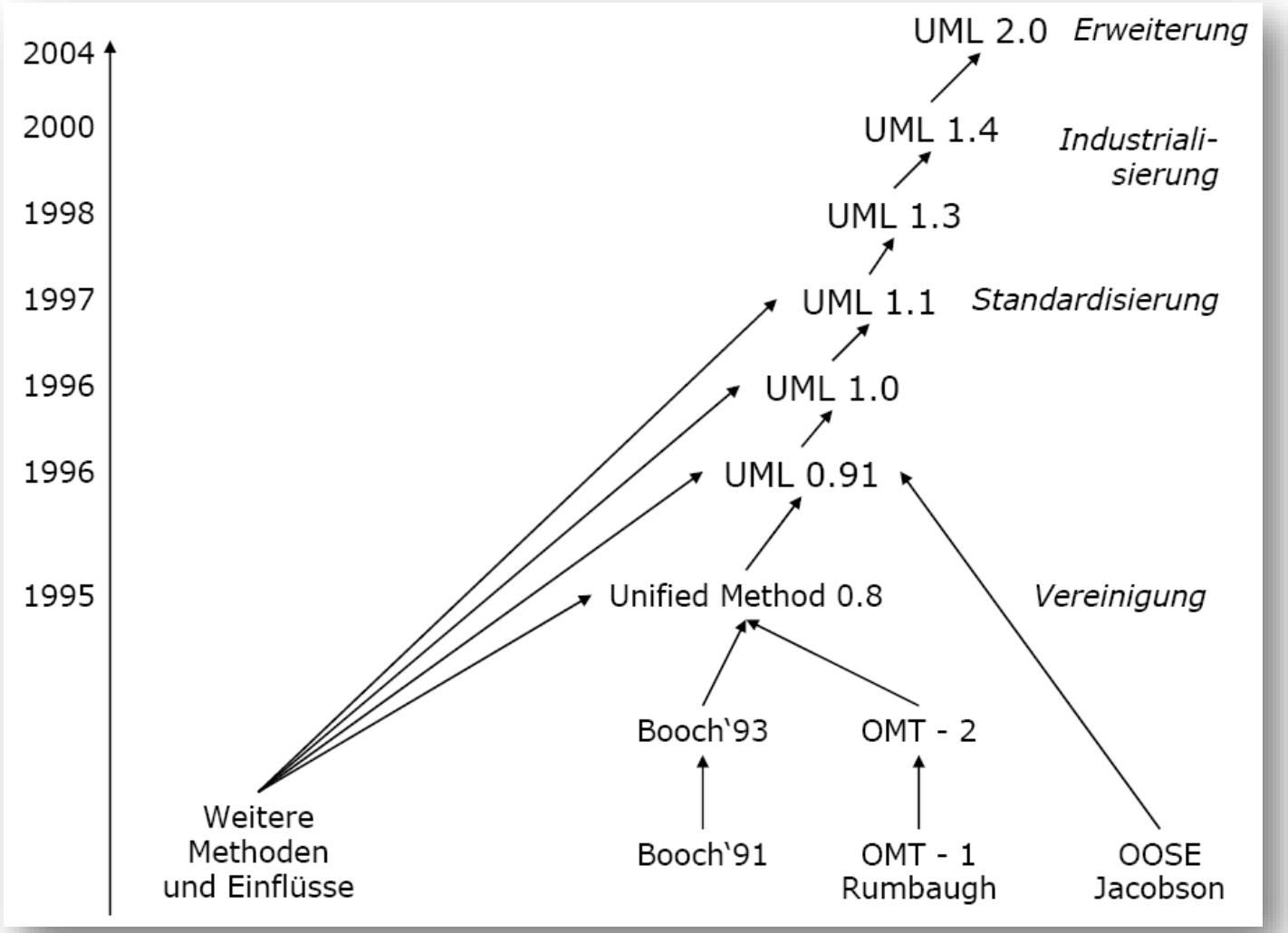
- standardisierte Notationssprache um Softwaresysteme und –projekte besser
  - analysieren
  - entwerfen
  - dokumentieren zu können
- (überwiegend) grafische Notation
- Mittel zur Dokumentation und Kommunikation zwischen Entwicklern und teilweise mit dem Kunden
- für Modellierung statischer und dynamischer Aspekte
- nicht fest mit einem bestimmten Entwicklungsprozess verknüpft
- der UML wird durch die OMG voran getrieben

Vorteile

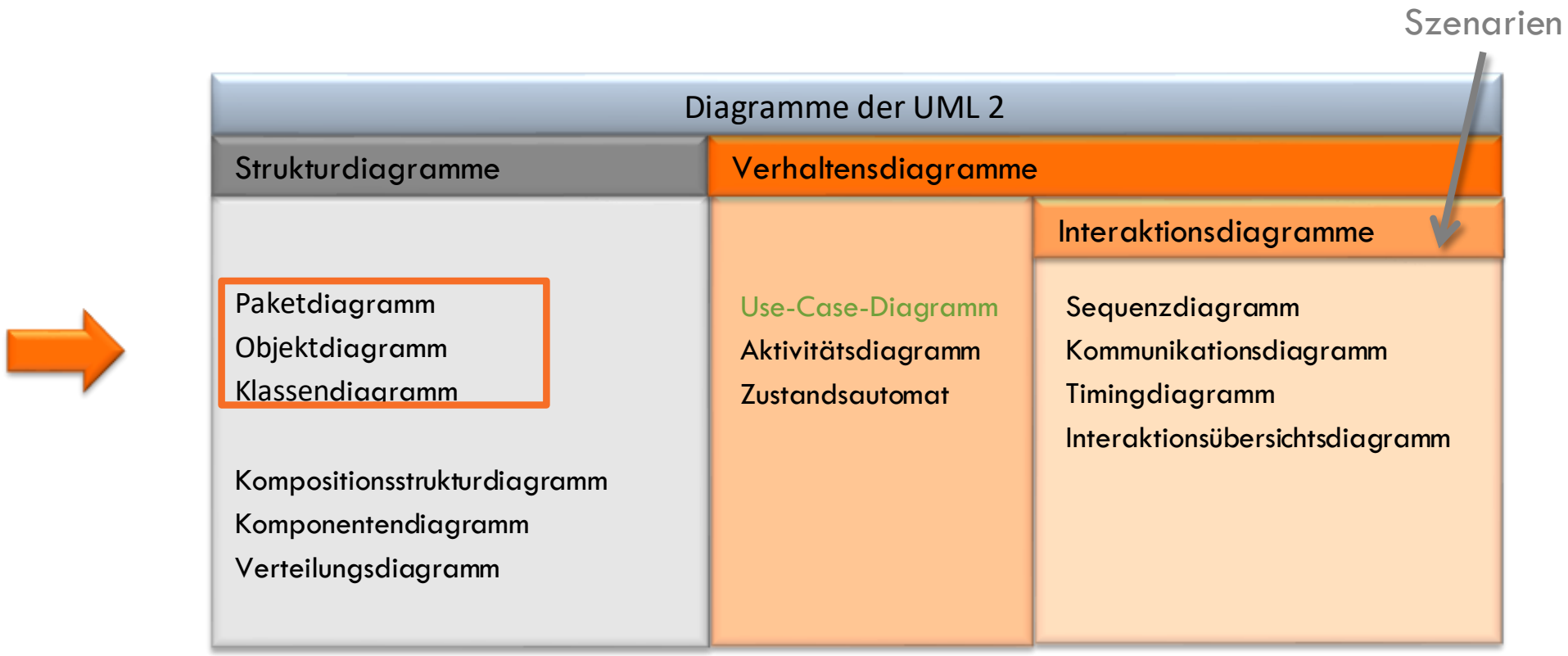
Vorteile einer vereinheitlichten Modellierungssprache

- Anforderungen an eine Modellierungssprache
  - Ausdrucksstärke, Anwendbarkeit, Eindeutigkeit, Toolunterstützung
- Vorteile einer einheitlichen Modellierungssprache
  - bessere Einarbeitung neuer Mitarbeiter
  - bessere Lesbarkeit der Dokumentation

UML - Historie



UML – Diagrammtypen der UML 2



Agenda

- Knoten
  - Paket
  - Objekt
  - Klasse
  - Attribut
- Assoziationen
  - Basis: Assoziationsname, Kardinalität/Multiplizität, Rollenname
  - Richtung
  - Eigenschaftswerte
  - Einschränkungen
  - Assoziationsklasse
  - Qualifizierte Assoziation
  - Abgeleitete Assoziation
  - Arten von Assoziationen
  - Höherwertige Assoziationen
- Generalisierung

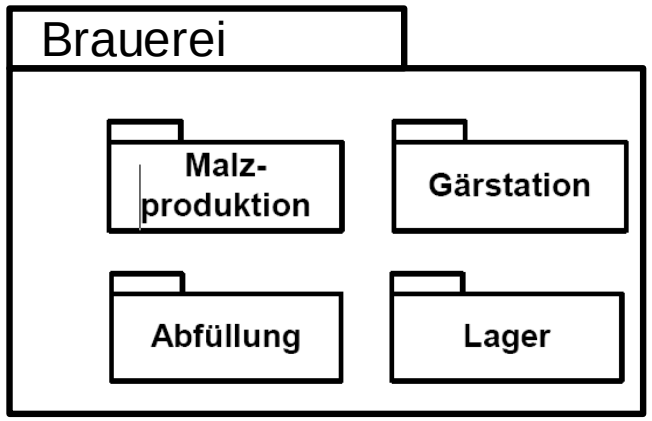
Teilsystem/Paket

Definition

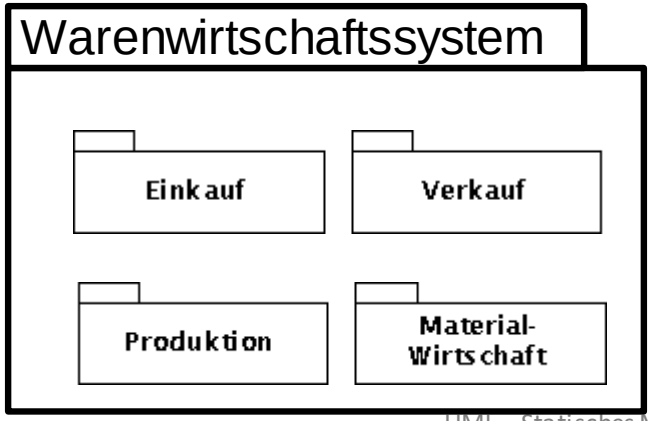
Ein Paket *(package)* fasst Modellelemente (z.B. Use Cases oder Klassen) zusammen.

- Pakete schaffen eine bessere Übersicht über ein großes Modell
- ein Paket kann selbst Pakete enthalten
- Beschreibung der Systemstruktur auf einer hohen Abstraktionsebene
- das vollständige System kann als ein großes Paket aufgefasst werden

Beispiel: Brauerei



Beispiel: Warenwirtschaftssystem



Teilsystem/Paket - Notation

Paket

- Paketname muss im gesamten System eindeutig sein
  ⇒ ein Paket definiert einen Namensraum für alle enthaltenen Elemente

Paket und Elemente

- jede Klasse (allgemeiner: jedes Modellelement) gehört zu höchstens einem Paket
- es kann in mehreren anderen Paketen darauf verwiesen werden
  - Verweis von außerhalb des Pakets
    `Paket::Element` oder `Paket1::Paket11::Paket111::Element`

Agenda

- Knoten
  - Paket
  - Objekt
  - Klasse
  - Attribut
- Assoziationen
  - Basis: Assoziationsname, Kardinalität/Multiplizität, Rollenname
  - Richtung
  - Eigenschaftswerte
  - Einschränkungen
  - Assoziationsklasse
  - Qualifizierte Assoziation
  - Abgeleitete Assoziation
  - Arten von Assoziationen
  - Höherwertige Assoziationen
- Generalisierung

Objekt

Definition

Ein Objekt ist allgemein ein Gegenstand des Interesses bei Beobachtungen und Untersuchungen.

Ein Objekt (*object*) ...

- besitzt einen bestimmten **Zustand** → Attributwerte
- reagiert mit definiertem **Verhalten** auf seine Umgebung → Operationen
- besitzt eine **Identität,** die es von allen anderen Objekten unterscheidet
- kann **Beziehungen** zu anderen Objekten haben

Objekt - Eigenschaften

- Wiederholung: Objekt-Eigenschaften sind Identität, Zustand, Verhalten und Beziehungen zu anderen Objekten
- Der **Zustand** (*state*) eines Objektes:
  - Attribute bzw. deren aktuelle Werte
    - Attribute (→ sind unveränderlich)
    - Attributwerte (→ sind veränderlich)
  - jeweilige Verbindungen zu anderen Objekten
  - ⇨ alle Daten, die ein Objekt beinhaltet
- Das **Verhalten** (*behavior*) eines Objekts:
  - Menge der Operationen (→ Methoden)
  - Änderung oder Abfrage des Zustandes nur mittels Operationen möglich (→ Geheimnisprinzip)

Objekt – Identität vs. Gleicheit

Objektidentitäts-Prinzip

Jedes Objekt ist per Definition unabhängig von seinen konkreten Attributwerten von allen anderen Objekten eindeutig zu unterscheiden.

