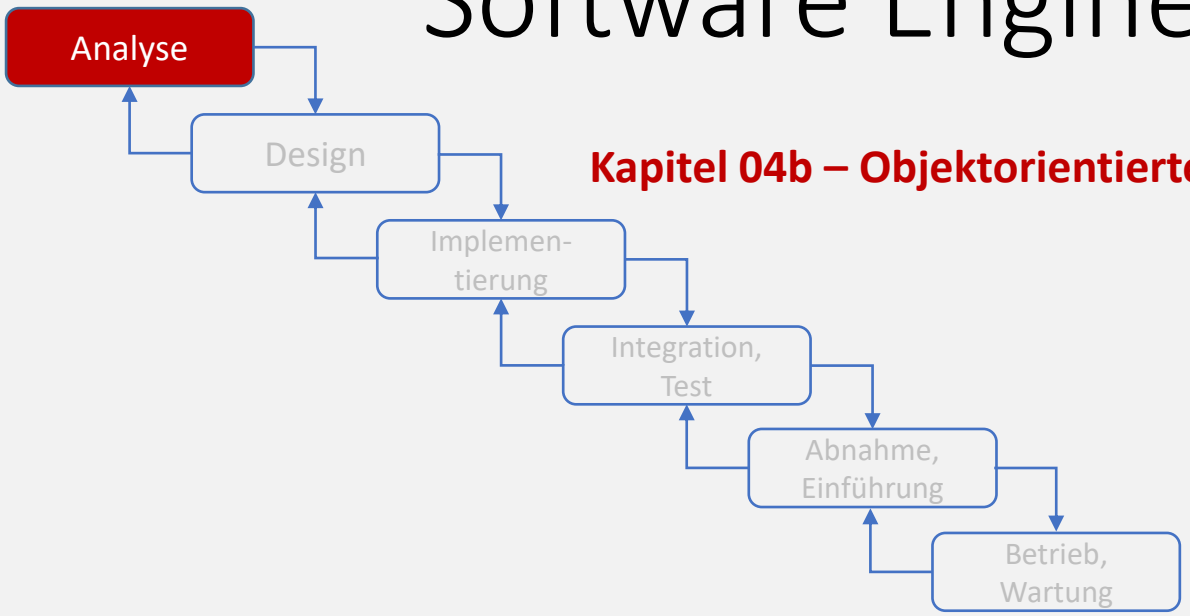
Software Engineering

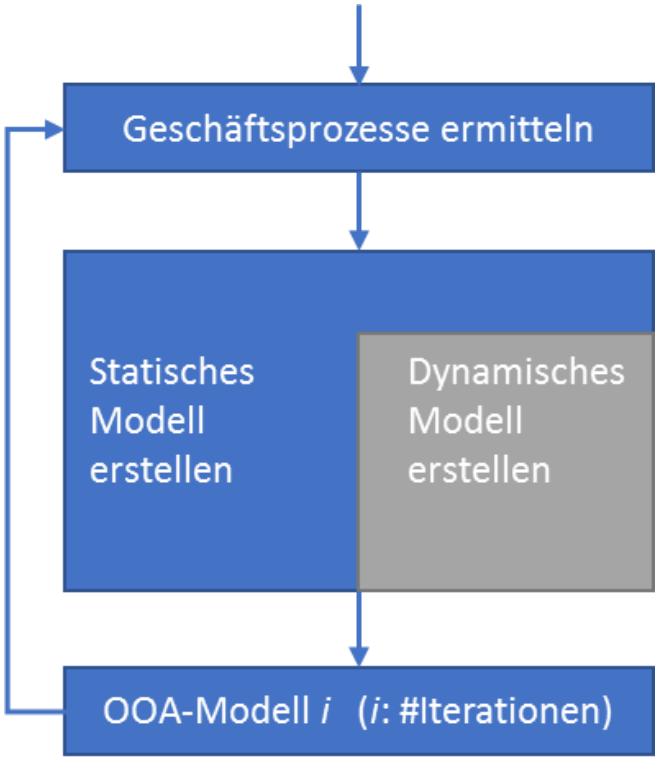


Kapitel 04b – Objektorientierte Analyse (OOA)

Klassendiagramme

OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG
INFORMATIK UND MATHEMATIK

Erinnerung: OOA Makroprozess



Aufgaben des Makroprozesses:

- Analyse im Großen
- 6 Schritte zum statischen Modell
- 4 Schritte zum dynamischen Modell

Analyse im Großen

Use Case Modell aufstellen, liefert als Ergebnisse:

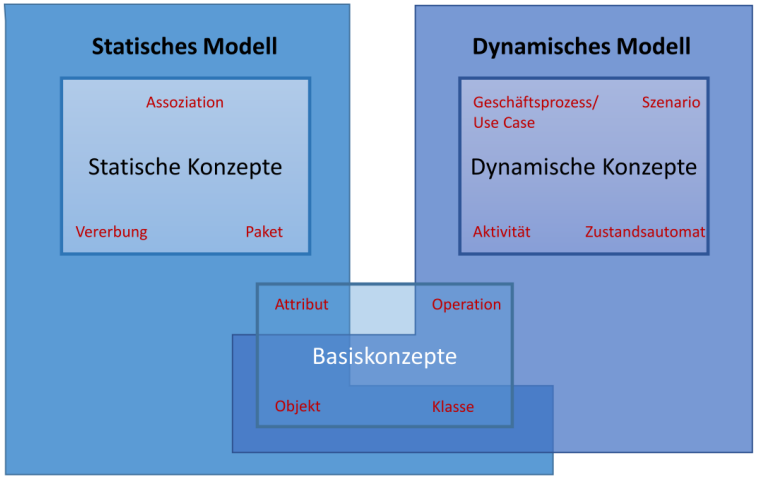
- Use-Case-Diagramm 
- Use-Case-Beschreibung 
- Aktivitätsdiagramme 
- Zustandsdiagramm 

Pakete bilden, beinhaltet: 

- Teilsysteme festlegen (Modellelemente zu Paketen zusammenfassen)
- Bei großen Systemen erfolgt Paketbildung meist zu Anfang
- Ergebnis: Paketdiagramm

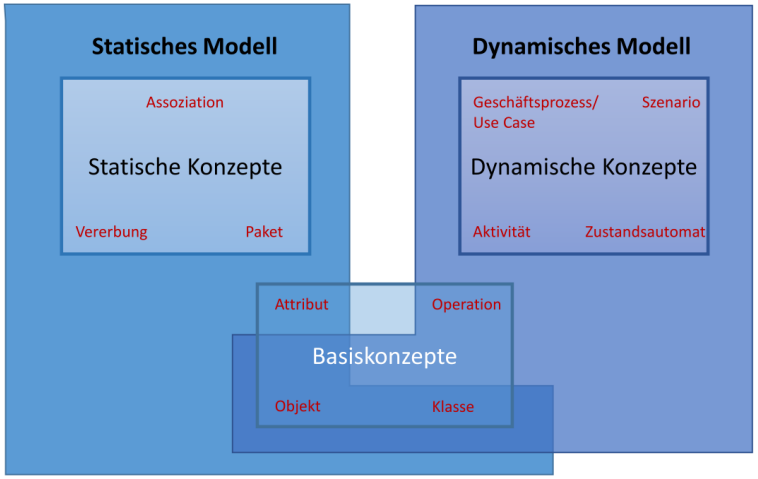
Entwicklung eines statischen Modells (6 Schritte)

1. Klassen identifizieren:
   - Ergebnis: Klassendiagramm, Kurzbeschreibung der Klassen
2. Assoziationen identifizieren:
   - Ergebnis: Klassendiagramm
3. Attribute identifizieren:
   - Ergebnis: Klassendiagramm
4. Vererbungsstruktur identifizieren
   - Ergebnis: Klassendiagramm
5. Assoziationen vervollständigen:
   - Ergebnis: Klassendiagramm, Objektdiagramm
6. Attribute spezifizieren:
   - Ergebnis: Attributspezifikation