- **Objektidentität** *(identity)*
  - Unterscheidet ein Objekt von allen anderen Objekten
  - Kann sich nicht ändern
  - Keine zwei Objekte besitzen dieselbe Identität (auch wenn sie zufällig die gleichen Attributwerte haben)
  - ⇨ **Objektidentität hat keinen inhaltlichen Bezug zu den anderen Eigenschaften des Objekts**

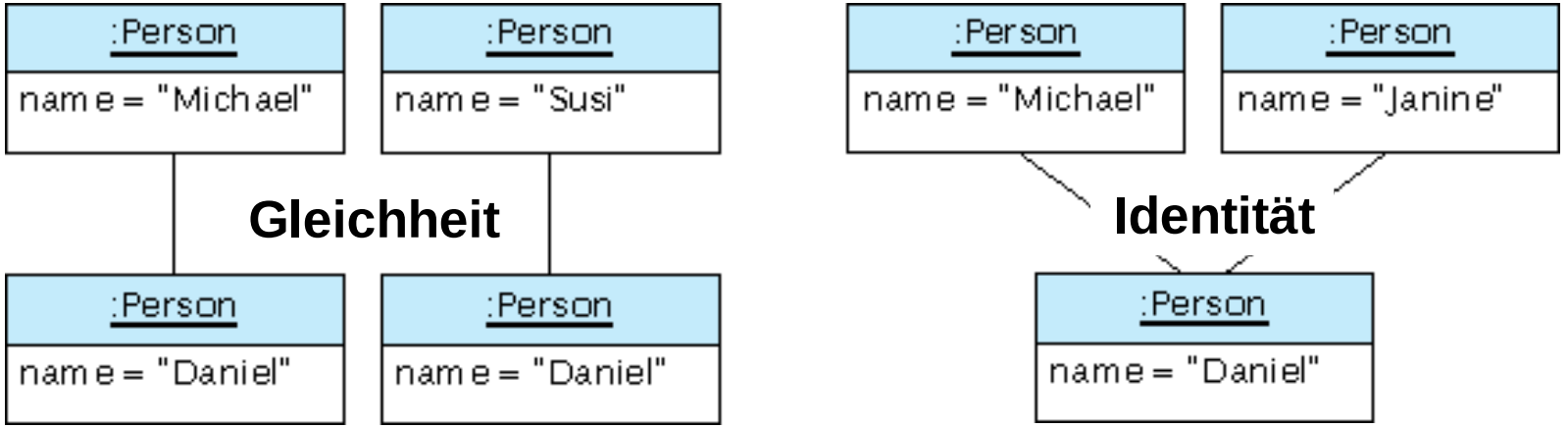
- **Gleichheit:** Verschiedene Objekte besitzen gleiche Attributwerte

- Objektorientierte Programmiersprachen beinhalten eigene Mechanismen zur Sicherstellung der Identität der Objekte (meist über Speicheradresse)

Objekt – Identität vs. Gleicheit

Beispiel

- Michael und Susi haben jeweils ein Kind mit dem Namen Daniel (Gleichheit)
- Michael und Janine sind Eltern desselben Kindes (Identität)
  - es handelt sich um dasselbe Objekt



Objekt – Notation

- Darstellung eines Objekts als Rechteck mit 2 Feldern
- Name des Objekts wird immer unterstrichen
- Operationen, werden in der UML nicht angegeben

- Aufbau von Objektbezeichung und Attributangaben (optional)

Irgendein Objekt der Klasse

:Klasse
attribut1 = Wert1 attribut2

Objekt soll über Namen angesprochen werden

objekt:Klasse

← sinnvoll, da Typ bereits bei Klasse definiert

Nur möglich, wenn Klasse eindeutig aus dem Kontext ersichtlich

objekt

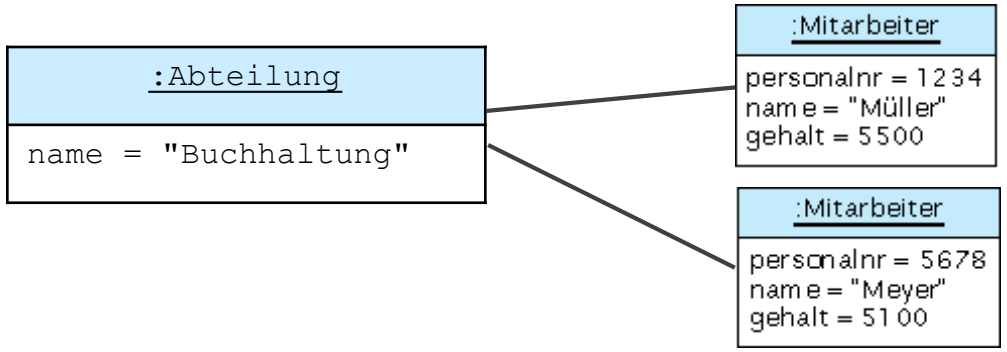
← nur sinnvoll, wenn Attributwert uninteressant

Beispiel
Das Objekt mit dem Namen einKreis
ist ein **Exemplar** der Klasse Kreis.

einKreis:Kreis
radius = 25 mittelpunkt = (10,10)

Objekt – Notation

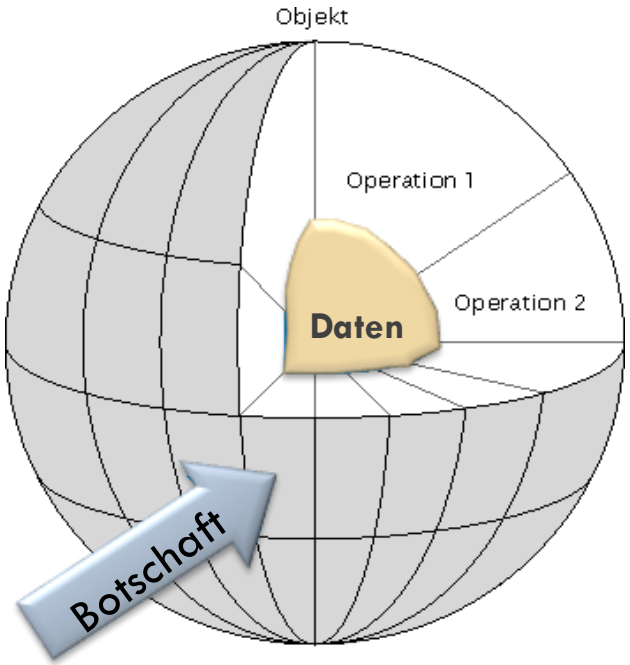
- ein Objektdiagramm beschreibt Objekte, Attributwerte und Verbindungen zwischen Objekten zu **einem bestimmten Zeitpunkt**
  (→ Momentaufnahme bzw. Schnappschuss des Systems).



- Objektname
  - muss nur **innerhalb eines Diagramms** eindeutig sein
  - in verschiedenen Diagrammen sind unterschiedliche Objekte mit gleichem Namen möglich

Objekt - Geheimnisprinzip

- Objektzustand (Daten) und -verhalten (Operationen) bilden eine Einheit
  →Kapselung
- Daten dürfen nur mittels der Operation gelesen und geändert werden.
  → Daten werden vor der Außenwelt verborgen
- ein Objekt realisiert das Geheimnisprinzip

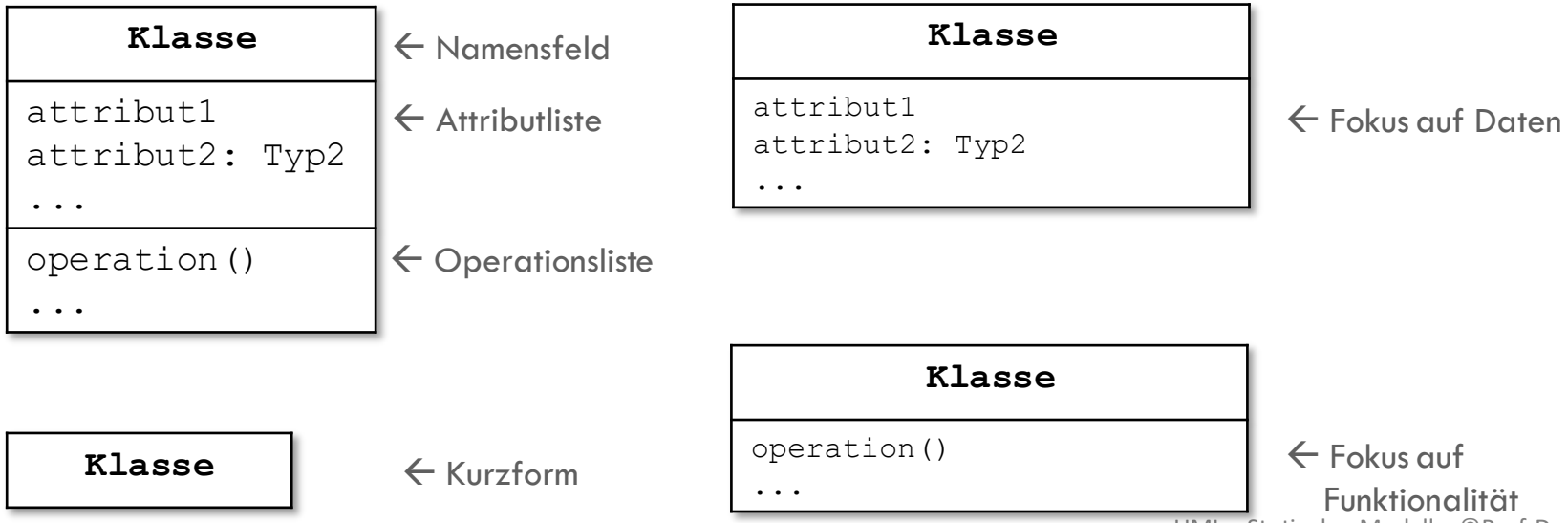


Agenda

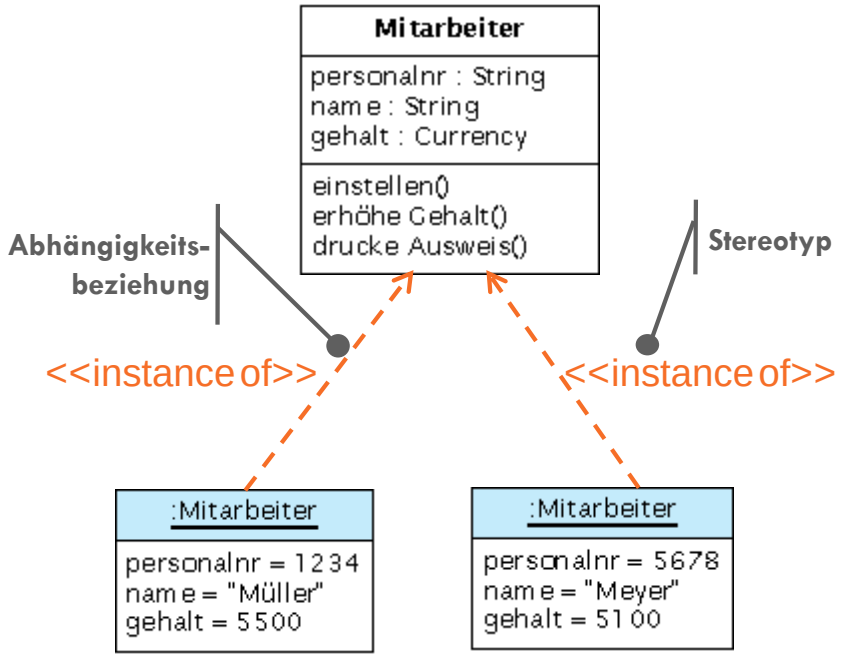
- Knoten
  - Paket
  - Objekt
  - Klasse
  - Attribut
- Assoziationen
  - Basis: Assoziationsname, Kardinalität/Multiplizität, Rollenname
  - Richtung
  - Eigenschaftswerte
  - Einschränkungen
  - Assoziationsklasse
  - Qualifizierte Assoziation
  - Abgeleitete Assoziation
  - Arten von Assoziationen
  - Höherwertige Assoziationen
- Generalisierung

Klasse - Notation

- Darstellung eines Objekts als *Rechteck* mit bis zu 3 Feldern
- Name der Klasse
  - wird fett und zentriert dargestellt
  - Substantiv im Singular (groß geschrieben), das durch ein Adjektiv ergänzt werden kann
  - Beschreibung eines einzelnen Objekts der Klasse
    - Beispiele: Mitarbeiter, PKW, Kunde
  - Eindeutigkeit innerhalb eines Pakets, besser innerhalb des gesamten Systems
  - Bei Bedarf wird der Klassenname in der UML wie folgt erweitert: Paket::Klasse



Klasse - Beispiel



Beziehung zwischen Klasse und Objekt
= Exemplarbeziehung

```
class Mitarbeiter
{
   private String personalnr;
   private String name;
   private float gehalt;

   public void einstellen()
   {
      ...
   }
   ...
   public static void main(
                    String [] args)
   {
      ...
      Mitarbeiter mueller =
            new Mitarbeiter ();
      Mitarbeiter meyer=
            new Mitarbeiter ();
      ...
   }
}
```

Klasse

Erweiterungen der Klassennotation in der UML

- Stereotyp *(stereotype)*
  - übergreifender Bezeichner, klassifiziert Elemente des Modells
  - Beispiel: **<<interface>> <<GUI>>**

- Merkmal (*property*)
  - beschreibt Eigenschaften (z.B. Einschränkungen) für Modellelement
  - Beispiel: **{Autor=Maier}** oder **{Version=1.0}**

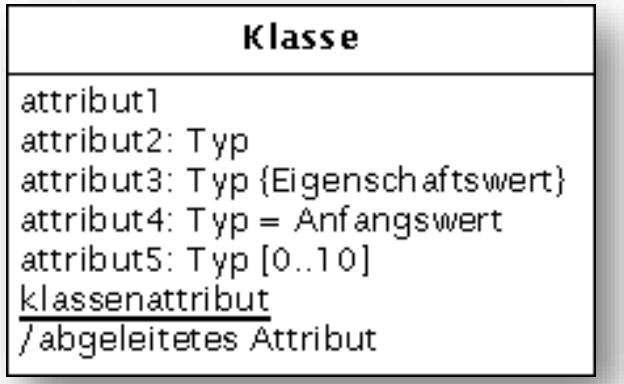
Agenda

- Knoten
  - Paket
  - Objekt
  - Klasse
  - Attribut
- Assoziationen
  - Basis: Assoziationsname, Kardinalität/Multiplizität, Rollenname
  - Richtung
  - Eigenschaftswerte
  - Einschränkungen
  - Assoziationsklasse
  - Qualifizierte Assoziation
  - Abgeleitete Assoziation
  - Arten von Assoziationen
  - Höherwertige Assoziationen
- Generalisierung

Attribut - Notation

Definition

Attribute beschreiben die **Daten**, die von den Objekten einer Klasse angenommen werden können.



Attribut: Typ = Anfangswert { Restriktion, Eigenschaftswert, ...}

```
matrikelnr: int = 0 {500 <= matrikelnr <= 8000000}
```

- Attributname
- Typ
- Anfangswert (optional)
- Liste von Merkmalen (optional):
  - Restriktionen (Beschränkungen)
  - Zusätzliche Eigenschaftswerte

- **Attributname**
  - muss **im Kontext der Klasse eindeutig** sein, außerhalb dagegen nicht
    → Beschreibung mit `Klasse.Attribut`
  - beschreibt die gespeicherten **Daten**
  - ist im Allgemeinen **ein klein geschriebenes Substantiv**

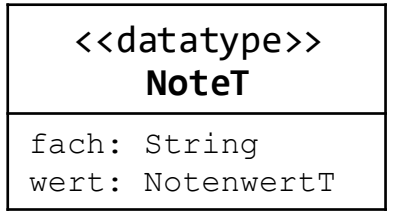
Attribut – Notation (Typ)

- Attributtyp

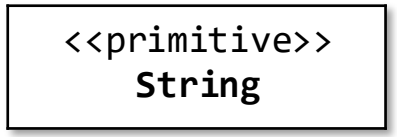
  `Typ = [Datentyp | Primitiver Datentyp | Aufzählungstyp | Klasse]`

  - Jedes Attribut wird durch einen Typ beschrieben.
  - Verwendung folgender Typen zur Erstellung des OOA-Modells
    - Datentypen
    - Primitive Datentypen
    - Aufzählungstypen
    - (elementare) Klassen

- **Datentypen**
  - Elemente eines Datentyps besitzen keine Identität
    - Beispiel:
      Kommt die Zahl 7 mehrfach vor, so handelt es sich um dieselbe Zahl.
  - dürfen Attribute und Operationen besitzen
  - werden als Klasse mit dem Stereotyp «datatype» modelliert



Attribut – Notation (Typ)

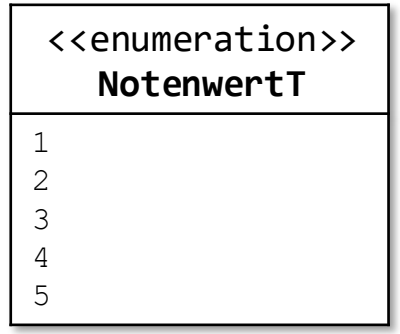


- Primitive Datentypen
  - Vordefiniert in der UML
    - `String` ⇨ Zeichenkette (Länge)
    - `Integer` ⇨ ganze Zahl
    - `UInt` ⇨ UnlimitedNatural ⇨ positive ganze Zahl
    - `Float` ⇨ Gleitkommanzahl 32 Bit
    - `Double` ⇨ Gleitkommazahl 64 Bit
    - `Fixed` (Vorkommastellen, Nachkommastellen) ⇨ Festkommazahl
    - `Boolean`
    - `Date`
    - `Time`
  - weitere können definiert werden
  - als Klasse mit dem Stereotyp «primitive» modelliert

Attribut – Notation (Typ)

Aufzählungstyp

- definiert endliche Menge von Werten
- wird als Klasse mit dem Stereotyp «enumeration» modelliert
- zur deutlichen Unterscheidung von Klassen, wird der Name der Klasse mit dem Postfix "T" versehen werden
- in der Regel: Einmalige Definition der Datentypen in Abhängigkeit von der jeweiligen Anwendung und Wiederverwendung in jedem Projekt



Elementare Klasse

- Typ eines Attributs kann selbst wieder durch eine Klasse beschrieben werden
  → Verwendung einer Klasse als komplexer Datentyp

Attribut – Notation (Mengenangaben und Anfangswert)

Mengenangaben (Multiplizitäten)

- Multiplizität *(multiplicity)* definiert, wie viele Werte ein Attribut besitzen kann
- wird in eckigen Klammern angegeben
- Beispiel: `note: NoteT [0..10]`
- es gilt:
  - [5..5] = [5]: genau 5 Werte
  - [0..*] = [*]: null bis beliebig viele Werte
  - [1..1] = [1]: genau ein Wert (gilt als Voreinstellung, wenn keine Multiplizität angegeben wird)
  - [0..1]: null oder ein Wert

Anfangswert

- der Anfangswert (initial value) definiert den Wert, den ein Attribut beim Erzeugen des zugehörigen Objekts erhält
- kann später geändert werden
- Beispiel: `rechnungsdatum: Date = heute`

Attribut – Notation (Eigenschaftswerte und Einschränkungen)

Eigenschaftswert

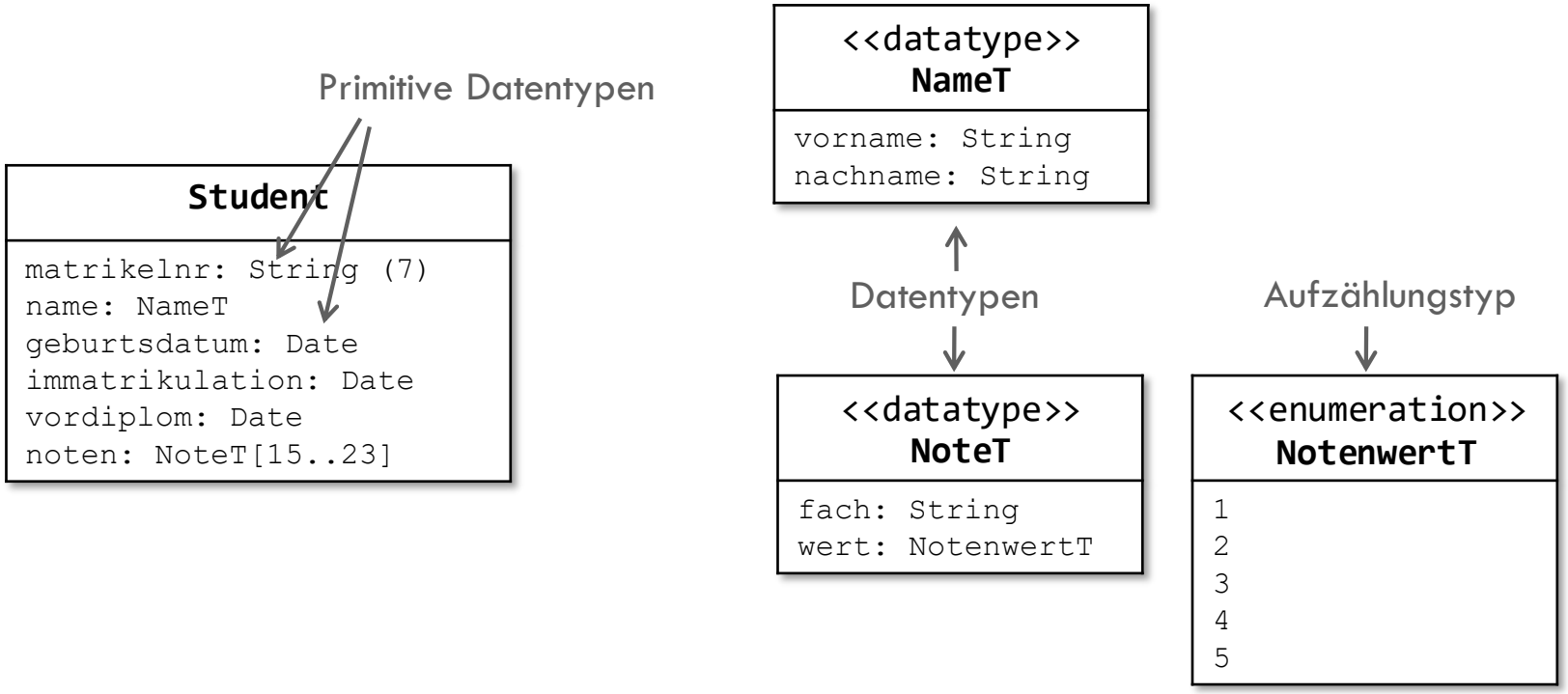
- *(property values)* spezifizieren, ob Attribute bestimmte Eigenschaften besitzen
- werden in geschweiften Klammern angegeben
- mehrere Werte werden durch Klammern getrennt
- UML bietet z.B:
  - {readOnly}: Attribut darf nicht verändert werden
  - {ordered}: Werte eines Attributs sind geordnet
- weitere Eigenschaften können definiert werden

Einschränkung

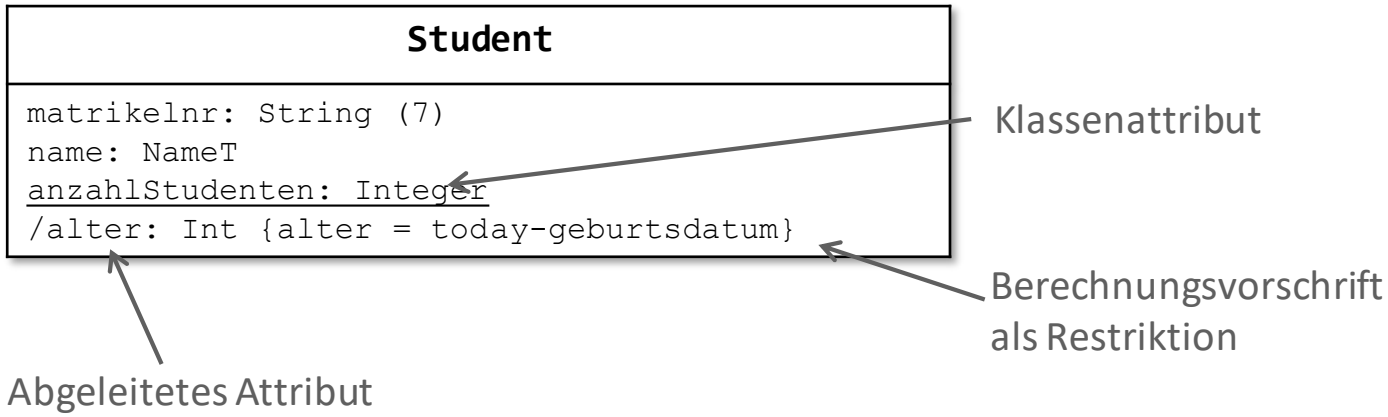
- (*constraint*) ist eine Invariante bzw. eine Zusicherung, die immer wahr sein muss
- werden in geschweiften Klammern angegeben
- können sich auf ein oder mehrere Attribute beziehen
- Beispiele:

```
{geburtsdatum <= aktuelles Datum}
{vordiplom > immatrikulation > geburtsdatum}
{verkaufspreis >= 1.5 * einkaufspreis}
```

Attribut - Beispiele



Attribut – Klassenattribut und abgeleitetes Attribut (Beispiel)



Agenda

- Knoten
  - Paket
  - Objekt
  - Klasse
  - Attribut

- Assoziationen
  - Basis: Assoziationsname, Kardinalität/Multiplizität, Rollenname
  - Richtung
  - Eigenschaftswerte
  - Einschränkungen
  - Assoziationsklasse
  - Qualifizierte Assoziation
  - Abgeleitete Assoziation
  - Arten von Assoziationen
  - Höherwertige Assoziationen
- Generalisierung

Assoziation

Begriff Assoziation

Definition

Eine Assoziation modelliert Verbindungen zwischen Objekten einer oder mehrerer Klassen.