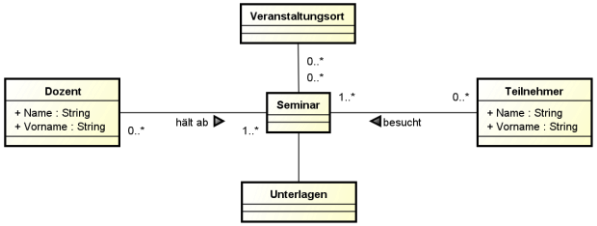


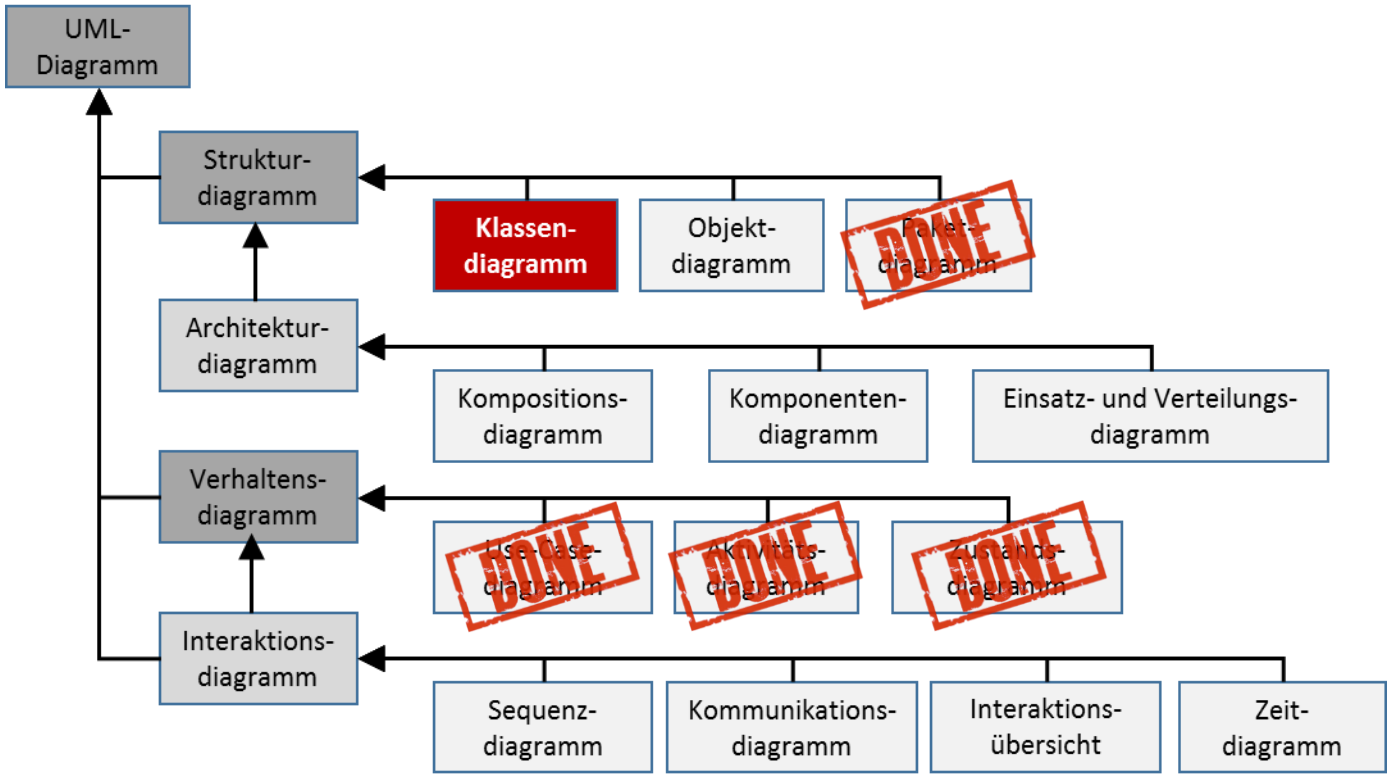
Entwicklung eines dynamischen Modells (4 Schritte)

1. Szenarios erstellen:
   - Ergebnis: Sequenzdiagramm, Kollaborationsdiagramm (Alternativ/ergänzend auch Aktivitätsdiagramme)
2. Zustandsautomat erstellen:
   - Ergebnis: Zustandsdiagramm
3. Operationen eintragen:
   - Ergebnis: Klassendiagramm
4. Operationen beschreiben:
   - Ergebnis: Klassendiagramm, fachliche Beschreibung der Operationen, Zustandsautomaten, Aktivitätsdiagramme