Assoziation modelliert Verbindungen zwischen Objekten, nicht zwischen Klassen!

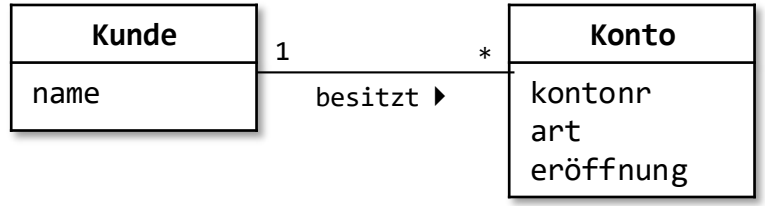
Beispiel

Objekte der Klasse Student (z.B. Paul, Susi oder Peter) haben eine Verbindung zu Objekten der Klasse Lehrveranstaltung (z.B. Einführung in die Informatik I am Dienstag)

Assoziation

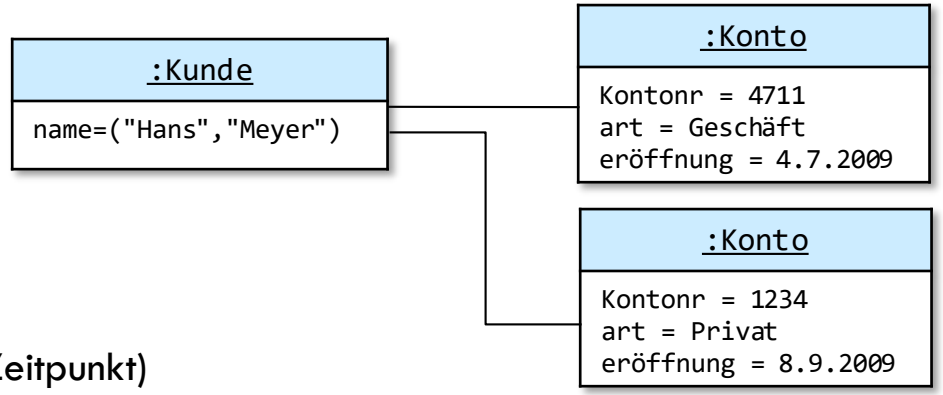
Beispiel: Assoziation zwischen Kunde und Konto

- Klassendiagramm



Zeigt Klassen und ihre Beziehungen (statische Modellelemente)

- Objektdiagramm



Zeigt Objekte und ihre Beziehungen (zu einem bestimmten Zeitpunkt)

⇨ Die Menge aller (möglichen) Verbindungen wird als Assoziation zwischen den Objekten der Klassen Kunde und Konto bezeichnet.

Assoziation

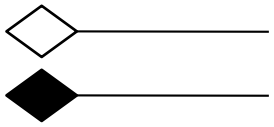
Eigenschaften von Assoziationen

- Es gibt **binäre** (zwischen zwei Objekten) **und höherwertige Assoziationen**
- Eine **reflexive Assoziation** besteht zwischen Objekten derselben Klasse
- Eine Assoziation hat eine **Richtung** (Navigierbarkeit)

  "Welches Objekt ist über die Beziehung informiert?"
  - unidirektional
  - bidirektional
  - Assoziationen sind in der Systemanalyse inhärent bidirektional

  Objekte "kennen" sich gegenseitig

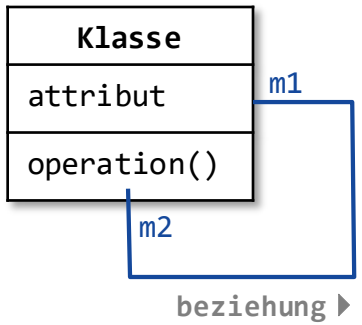
- 3 Arten von Assoziationen
  - einfache Assoziation
  - Aggregation
  - Komposition



Assoziation

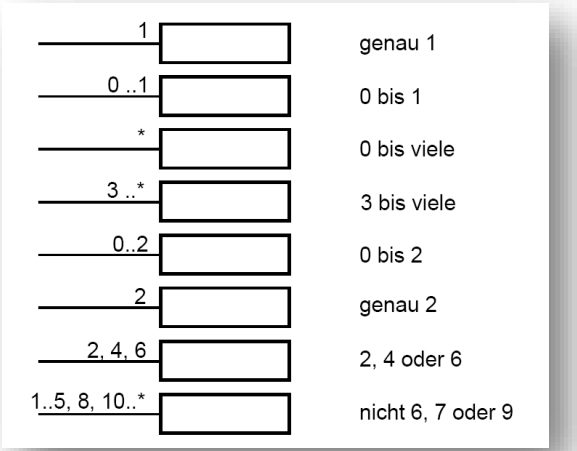
UML Notation der Multiplizitäten

- Binäre Assoziations-Linie zwischen einer oder zwei Klassen
  - Assoziationsname (optional)
  - An jedem Ende der Linie steht die Wertigkeit bzw. Kardinalität (multiplicity) → "Wie viele Objekte kann ein bestimmtes Objekt kennen"
  - An jedem Ende kann ein Rollenname stehen



reflexive Assoziation

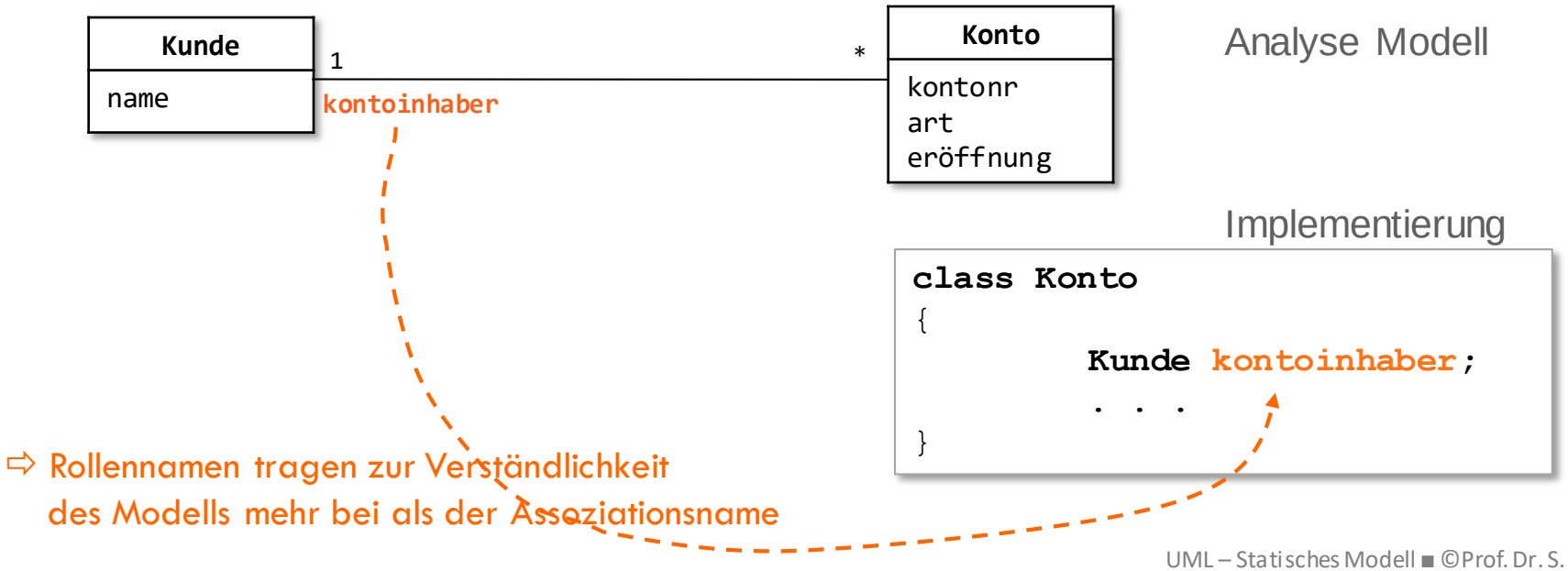
Kardinalitäten



Assoziation

Rollenname (1/2)

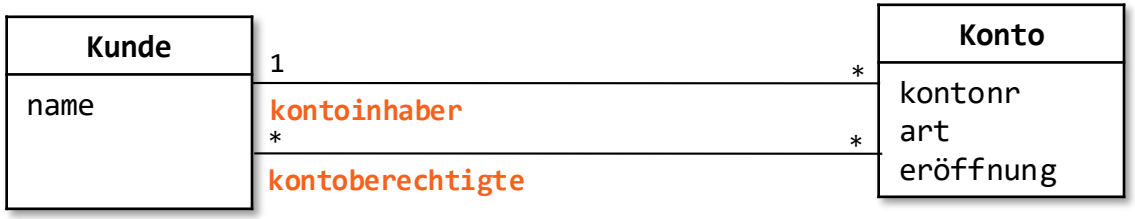
- Beschreibt Bedeutung eines Objekts in einer Assoziation
- Binäre Assoziationen besitzen maximal zwei Rollen
- Wird an das Ende der Assoziation bei der Klasse geschrieben, deren Bedeutung in der Assoziation die Rolle beschreibt



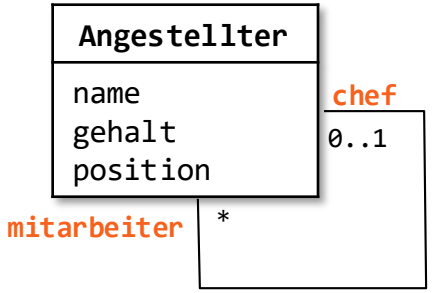
Assoziation

Rollenname (2/2)

- Rollenname ist nicht optional …
  - wenn zwischen zwei Klassen mehr als eine Assoziation besteht



  - bei reflexiven Assoziationen



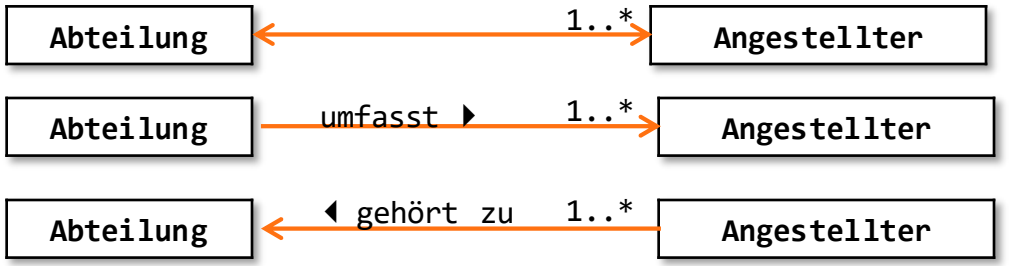
Assoziation: Richtung

Gerichtete Assoziation

- Nur ein Objekt ist über Beziehung informiert (unidirektionale Navigierbarkeit)
- Nur in die Navigationsrichtung können Botschaften geschickt werden.
- Darstellung durch geöffnete Pfeilspitze

⇨ Einem Dozenten ist nicht bekannt, wer ihm zuhört.

- Jede bidirektionale Assoziation kann durch zwei unidirektionale Assoziationen ausgedrückt werden → gleiche Semantik?



⇨ Navigierbarkeit in der Analyse nur in Ausnahmefällen festlegen!!!

Assoziation: Eigenschaftswerte

Eigenschaftswerte für Assoziationen

An ein Assoziationsende kann ein Eigenschaftswert *(property string)* angetragen werden, um die einzelnen an einer Beziehung teilnehmenden Assoziationsenden separat mit Einschränkungen zu versehen.

In der Analysephase werden häufig folgende Eigenschaftswerte benötigt:

- **`{subsets <Rolle>}:`** beschreibt eine Teilmenge von Objektbeziehungen
- **`{ordered}:`** definiert eine Ordnung auf der Menge der Objektbeziehungen (Multiplizität > 1)

⇨Fehlt der Rollenname, gilt:
Name der Klasse = Rollenname

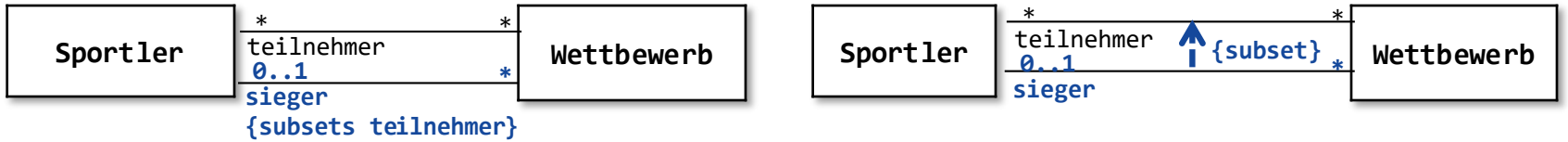
Weitere Eigenschaftswerte der UML sind:

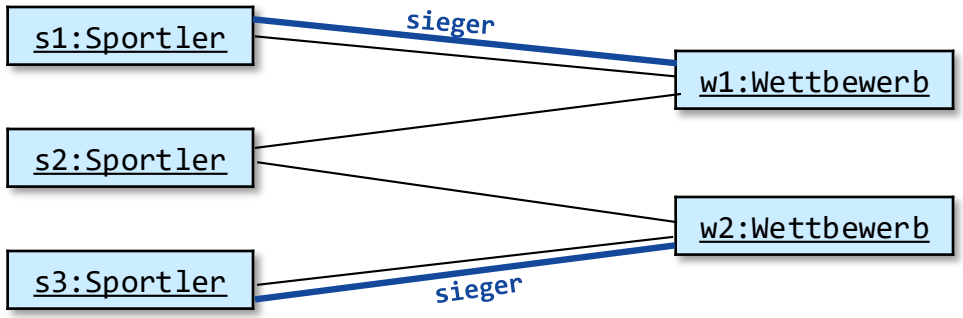
- **`{bag}:`** ein Objekt kann mehrfach an derselben Assoziation teilnehmen
  per Definition geht die UML ansonsten immer von verschiedenen Objekten aus (→ set-Eigenschaft)
- **`{sequence}`** oder **`{seq}:`** mehrfache Teilnahme, aber geordnet ⇨ `bag + ordered`
- **`{redefined <Rolle>}:`** überschreiben von Rollen im Rahmen der Vererbung
- **`{union}:`** vereinigt alle Objekte, die an einer Assoziation teilnehmen **und** mit der **`subsets`**-Eigenschaft versehen sind

Assoziation: Eigenschaftswerte

Teilmengen von Assoziationen

- Zeichnet Teilmengen von Objektverbindungen mit speziellem Bezeichner aus.





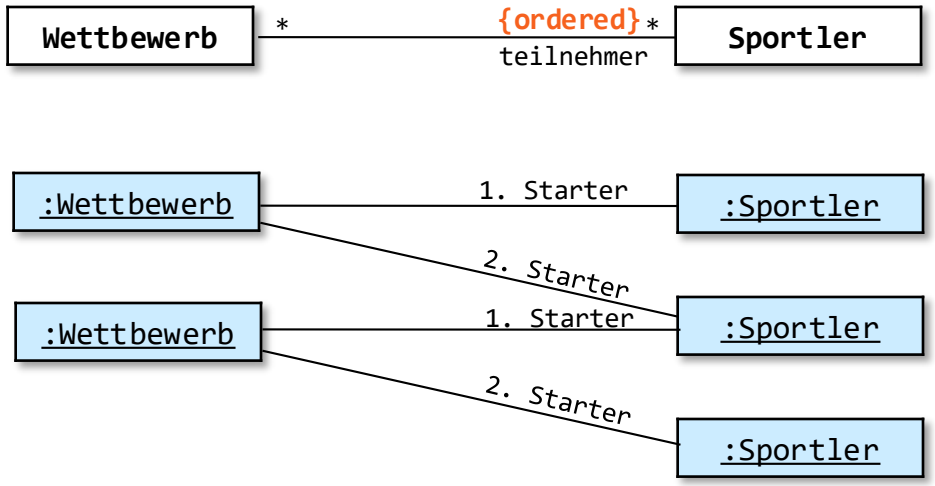
⇨ Eine blaue Objektbeziehung (Link) kann nur zwischen Objekten aufgebaut werden, zwischen denen bereits eine schwarze Objektbeziehung besteht.

Assoziation: Eigenschaftswerte

Geordnete Assoziation

- Zeigt an, dass die Menge der Objektverbindungen geordnet ist
- Möglich bei Kardinalität größer eins
- Kennzeichnung der Ordnung durch das Schlüsselwort **{ordered}**

Keine Aussage über Definition der Ordnung (z.B. zeitlich, alphabetisch)



Assoziation: Einschränkungen

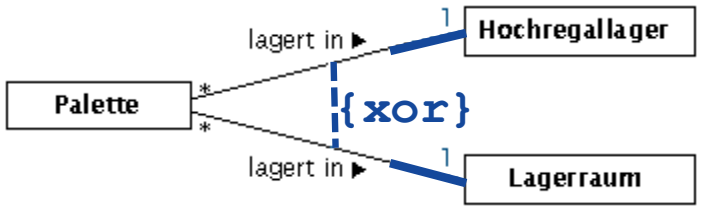
Einschränkungen einer Assoziation

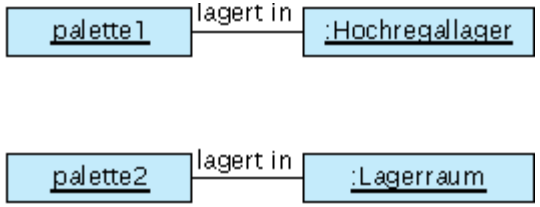
Zwischen Assoziationen können Einschränkungen formuliert werden, die als gestrichelte Linie dargestellt werden. An diese wird die einschränkende Bedingung in geschweiften Klammern notiert.

Eine Einschränkung *(constraint)* ist eine Zusicherung, die immer wahr sein muss

Beispiel 1:

- `{xor}`: sagt aus, dass zu einem Zeitpunkt **genau eine** der gekennzeichneten Assoziationen gilt





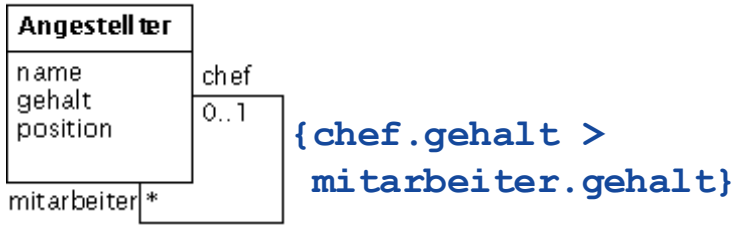
Assoziation: Einschränkungen

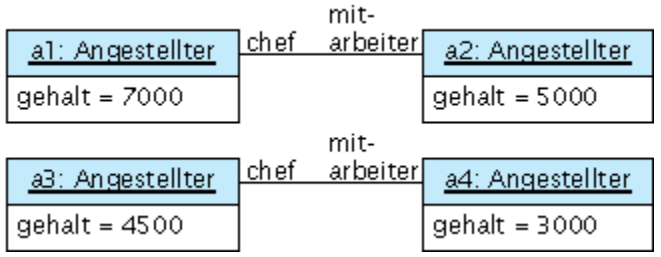
Einschränkungen einer Assoziation

Beispiel 2:

Einschränkungen können auch für einzelne Assoziationen formuliert werden.
In diesem Fall entfällt die gestrichelte Linie.

Einschränkungen können frei formuliert werden. → Definition von Standards sinnvoll.

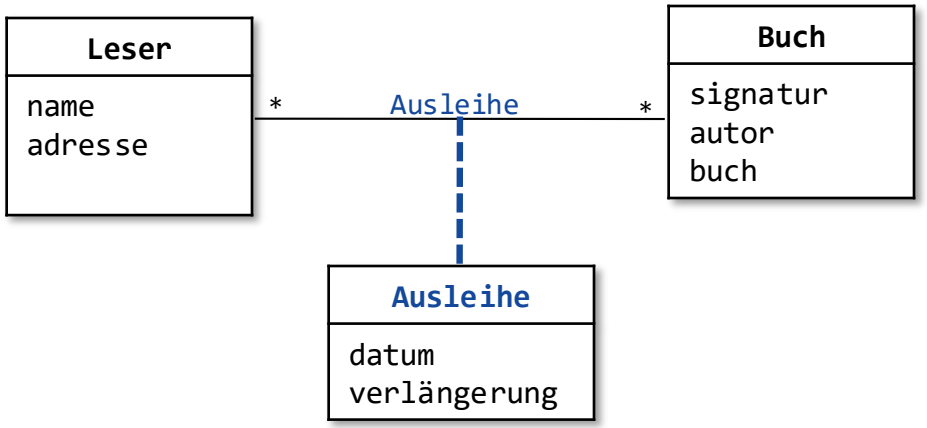




Assoziation: Assoziationsklasse

Assoziationsklasse (1/2)

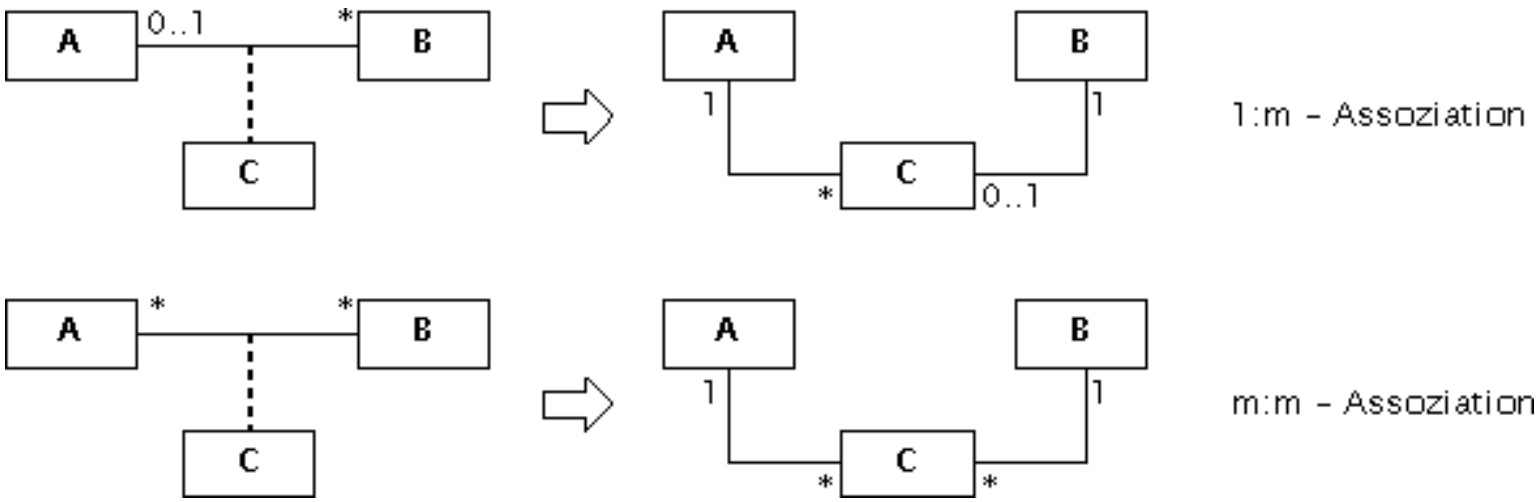
- Eine Assoziationsklasse *(association class)* besitzt die Eigenschaften **einer Klasse und einer Assoziation**.
- Beim Aufbau der Objektbeziehung zwischen zwei Objekten wird ein Objekt der Assoziationsklasse erzeugt und mit den entsprechenden Attributwerten gefüllt.
- Der **Vorteil** dieser Modellierung ist, dass die **ursprüngliche Assoziation** zwischen zwei Klassen im Modell **deutlich sichtbar** ist.



Assoziation: Assoziationsklasse

Assoziationsklasse (2/2)

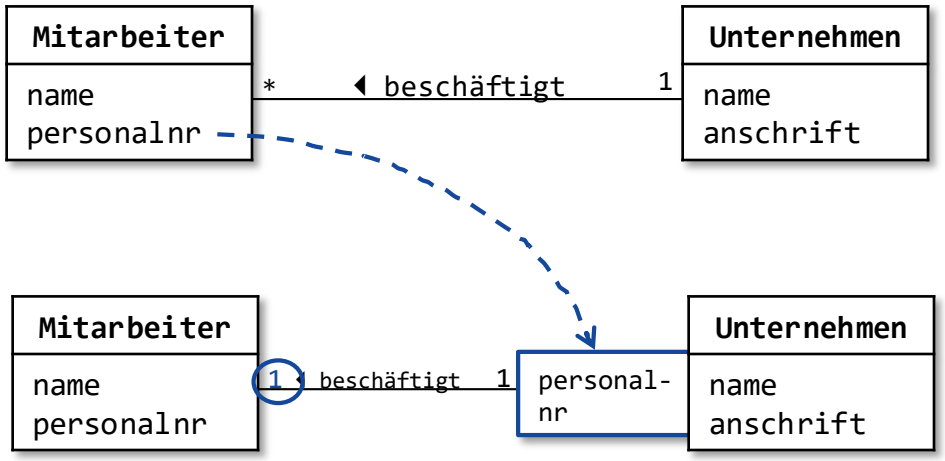
- Prinzipiell kann jede Assoziationsklasse **in eine normale Klasse transformiert** werden.
- Assoziationsklassen werden nur in der Analyse verwendet und im Entwurf aufgelöst. → s.u.
- Wichtig ist dabei die Übernahme der **Kardinalitäten**.