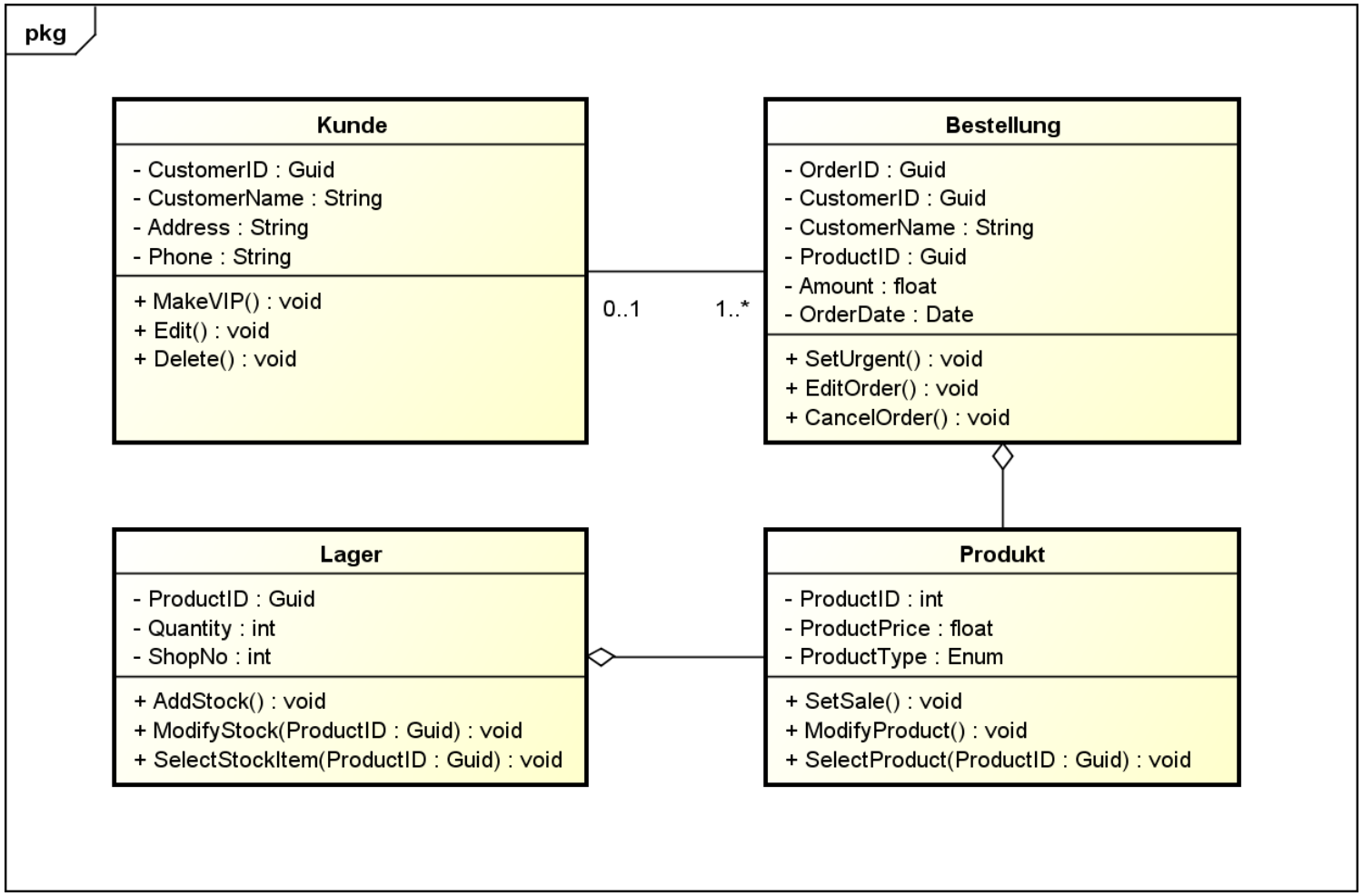


UML: Klassendiagramm





UML: Klassendiagramm Beispiel

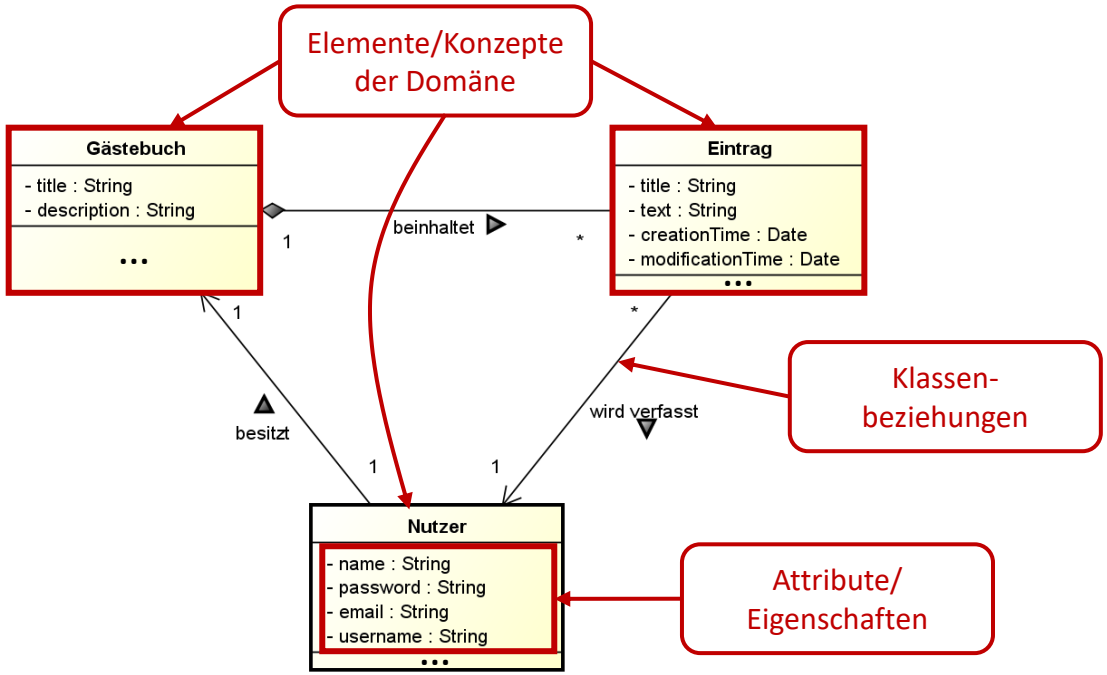


UML: Klassendiagramm Domänenmodelle

Ziel: Übergang vom Problem zur Implementierung soll ohne Bruch erfolgen

Lösung: **Domänenmodellierung (engl.: Domain Modeling)**

- Problembereich in der Sprache der Anwender modellieren
- Wissen des Problembereichs als abstraktes Modell darstellen
- Modell ist das destillierte Wissen der Phasen vor der Implementierung



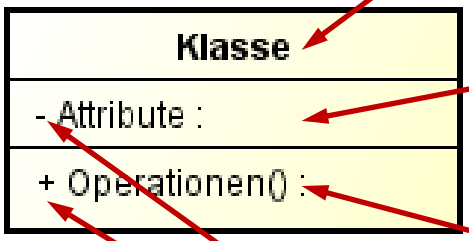
UML: Klassendiagramm

Welche Informationen werden in einem Klassendiagramm transportiert?

- Existenz von Klassen
- Eigenschaften (Attribute) der Klassen
- Operationen (Methoden) der Klassen
- Beziehungen zwischen Klassen:
  - Generalisierung (Beziehung zwischen generellerer und speziellerer Klasse)
  - Assoziation (Allgemeine Beziehung)
  - Aggregation (Ganzes-Teile-Beziehung)
  - Komposition (Ganzes-Teile-Beziehung: Spezialfall der Aggregation (Teile können nicht ohne das Ganze existieren))

Beispiele?

UML: Klassendiagramm
Klassen



Klassenname:

- Definiert den Namen der Klasse
- Ist eine Pflichtangabe
- Sollte eindeutig und sprechend sein

Attribute:

- Definieren die Eigenschaften (Objektzustand)
- Sind optional (können im konzeptuellen Modell weggelassen werden; **im Projektbericht vorhanden!**)

Operationen:

- Definieren das Verhalten
- Sind optional (können im konzeptuellen Modell weggelassen warden; **im Projektbericht vorhanden!**)

Sichtbarkeiten

UML: Klassendiagramm
Erstellung, Schritt 1: Definition der Klassen

Zunächst müssen die Basisbausteine des Klassendiagramms – *die Klassen* – identifiziert werden:

→ **Klassen identifizieren**

UML: Klassendiagramm
Erstellung, Schritt 1: Definition der Klassen

Ableitung der Klassen z.B. aus:

- Dokumentenanalyse:
  - Formulare, Listen
  - Re-Engineering: Funktionalität eines bestehenden Systems, Benutzerhandbücher, Bildschirmmasken
- Use-Case-Beschreibungen:
  - Beschreibung nach Klassen durchsuchen
- Verständnis der Domäne

UML: Klassendiagramm
Erstellung, Schritt 1: Definition der Klassen

Allgemeine Konventionen, analytische Schritte:

- Aussagekräftige Klassennamen nutzen:
  - Fachterminologie
  - (Meist) Substantiv im Singular
  - Soll dasselbe ausdrücken wie Gesamtheit der Attribute und Methoden
  - Soll nicht die Rolle beschreiben, die Klasse in Beziehung zu einer anderen einnimmt
- Verständlich für den Auftraggeber
- Klassen repräsentieren fachliche Konzepte der Anwendungsdomäne
- Keine Entwurfs- oder Implementierungsdetails
- Angemessenes Abstraktionsniveau (nicht zu viele (kleine) Klassen)

UML: Klassendiagramm
Erstellung, Schritt 1: Definition der Klassen

Häufige Fehlerquellen bei Klassenerstellung:

- Zu kleine Klassen
- Aus jedem Detail wird eine Klasse modelliert
- Klasse modelliert Benutzungsoberfläche/Entwurfs-/Implementierungsdetails

UML: Klassendiagramm
Erstellung, Schritt 1: Definition der Klassen

Beispiel:

UML: Klassendiagramm
Erstellung, Schritt 2: Assoziationen modellieren

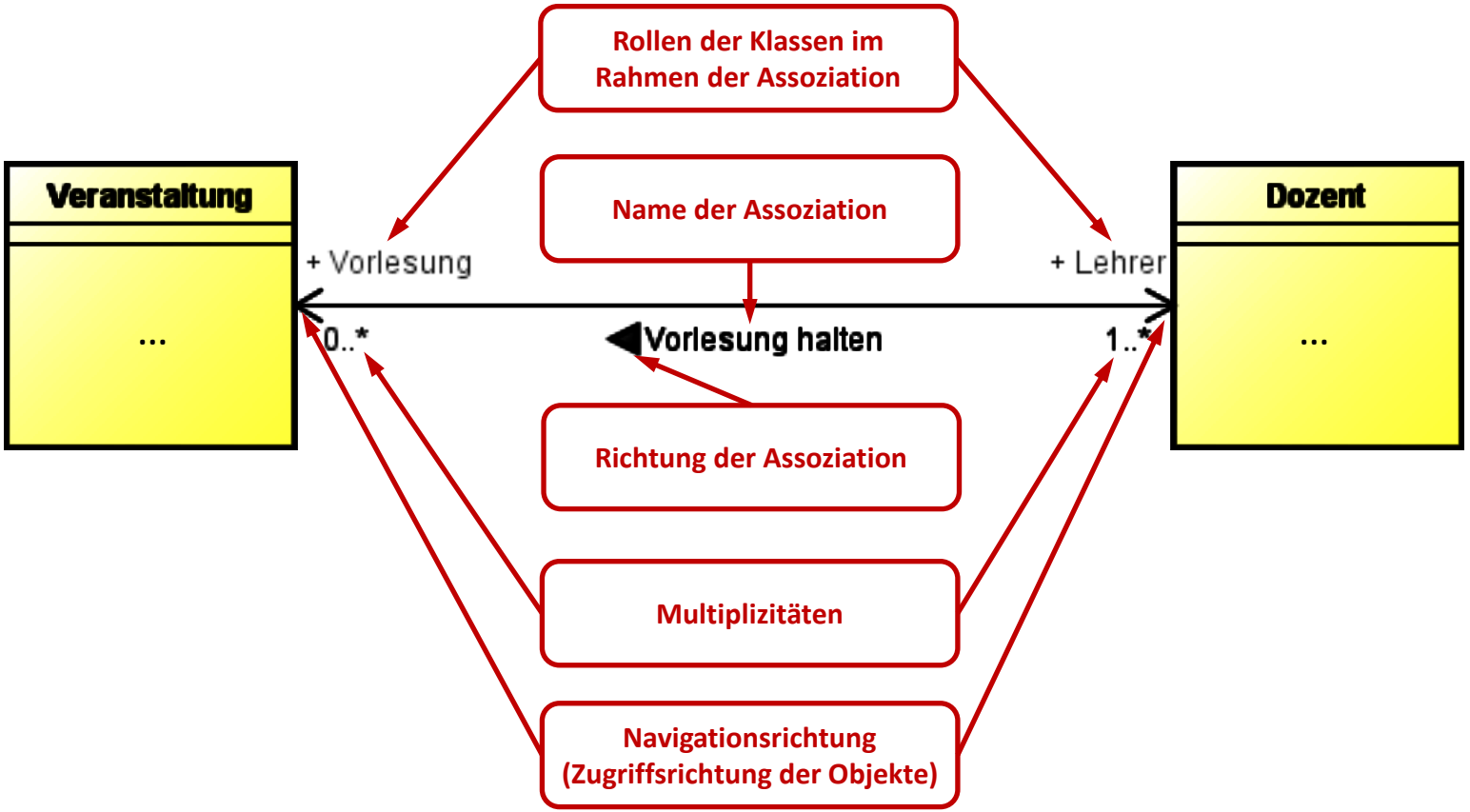
Nach der Identifikation der Klassen können die Verbindungen der Klassen über Assoziationen modelliert werden:

- Klassen identifizieren 

→ **Assoziationen modellieren**

UML: Klassendiagramm
Erstellung, Schritt 2: Assoziationen modellieren

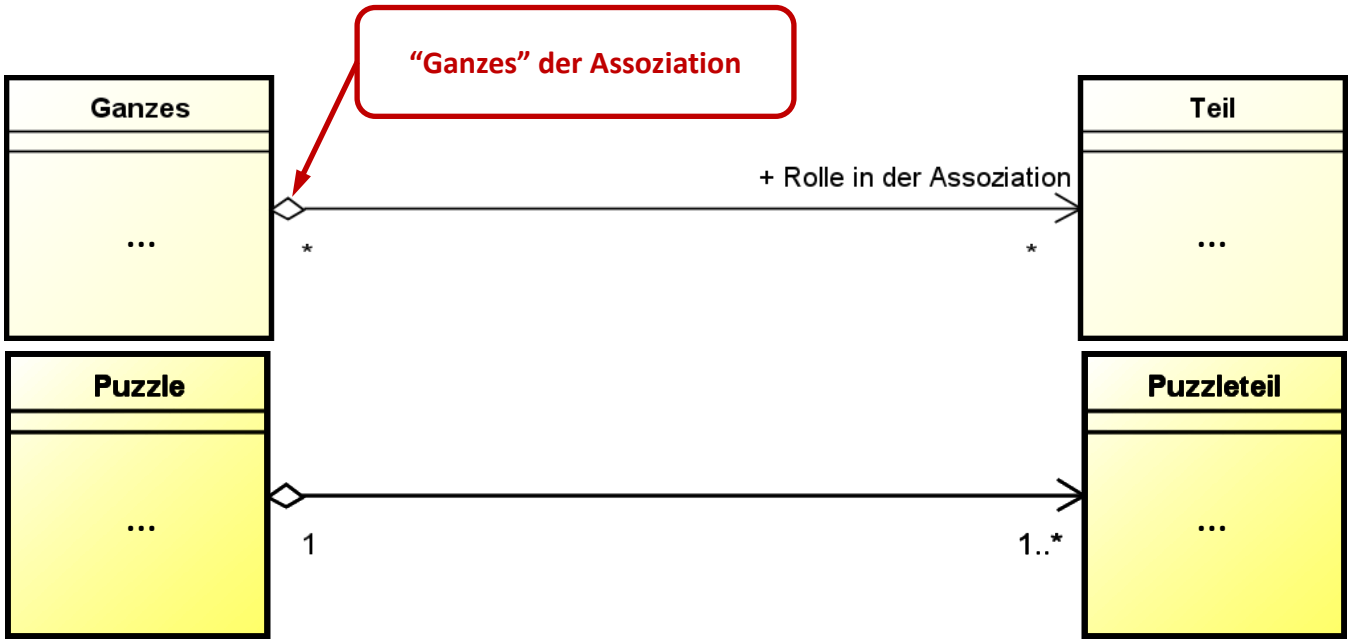
Assoziation: beschreibt Beziehung zwischen einer oder mehreren Klassen (genauer: zwischen den Objekten der beschriebenen Klassen)



UML: Klassendiagramm
Erstellung, Schritt 2: Assoziationen modellieren

Spezielle Assoziationen:

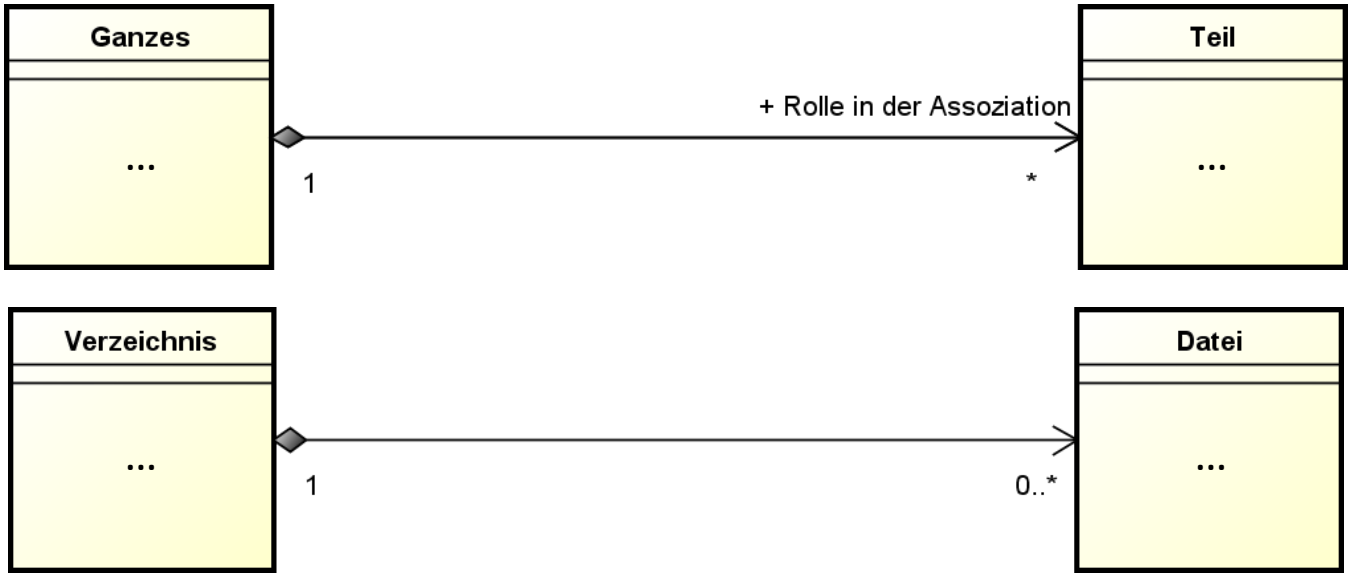
- Aggregation:
  - Modellierung einer Teile-Ganzes-Beziehung (“besteht aus”)
  - Navigierbarkeit: nur von “Ganzes” nach “Teile” möglich
  - Multiplizität: keine Einschränkungen



UML: Klassendiagramm
Erstellung, Schritt 2: Assoziationen modellieren

Spezielle Assoziationen:

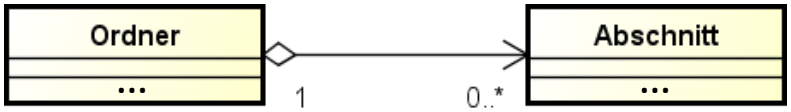
- Komposition:
  - Modellierung einer Beziehung eines Ganzen zu einem oder mehreren Teilen, bei der die Teile nicht ohne das Ganze existieren können
  - Wird also das Ganze zerstört, so werden auch alle Teile zerstört
  - Multiplizität 1 (auf Seite des Ganzen)
  - Strengere Form/Spezialfall der Aggregation



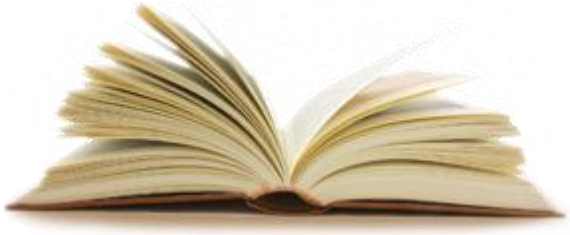
Beispiele?