Assoziation: qualifizierte Assoziation

Qualifizierte Assoziation (1/2)

- **Einteilung der Menge** der assoziierten Objekte **durch spezielles Attribut**, dessen Wert ein oder mehrere Objekte auf der anderen Seite selektiert
- Erhöhen den Informationsgehalt des Klassenmodells
- Nur in binären Assoziationen zulässig
- Das qualifizierende Attribut wird in einem Rechteck an der Seite der Klasse notiert



⇨Ein Unternehmen beschäftigt eine Menge von Mitarbeitern

⇨ Jeder Mitarbeiter gehört genau zu einem Unternehmen

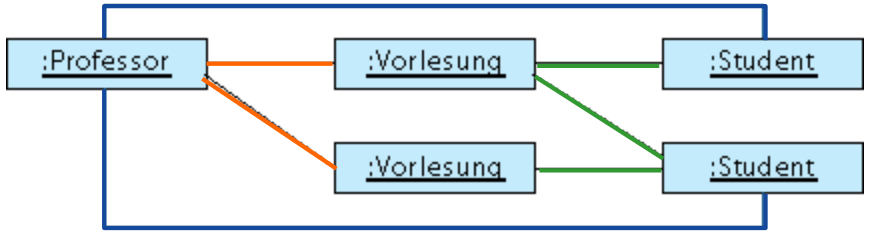
⇨ Mitarbeiter werden über ihre Personalnummer identifiziert

Qualifikationsangaben können die Kardinalität verändern!

Assoziation: abgeleitete Assoziation

Abgeleitete Assoziation *(derived association)*

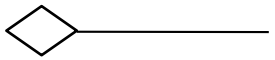
- Assoziation, deren konkrete Objektbeziehungen jederzeit aus den Werten anderer Objektbeziehungen und Objekte abgeleitet werden (→redundant)
- wird durch das Präfix „/“ gekennzeichnet
- Ableitungsvorschrift wird ggf. als Restriktion notiert



Assoziation: Arten von Assoziation

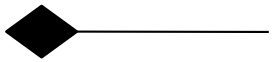
Einfache Assoziation *(ordinary association)*

Aggregation *(aggregation)*



- Lässt sich durch „ist Teil von“ bzw. „besteht aus“ beschreiben
  (→ Ganzes und Teile, whole part)

Komposition *(composition)*



- »starke« Aggregation
- Multiplizität der Aggregatklasse ist 1 oder 0..1
- Wird das Ganze gelöscht, so werden automatisch auch seine Teile gelöscht
  (→ they live and die with it)
- Ein Teil darf einem anderem Ganzen zugeordnet werden
- Das Ganze ist verantwortlich für das Erzeugen und Löschen seiner Teile

Assoziation: Arten von Assoziation

Aggregation

- ist **asymmetrisch**

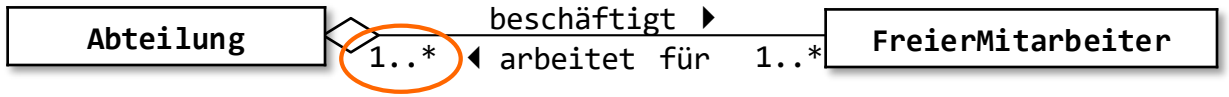
  wenn B Teil von A ist, dann darf A nicht Teil von B sein

- ist **transitiv**

  wenn A Teil von B und B Teil von C, dann ist auch A Teil von C

- muss **nicht exklusiv** sein

  B darf gleichzeitig Teil von A und Teil von C sein



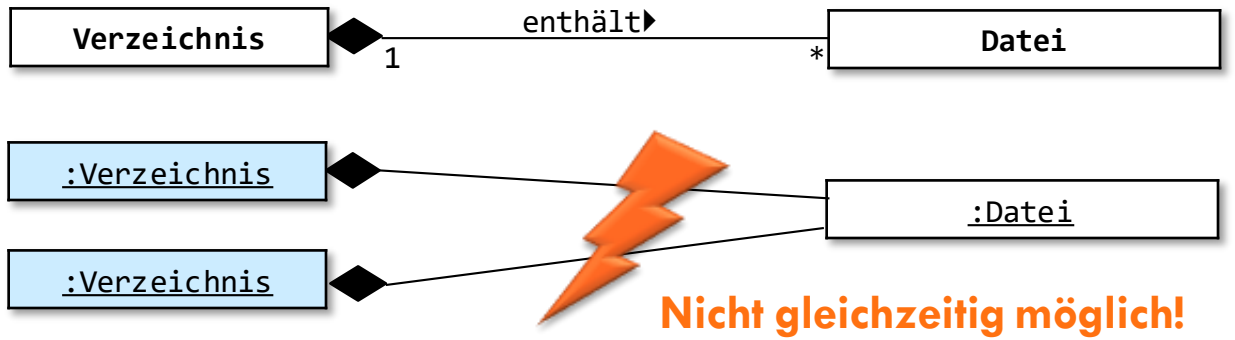
- Das Ganze übernimmt Aufgaben stellvertretend für seine Teile

Assoziation: Arten von Assoziation

Komposition

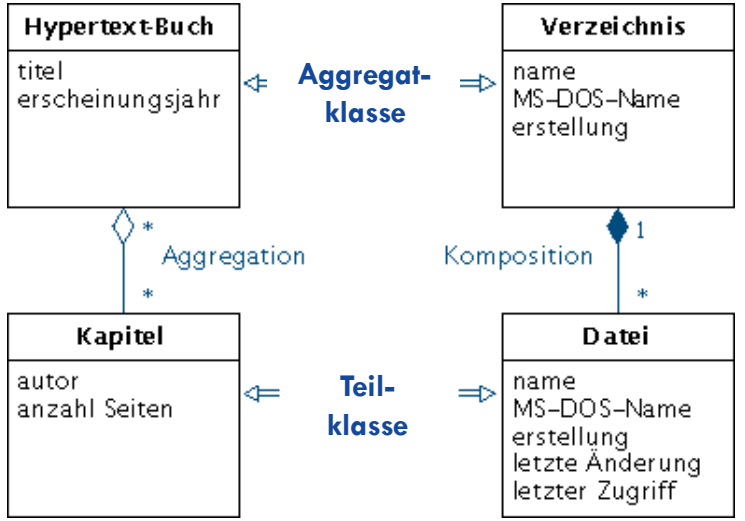
Zusätzlich zur Aggregation gilt:

- Jedes Objekt der Teilklasse kann zu einem bestimmten Zeitpunkt nur Komponente eines einzigen Objekts der Aggregatklasse sein
  - Kardinalität der Aggregatklasse <= 1
  - Ein Teil darf evtl. auch anderem Ganzen zugeordnet werden (aber nicht gleichzeitig)
- Dynamische Semantik des Ganzen gilt auch für seine Teile *(propagation semantics)*
  Wird das Ganze kopiert, werden auch seine Teile kopiert



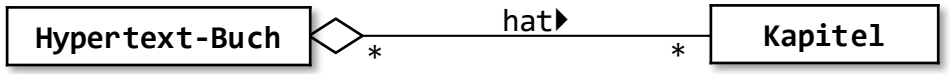
Assoziation: Arten von Assoziation

Aggregation versus Komposition



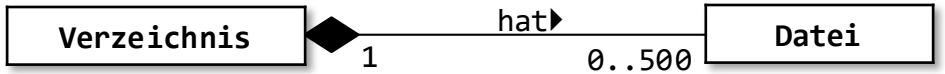
Aggregation: "Hypertext-Buch hat Kapitel"

- Kapitel gehören notwendigerweise zu einem (Hypertext)-Buch
  → Aggregation
- Ein Kapitel kann aber auch in verschiedene Bücher eingebunden sein
  → keine Komposition



Komposition: „Verzeichnis hat Datei"

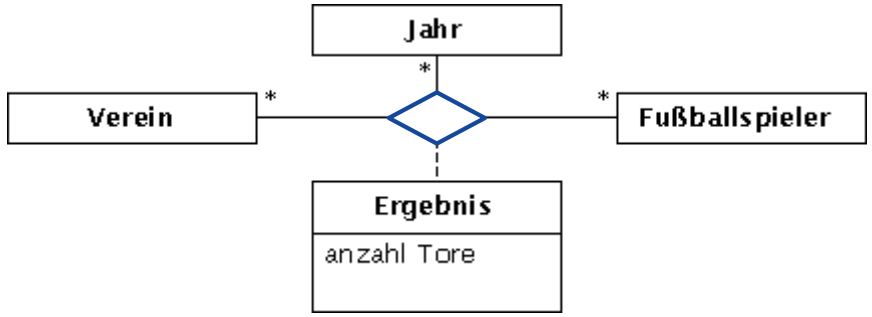
- Wird das Verzeichnis gelöscht, werden auch alle existenzabhängigen Einzelteile mitgelöscht
  → Komposition



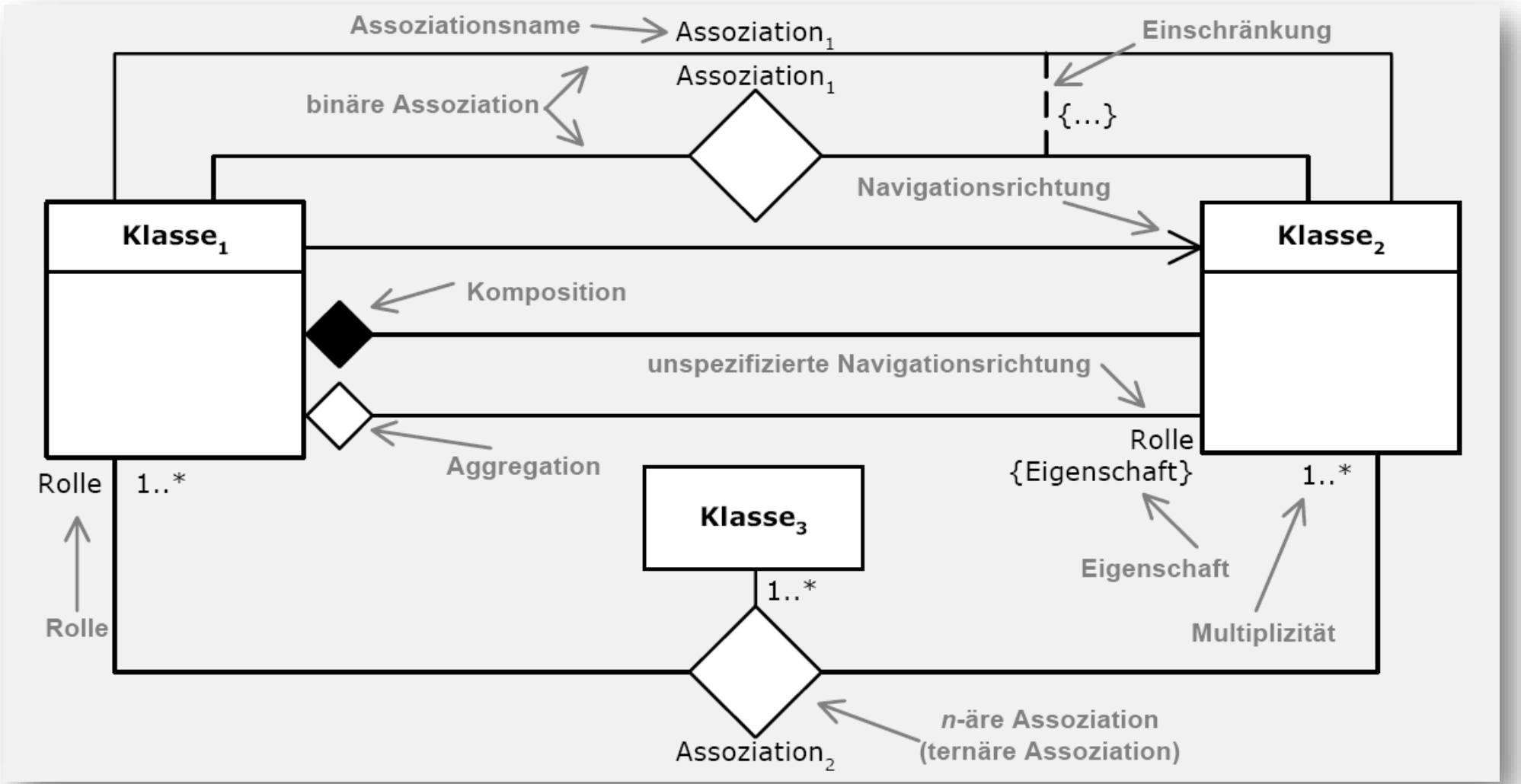
Assoziation: Höherwertige Assoziationen

Höherwertige Assoziationen

- Prinzipiell sind auch Assoziationen zwischen drei und mehr Klassen möglich
  Symbol: Diamant
- Meist wird statt des Diamanten nur eine einfache Linie verwendet
- Bezeichnung: n-äre Assoziation
- Ternäre und höhere Assoziationen können keine Aggregation oder Komposition bilden



Assoziationen



Agenda

- Knoten
  - Paket
  - Objekt
  - Klasse
  - Attribut

- Assoziationen
  - Basis: Assoziationsname, Kardinalität/Multiplizität, Rollenname
  - Richtung
  - Eigenschaftswerte
  - Einschränkungen
  - Assoziationsklasse
  - Qualifizierte Assoziation
  - Abgeleitete Assoziation
  - Arten von Assoziationen
  - Höherwertige Assoziationen

- Generalisierung

Generalisierung

Definition

Eine Vererbung *(generalization)* ist eine Beziehung zwischen einer allgemeinen Klasse und einer spezialisierten Klasse.

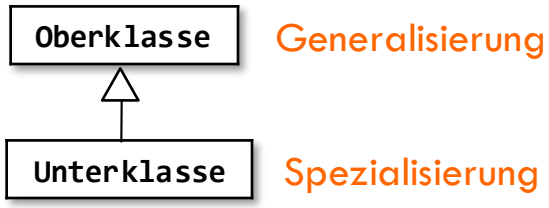
- Vererbung ist ein Abstraktionsprinzip zur hierarchischen Strukturierung (Einordnung der Klassen in eine Hierarchie)

  → Vererbung findet nur zwischen Klassen statt

- Ziel: Gemeinsame Eigenschaften und Verhaltensweisen zusammenfassen
- Spezialisierte Klasse ist vollständig konsistent mit allgemeiner Klasse, kann aber zusätzliche Informationen (Attribute, Operationen, Assoziationen) haben

- Allgemeine Klasse = Oberklasse *(super class)*

- Spezialisierte Klasse = Unterklasse *(sub class)*

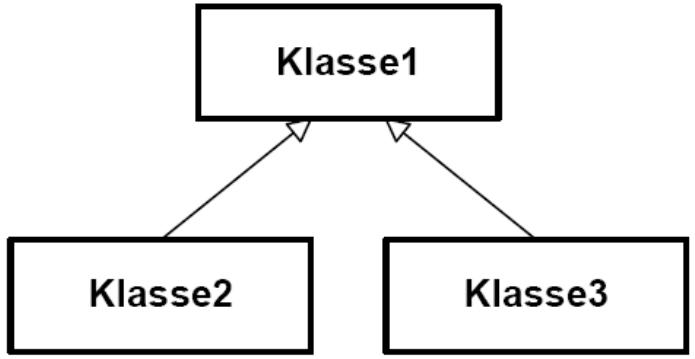


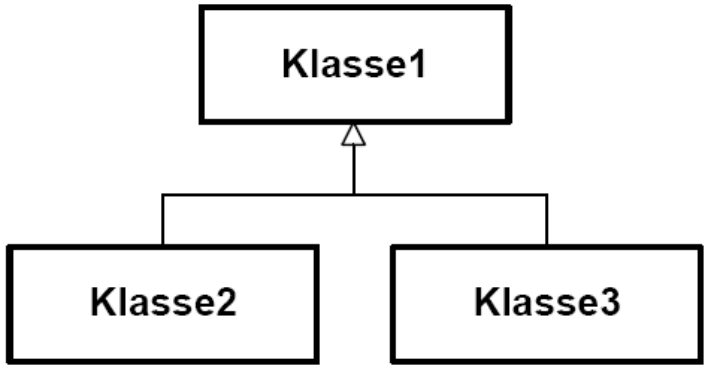
Substitutionsprinzip: Objekt der Unterklasse kann überall verwendet werden, wo ein Objekt der Oberklasse erlaubt ist, aber nicht umgekehrt!

Generalisierung

UML Notation der Vererbung

- Weißes bzw. transparentes Dreieck bei der Oberklasse



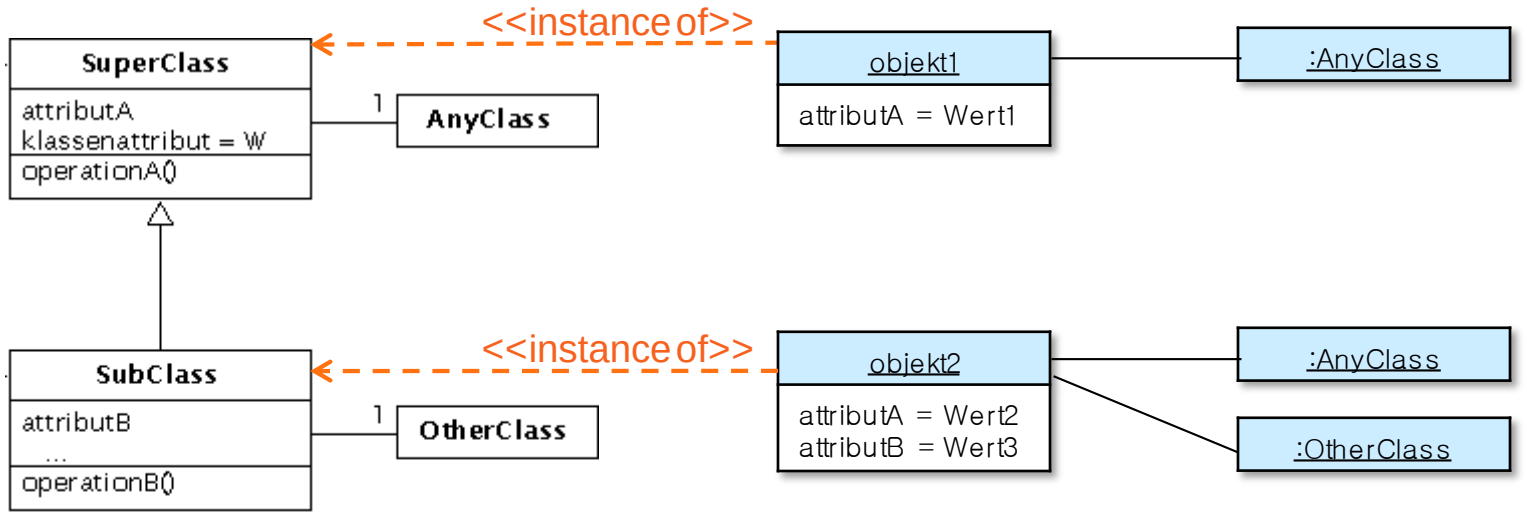


Darstellungen gleichwertig ⇨ alternativ verwendbar

Generalisierung

Was wird vererbt?

Operationen, Attribute und Assoziationen



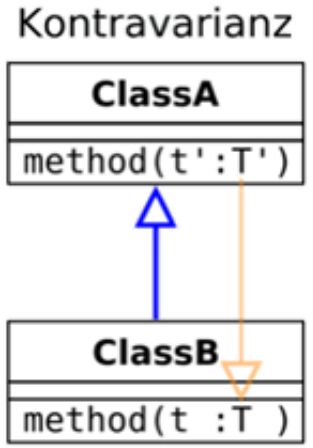
1. `Attribut A` von `Superclass` wird nach `Subclass` vererbt
2. `Operationen A()` von `Superclass` auch auf `Subclass` anwendbar
3. Klassenattribut mit dem Wert `W` von `Superclass` an `Subclass` vererbt
4. Assoziation zwischen `Superclass` und `AnyClass` an `Subclass` vererbt

Generalisierung

Überschreiben von Operationen

- Unterklassen können das Verhalten ihrer Oberklassen verfeinern, redefinieren bzw. überschreiben *(redefine bzw. override)*
  → *Nicht Verwechseln mit überladen!*
- Operationen können in der Unterklasse überschrieben, aber nicht eliminiert werden

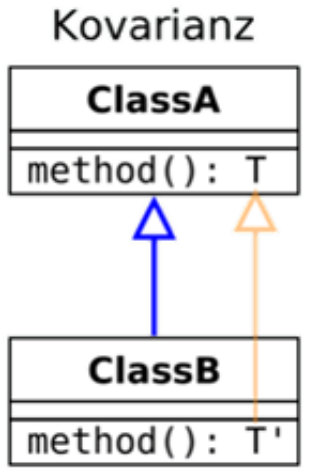
Gleiche Signatur der Operationen erforderlich bzw. gemäß dem Substitutionsprinzip!



Dem Substitutionsprinzip entspricht die Unterstützung von
- für Eingangs- und
- Kovarianz für Ausgangsparameter/Rückgabewerte

⇨ Methoden der Unterklasse müssen mindestens die Eingangs- parameter akzeptieren, die die Oberklasse auch akzeptieren würde, d.h. sie können allgemeiner sein (**Kontravarianz**)

⇨ der Typ des Rückgabewerts darf spezieller als in der Oberklasse sein (**Kovarianz**)



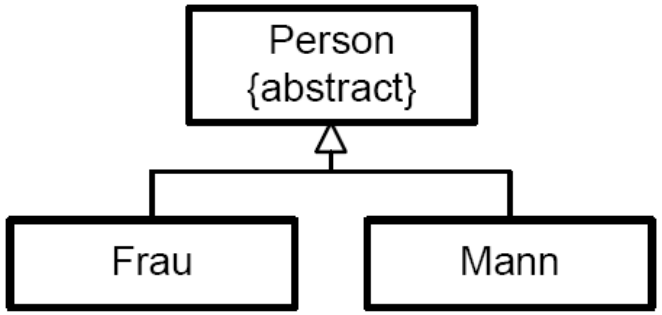
Viele Programmiersprachen unterstützen keine Varianz!

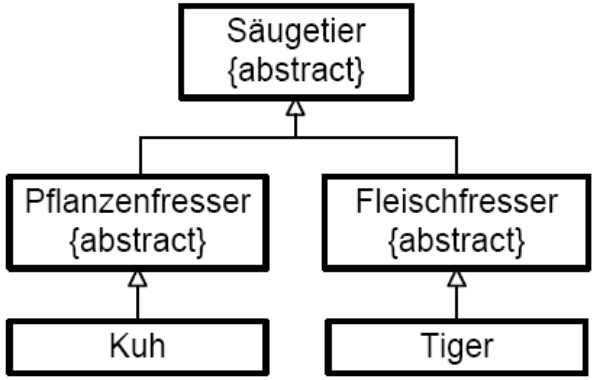
Generalisierung

Abstrakte Klasse

- Unterscheidung zwischen abstrakten und konkreten Klassen
- Von einer abstrakten Klasse können keine Objekte erzeugt werden
- Verwendung, um ihre Informationen an spezialisierte Klassen zu vererben
- Kennzeichnung durch **kursiven Namen** oder durch **Merkmal {abstract}**
- Eine abstrakte Klasse kann abstrakte Operationen enthalten.
  Abstrakte Operationen müssen in der Unterklasse implementiert werden.

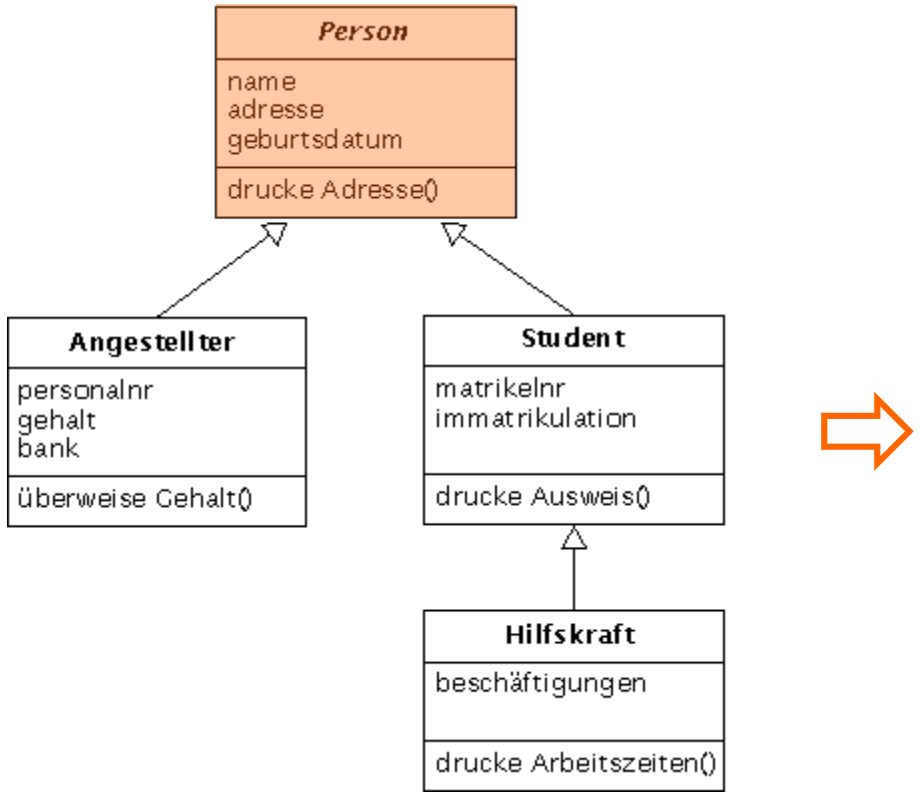
Die Verwendung abstrakter Klassen ohne Vererbung ist sinnlos !