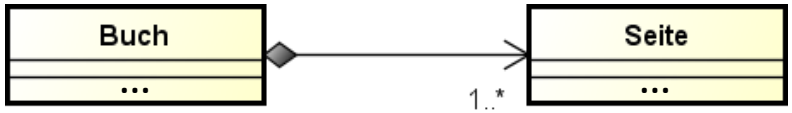
UML: Klassendiagramm
Erstellung, Schritt 2: Assoziationen modellieren

Vergleich: Aggregation **vs.** **Komposition**



- Schwache Beziehung zwischen Objekten
- Das Teil kann ohne das Ganze existieren





- Starke Beziehung zwischen Objekten
- Das Teil kann ohne das Ganze nicht existieren

UML: Klassendiagramm
Erstellung, Schritt 2: Assoziationen modellieren

Vorgehen bei der Assoziationsmodellierung:

- Assoziationen über Dokumentenanalyse identifizieren (bottom-up)
- Assoziationen über die Use Cases ermitteln (top-down)
- Multiplizitäten/Kardinalitäten der Assoziationen bestimmen
- Rollen der beteiligten Klassen feststellen
  - Rollennamen angeben, um Rolle der Klasse zu spezifizieren
- Assoziationen und Rollen benennen
  - Insbesondere, wenn mehrere Assoziationen zwischen zwei Klassen existieren
  - Oder wenn Rolle reflexiv ist (Assoziation von Klasse zu sich selbst)
- Auf “Teile-Ganzes”-Beziehung prüfen

UML: Klassendiagramm
Erstellung, Schritt 2: Assoziationen modellieren

Häufige Fehlerquellen bei Assoziationsmodellierung:

- Assoziation wird mit Vererbung verwechselt
- Aggregation und Komposition falsch verwendet (nur bei “part-of”-Beziehungen)

UML: Klassendiagramm
Erstellung, Schritt 2: Assoziationen modellieren

UML: Klassendiagramm
Erstellung, Schritt 3: Generalisierung modellieren

Nach Identifikation der Klassen und Modellierung der Assoziationen wird nun die Vererbung modelliert:

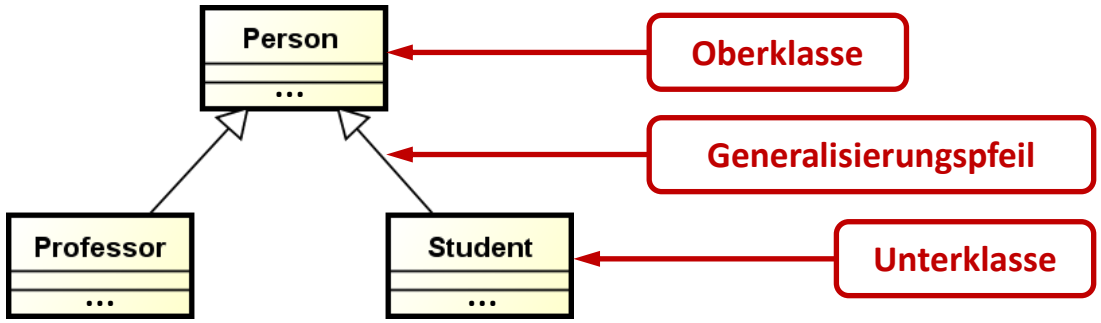
- Klassen identifizieren 
- Assoziationen modellieren 

→ **Vererbung modellieren**

UML: Klassendiagramm
Erstellung, Schritt 3: Generalisierung modellieren

Generalisierung: Drückt eine gerichtete Beziehung zwischen einer generelleren und einer spezialisierteren Klasse aus:

- Spezialisierte Klasse erbt Merkmale (Attribute, Operationen, Assoziationen) der generelleren Klasse
- Spezialisierte Klasse kann Eigenschaften und Verhalten hinzufügen
- Instanzen der spezialisierten Klasse können überall verwendet werden, wo Instanz der generelleren Klasse erlaubt ist (Substituierbarkeit)
- Mehrfachvererbung zulässig
- Spezialisierung kann auch auf Akteure, Assoziationen etc. angewandt werden



UML: Klassendiagramm
Erstellung, Schritt 3: Generalisierung modellieren

Vorgehen bei Generalisierung:

- Aus gleichartigen Klassen neue (evtl. abstrakte) Oberklasse bilden
- Prüfen, ob vorhandene Klasse als Oberklassen dienen kann
- Spezialisierung: prüfen, ob Substituierbarkeit gegeben ist
- Gemeinsamkeiten in der Oberklasse zusammenfassen

Prüfen auf eine gute Vererbungsstruktur:

- Substituierbarkeit gegeben?
- Wird Verständnis des Modells verbessert?
- Liegt eine “is-a”-Beziehung vor?
- Maximal drei bis fünf Hierarchiestufen
- Handelt es sich um eine natürliche Vererbungsstruktur?
- Wichtig: Klassenzugehörigkeit darf sich während der gesamten Lebenszeit nicht verändern

UML: Klassendiagramm
Erstellung, Schritt 3: Generalisierung modellieren

Häufige Fehlerquellen bei Vererbungsmodellierung:

- Unterklassen werden nur für Bezeichnung verschiedener Arten verwandt, in den Attributen (Eigenschaften) und Operationen (Verhalten) kein Unterschied

UML: Klassendiagramm
Erstellung, Schritt 3: Generalisierung modellieren

UML: Klassendiagramm Erstellung, Schritt 4: Attribute ergänzen

Nach der Modellierung der äußeren Struktur der Klassen wird nun begonnen, die innere Struktur zu modellieren:

- Klassen identifizieren 
- Assoziationen modellieren 
- Vererbung modellieren 

→ **Attribute ergänzen**

UML: Klassendiagramm Erstellung, Schritt 4: Attribute ergänzen

Vorgehen bei der Attributergänzung:

- Attribute über Dokumentenanalyse identifizieren (bottom-up)
- Attribute über Use Cases ermitteln (top-down)

Bei Attributidentifizierung prüfen:

- Ist Attributname geeignet? (kurz, eindeutig, verständlich, …)
- Liegt eine Klasse oder ein komplexes Attribut vor?
- Liegt geeignetes Abstraktionsniveau vor?
- Gehört das Attribut zu einer Klasse oder zu einer Assoziation?
- Schlüsselattribute nur dann einführen, wenn fachlich notwendig
- Werden abgeleitete Attribute richtig verwendet?