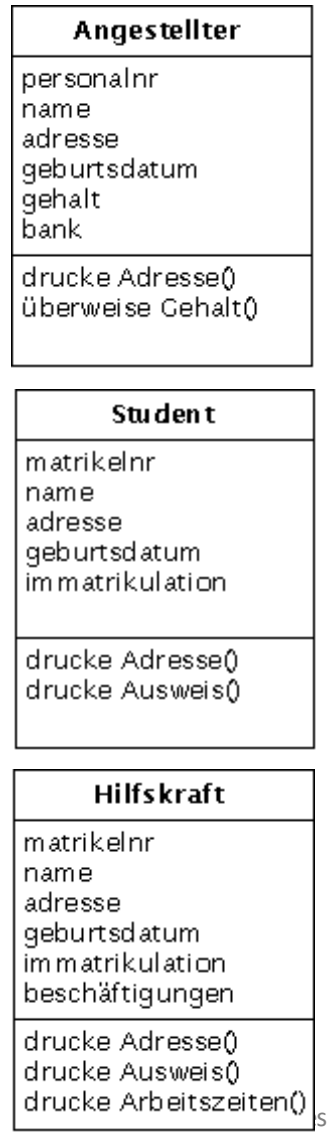




Generalisierung

Beispiel





Generalisierung

Polymorphismus

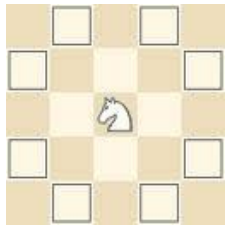
Definition

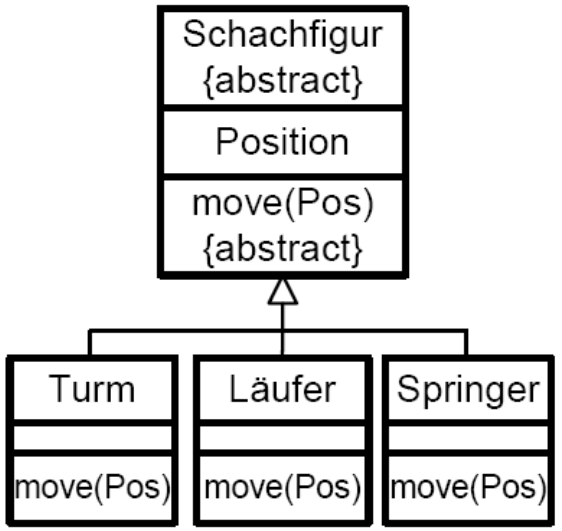
Eine Objektreferenz/-variable ist polymorph, wenn sie auf unterschiedliche Objekttypen (→ Klassen) verweisen kann. Wird über diese Objektreferenz/-variable eine Methode aktiviert, kann die Methode abhängig vom Objekttyp (→ Klassenzugehörigkeit) verschiedene Semantiken besitzen.

Beispiel

- Methode `move(Position)` bei den verschieden Spielfiguren unterschiedlich implementiert
  - Turm: Nur geradeaus
  - Läufer: Nur diagonal
  - Springer: 1 Feld gerade, 1 Feld diagonal

⇨ Gleiche Botschaft `move(Position)` löst unterschiedliche Reaktion aus!

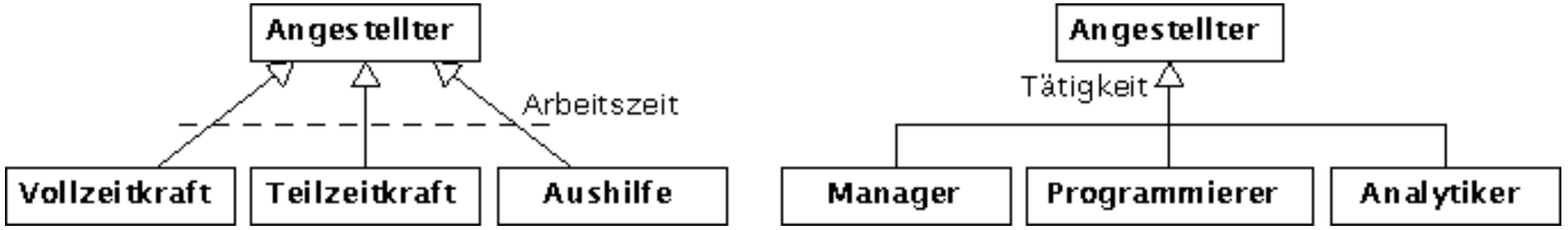




Generalisierung

Generalisierungsmenge

Die Generalisierungsmenge *(generalization set)* gibt an, nach welchem Kriterium die Struktur gebildet wird



Generalisierung

Vor- und Nachteile der Vererbung

\+ Unterstützung der Änderbarkeit
- Änderung von Attributen in der Oberklasse wirkt sich automatisch auf alle Unterklassen aus

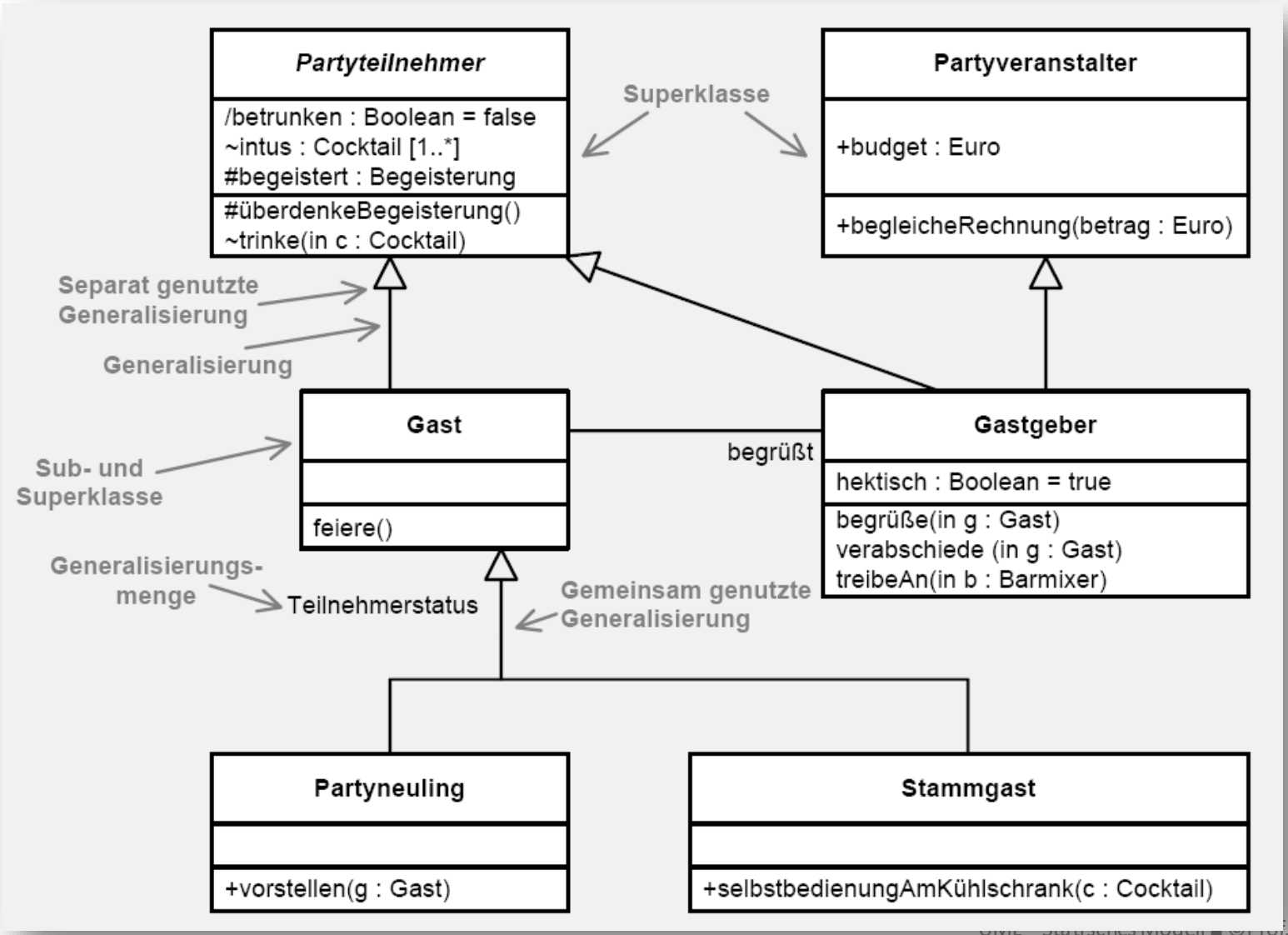
\+ Einsparung von Code

\+ Unterstützung der Wiederverwendbarkeit

\- Verletzung des Geheimnisprinzips
- Unterklasse ist von Änderungen der Oberklasse abhängig
- Um die Unterklasse zu verstehen, muss auch die Oberklasse betrachtet werden

⇨ Vererbung ist ein mächtiges, aber auch schwieriges Konzept!

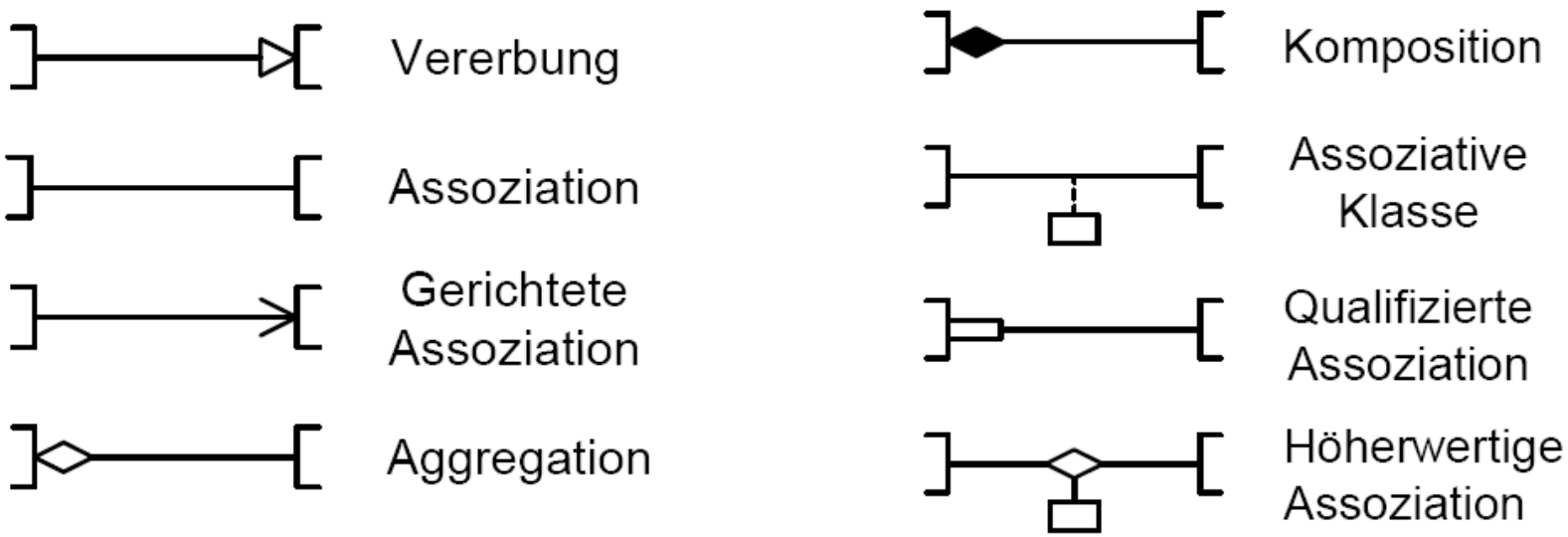
Generalisierung



Beziehungen

Zusammenfassung

- Eine Assoziation modelliert Verbindungen zwischen Objekten einer oder mehrerer Klassen
- Sonderfälle der Assoziation sind Aggregation und Komposition
- Vererbung beschreibt die Beziehung zwischen einer allgemeinen Klasse und einer spezialisierten Klasse



Agenda

- Knoten
  - Paket
  - Objekt
  - Klasse
  - Attribut

- Assoziationen
  - Basis: Assoziationsname, Kardinalität/Multiplizität, Rollenname
  - Richtung
  - Eigenschaftswerte
  - Einschränkungen
  - Assoziationsklasse
  - Qualifizierte Assoziation
  - Abgeleitete Assoziation
  - Arten von Assoziationen
  - Höherwertige Assoziationen

- Generalisierung

www.fh-dortmund.de

facebook.com/fhdortmund instagram.com/fhdortmund twitter.com/fh_dortmund youtube.com/FachhochschuleDO