UML: Klassendiagramm Erstellung, Schritt 4: Attribute ergänzen

Häufige Fehlerquellen bei Attributidentifizierung:

- Verwenden atomarer Attribute statt komplexer Datenstrukturen
- Formulieren von Assoziationen als Attribut (Fremdschlüssel)
- Attribut beschreibt Implementierungs- und Entwurfsdetails

UML: Klassendiagramm
Erstellung, Schritt 4: Attribute ergänzen

Schema zur Definition von Klassenattributen:

```
Sichtbarkeit [/] Name [:Typ] [Multiplizität] [=Vorgabewert] (Eigenschaft)*
```

- Sichtbarkeit (des Attributs):
  - "+": public: Öffentlich; für alle Elemente sichtbar
  - "~": package: Sichtbar für alle Elemente im gleichen Paket
  - "#": protected: Sichtbar für Instanzen und Instanzen abgeleiteter Klassen
  - "-": private: Sichtbar nur für Instanzen dieser Klasse
- Typ (des Attributs):
  - Elementare Datentypen (z.B. je nach Programmiersprache: `int`, `String`, `double`, etc.)
  - Datentypen, die durch Klassen definiert sind

UML: Klassendiagramm Erstellung, Schritt 4: Attribute ergänzen

Schema zur Definition von Klassenattributen:

```
Sichtbarkeit [/] Name [:Typ] [Multiplizität] [=Vorgabewert] (Eigenschaft)*
```

- Multiplizität: (Anzahl der Instanzen, die unter diesem Attribut ablegbar sind)
  - Nutzung der Schreibweise: “[min,max]”
  - “[0..1]”: Optionalität
  - “[1..1]”=“[1]”: Standard (zwingendes Attribut)
  - “[0..*]”=“[*]”: 0 oder beliebig viele
  - “[1..N]”: Mindestens eins, höchstens N
- Vorgabewert (Angabe eines Standardwertes):
  - Konstante
  - Berechenbarer Ausdruck

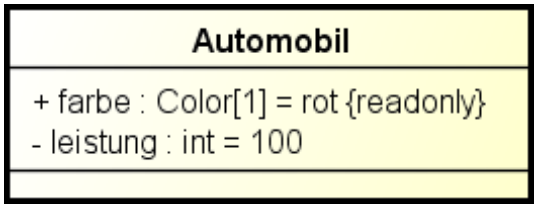
UML: Klassendiagramm
Erstellung, Schritt 4: Attribute ergänzen

Schema zur Definition von Klassenattributen:

```
Sichtbarkeit [/] Name [:Typ] [Multiplizität] [=Vorgabewert] (Eigenschaft)*
```

- /:
  - Attribut ist abgeleitet (kann zur Laufzeit aus anderen Attributen berechnet werden)
- Eigenschaft:
  - Beschreibt eine Eigenschaft des Attributs (z.B.: `unique`, `ordered`, `readonly`, `id`)

Beispiel:



Bedeutung?

UML: Klassendiagramm
Erstellung, Schritt 4: Attribute ergänzen

UML: Klassendiagramm Erstellung, Schritt 5: Operationen ergänzen

Nach Modellierung der äußeren Struktur und der Attribute der Klassen wird nun deren Verhalten (Operationen) festgelegt:

- Klassen identifizieren 
- Assoziationen modellieren 
- Vererbung modellieren 
- Attribute ergänzen 
- → **Operationen ergänzen**

UML: Klassendiagramm Erstellung, Schritt 5: Operationen ergänzen

Schema zur Definition von Klassenoperationen:

```
Sichtbarkeit Name (Parameterliste) : Rückgabetyp {Eigenschaften}
```

- Sichtbarkeit (der Operation):
  - "+": public: Öffentlich; für alle Elemente sichtbar
  - "~": package: Sichtbar für alle Elemente im gleichen Paket
  - "#": protected: Sichtbar für Instanzen und Instanzen abgeleiteter Klassen
  - "-": private: Sichtbar nur für Instanzen dieser Klasse

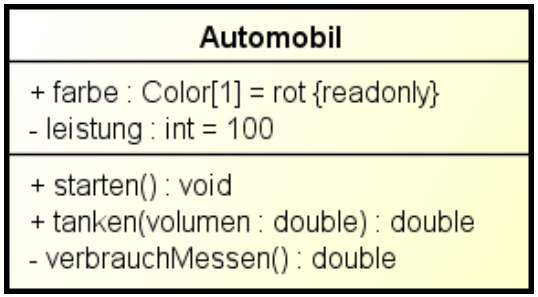
UML: Klassendiagramm
Erstellung, Schritt 5: Operationen ergänzen

Schema zur Definition von Klassenoperationen:

```
Sichtbarkeit Name (Parameterliste) : Rückgabetyp {Eigenschaften}
```

- Rückgabetyp (der Operation):
  - Datentyp, Klasse, `void`
- Eigenschaften (Einschränkungen):
  - Preconditions (Zu erfüllende Bedingungen zur Aktivierung der Operation)
  - Postconditions (Zusicherung über Zustand des Systems nach der Ausführung)
  - Body conditions (Einschränkungen des Rückgabewerts)

Beispiel:



UML: Klassendiagramm Erstellung, Schritt 5: Operationen ergänzen

Schema zur Definition von Klassenoperationen (Parametern):

```
[Richtung] Name : Typ [Multiplizität] [= Defaultwert] {Eigenschaften}
```

- Richtung (Übergaberichtung):
  - "in": Wert des Parameters wird von aufrufender Methode übergeben
  - "out": Wert des Parameters wird an die aufrufende Methode zurückgegeben
  - "inOut": Erst "in" dann "out"
- Name/Typ/Multiplizität/Eigenschaften:
  - Analog zu Attributen
- Defaultwert:
  - Wert des Parameters, falls kein Wert explizit übergeben wurde

UML: Klassendiagramm Erstellung, Schritt 5: Operationen ergänzen

Vorgehen bei der Operationendefinition:

- Operationen aus dynamischen Modellen eintragen:
  - Alle Operationen aus den Sequenz- und Kommunikationsdiagrammen, Zustandsautomaten
  - Für Aktivitäten in Zustandsautomaten prüfen, durch welche Operationen sie realisiert werden
- Verwaltungsoperationen nicht eintragen:
  - `new()`, `delete()` etc. nicht ins Klassendiagramm eingetragen
- Operationen so hoch wie möglich in der Hierarchie eintragen
- Beschreibung der Operationen erstellen:
  - Bei einfachen Operationen informal, d.h. Text
  - Bei komplexeren Operation auch semiformale Spezifikation, z.B. Aktivitätsdiagramm, Zustandsdiagramm

UML: Klassendiagramm Erstellung, Schritt 5: Operationen ergänzen

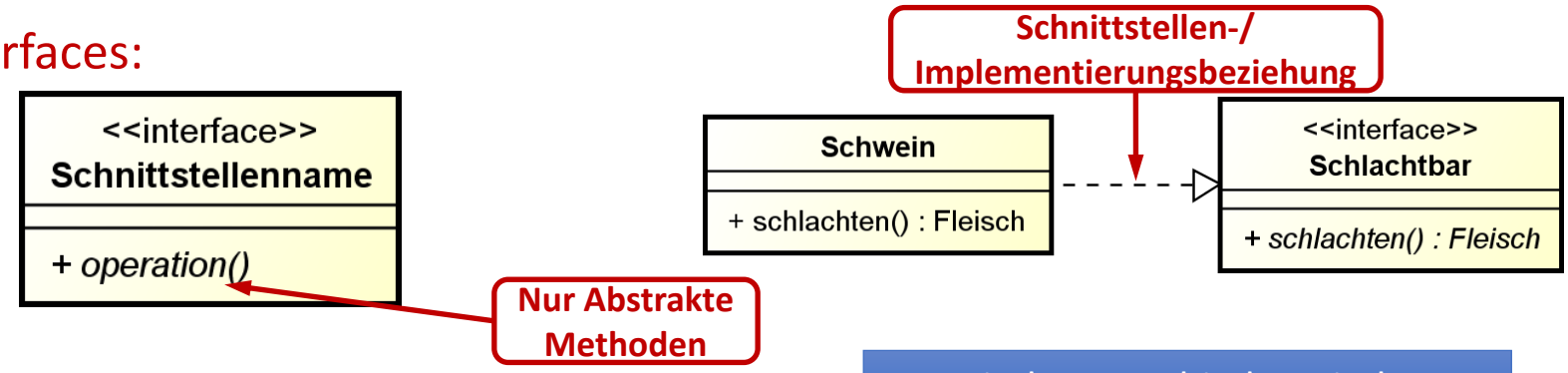
Analytische Schritte:

- Besitzt Operation einen geeigneten Namen:
  - Beginnt mit Verb
  - Beschreibt, was Operation tut
- Erfüllt Operation die geforderten Qualitätskriterien:
  - Angemessener Umfang (z.B. nicht zu umfangreich)
  - Funktionale Bindung (z.B. in sich abgeschlossene Funktion)
- Ist Balancing erfüllt:
  - Alle Attribute der Klasse werden von den Operationen benötigt.

UML: Klassendiagramm
Erstellung, Schritt 5: Operationen ergänzen

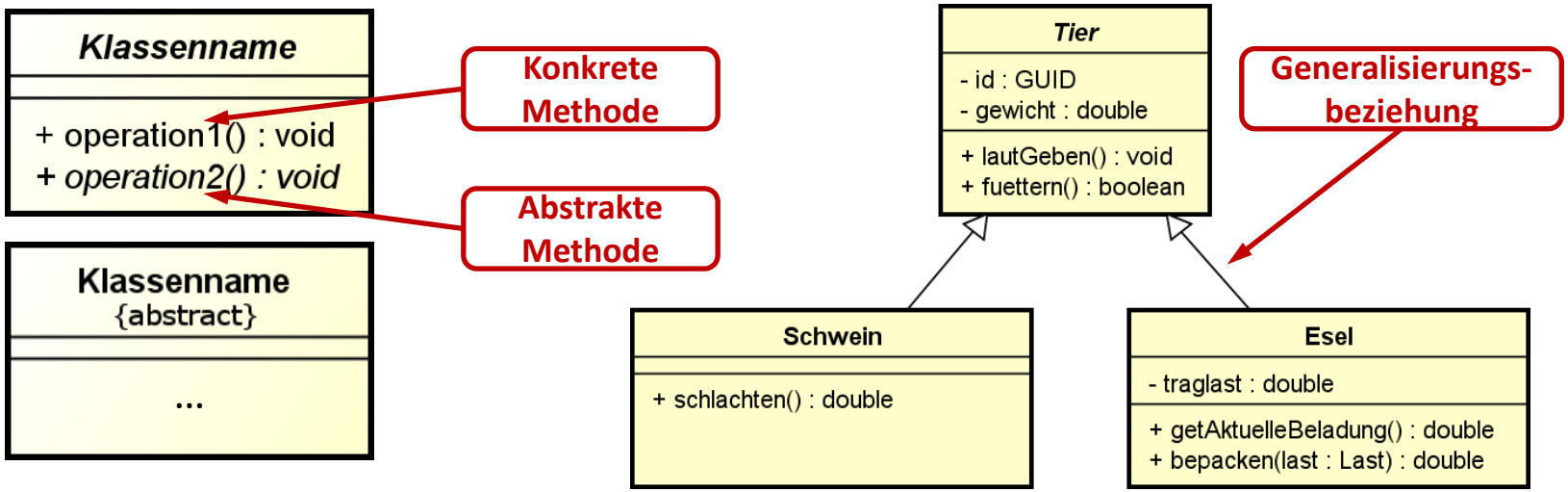
UML: Klassendiagramm
Weiterführende Konzepte

- Interfaces:



Was sind Unterschiede zwischen abstrakten Klassen und Interfaces?

- Abstrakte Klassen:



UML: Klassendiagramm Beispiel

Erstellen Sie ein Klassendiagramm für das Thema: “Bachelorarbeit schreiben”:

Das Klassendiagramm soll folgende (abstrakte) Klassen/Interfaces enthalten:

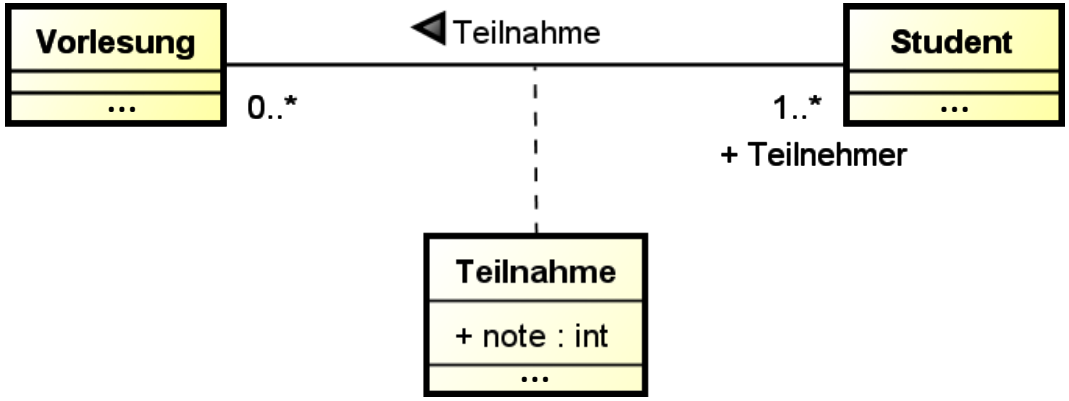
- Professor
- Person
- Bachelorarbeit
- Studierender

Außerdem sollen Sie Attribute, Methoden und deren Sichtbarkeiten berücksichtigen.

UML: Klassendiagramm
Weiterführende Konzepte

Assoziationsklassen:

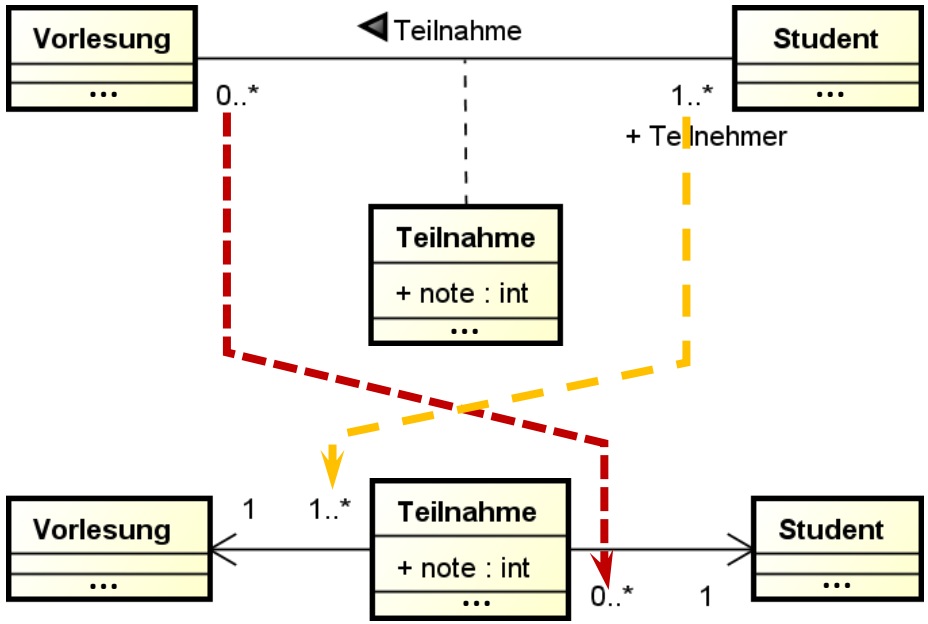
- Modellierung von Eigenschaften, die nicht direkt einem der Partner einer Assoziation zugeordnet werden können, sondern zur Assoziation selbst gehören
- Hat also Eigenschaften einer Klasse und gleichzeitig einer Assoziation
- Einschränkung: Zwei beteiligte Objekte dürfen maximal eine Beziehung zueinander haben



UML: Klassendiagramm
Weiterführende Konzepte

Assoziationsklassen:

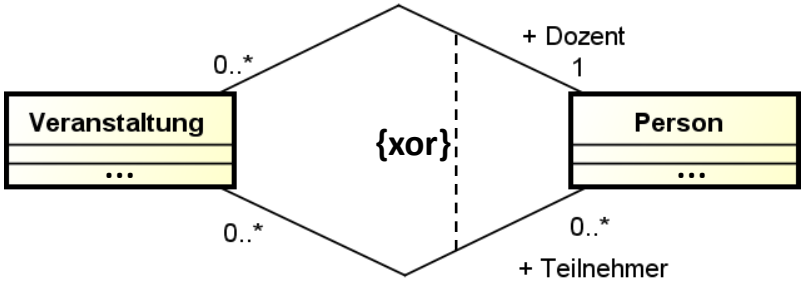
- Problem: können nicht natürlich in eine OOP-Sprache überführt werden
- Hinweis: daher sollten Assoziationsklassen nicht verwendet werden! **Lösung?**
- Lösung: Überführung der Assoziationsklasse in eine normale Klasse (semantische Einschränkungen über Zusicherungen definieren)



UML: Klassendiagramm
Weiterführende Konzepte

Semantische Einschränkungen von Assoziationsenden:

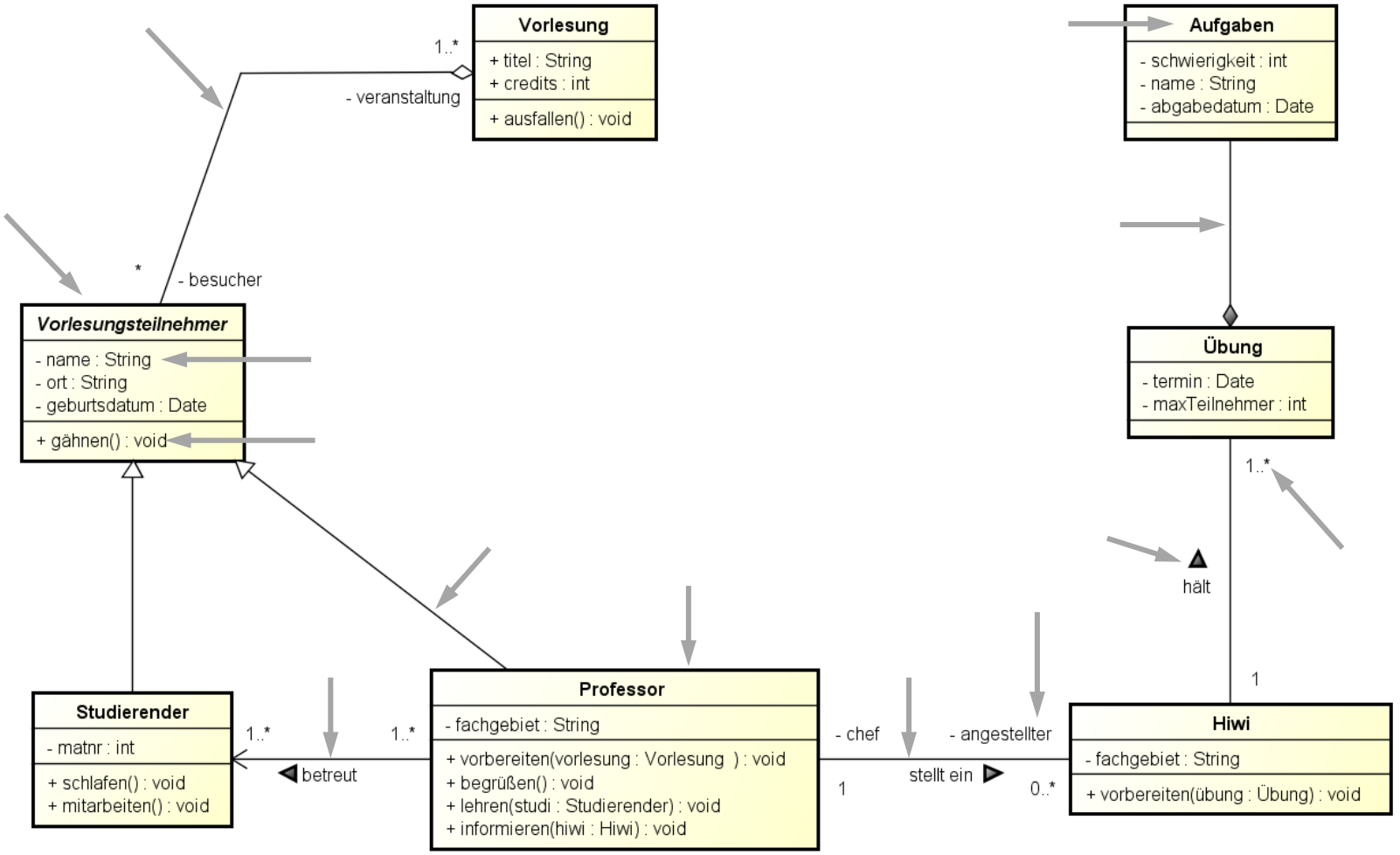
- {xor}: Objekte verschiedener Klassen stehen in “entweder-oder”-Beziehung. (Zu einem Zeitpunkt kann also nur eine der Assoziationen gültig sein)



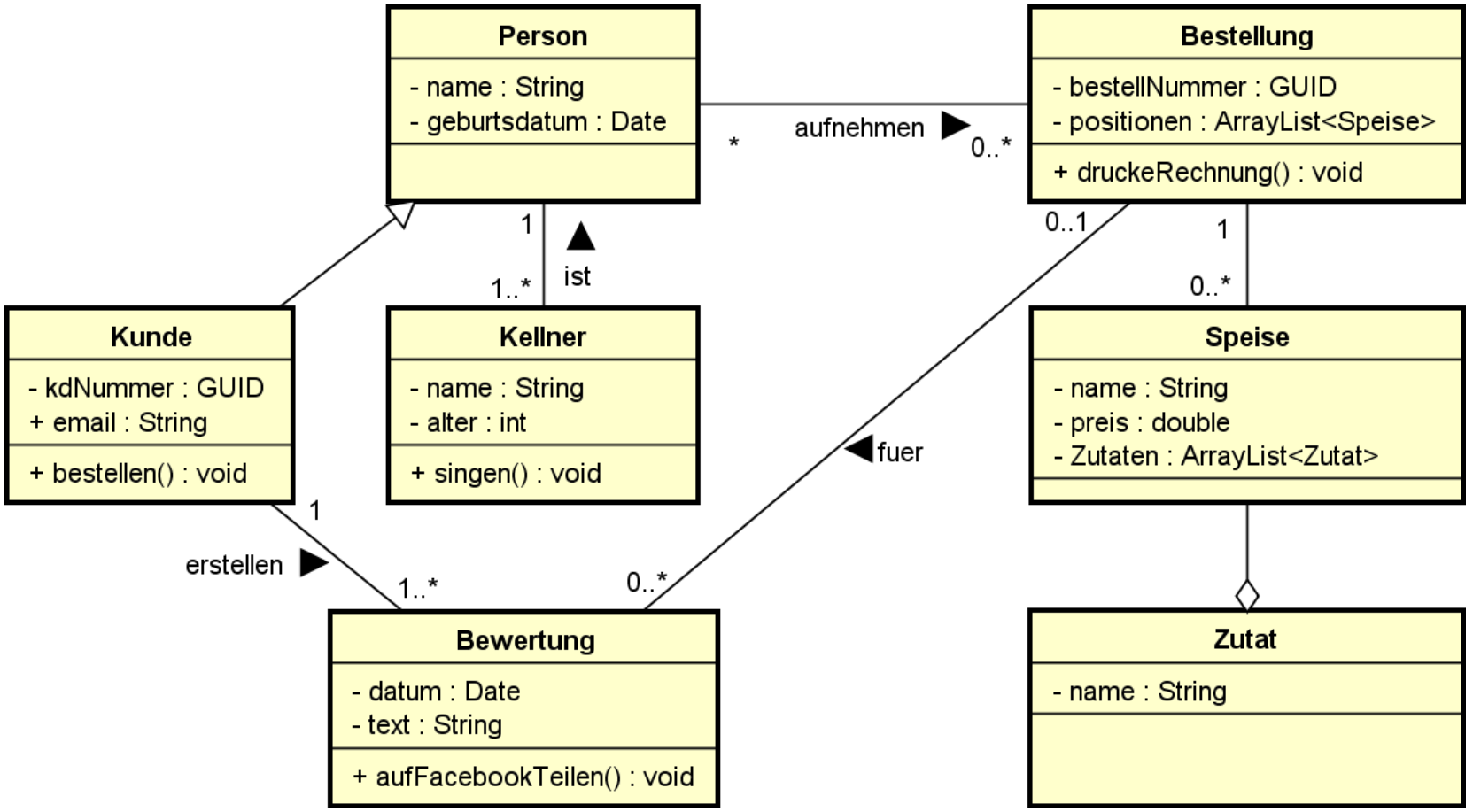
Bedeutung:

- Eine Veranstaltung wird von genau einem Dozenten abgehalten
- Ein Dozent kann beliebig viele Veranstaltungen halten
- Eine Veranstaltung hat beliebig viele Teilnehmer
- Eine Person kann an beliebig vielen Veranstaltungen teilnehmen
- Eine Person kann entweder als Teilnehmer oder Dozent mit einer Veranstaltung in Beziehung stehen

Zusammenfassendes Beispiel: Klassendiagramme



Finde die Fehler



Literatur

- *UML 2 glasklar*, Chris Rupp et al., Hanser, 2012