Inhaltsverzeichnis

Rückblick – Erstellung OOA-Modell

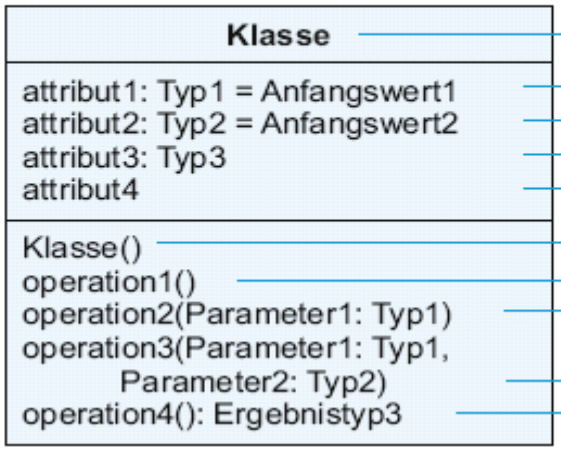
- Möglicher Ablauf
  - Ermittlung der relevanten Geschäftsprozesse und Use Cases
    - Use Case-Diagramme, Aktivitätsdiagramme
  - Erstellen des statischen Modells
    - Ableiten von Klassen aus den Use Cases
    - Klassen- und Objektdiagramme
  - Erstellung des dynamischen Modells
    - Szenarien erstellen
      (jeden Use Case durch eine Menge von Szenarien präzisieren)
    - Sequenz- und Kommunikationsdiagramme, Zustandsdiagramme, Aktivitätsdiagramme
  - Berücksichtigung der Wechselwirkung beider Modelle

Bedeutung des Klassendiagramms

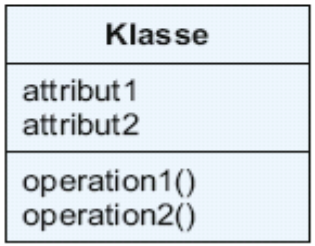
- Ziel
  - Visualisierung der strukturellen Zusammenhänge zwischen Klassen
- Verwendung
  - Analyse
    - Identifizierung der wichtigsten fachlichen Gegenstände, die vom zu entwickelnden System repräsentiert werden sollen
    - Fachklassenmodell zur Diskussion mit den Experten aus dem Anwendungsbereich
    - Strukturierung und Erläuterung des Anwendungsbereichs
  - Entwurf
    - Diskussion von Entwurfsalternativen
    - Dokumentation von Entwurfsentscheidungen
    - Vorgabe für Implementierung

Von UML-Klasse zu Java-Klassengerüst

UML-Klasse



Hinweis:
Beim Zeichnen von UML-Klassen werden oft nur die Attribute und Operationsnamen angegeben.
In Java müssen die restlichen Angaben dann ergänzt werden:



Java-Klasse

```
class Klasse
{ //Attribute
  private Typ1 attribut1 = Anfangswert1;
  private Typ2 attribut2 = Anfangswert2;
  private Typ3 attribut3;
  private Typ4 attribut4; //Typ4 muss
  ergänzt werden
  //Konstruktor(en)
  Klasse()
  { //Standardkonstruktor
  }
  //Operationen (Methoden)
  void operation1()
  { //Anweisungen der Operation1 ergänzen
  }
  void operation2(Typ1 Parameter 1)
  { //Anweisungen der Operation2 ergänzen
  }
  void operation3( Typ1 Parameter1,
    Typ2 Parameter2)
  { //Anweisungen der Operation3 ergänzen
  }
  Ergebnistyp3 operation4()
  { //Anwesiungen der Operation4 ergänzen
  }
} //Ende der Klasse
```

Hinweis: Zur Sicherstellung des Geheimnisprinzips ist vor die Attribute `private` zu schreiben.

Quelle: Balzert, H. (2009): Lehrbuch der Softwaretechnik, S. 133

Objekt

- Bedeutung
  - Objekt ist eigenständige, abgeschlossene, abstrakte Einheit
- Beispiele
  - Physikalische, reale Entität
    - Z.B. Person, Fahrzeug, Dokument
  - Konzeptionelle Identität
    - Abstraktion eines Vorgangs
      - z.B. betriebswirtschaftlicher Prozess
    - Abstrakte Dinge/Begriffe
      - z.B. Schulung, Lebensgemeinschaft
  - Softwareentität
    - Z.B. Datenstruktur

Objekt – Eigenschaften

- Konkretes Objekt
  - Eigenschaften
  - Dynamisches Verhalten
    - Methoden; legen fest auf welche Nachrichten ein Objekt reagiert
    - Interaktion mit anderen Objekten
  - Identität (systemweit eindeutig identifizierbar)
- Zustand
  - Dateninhalte des Objekts (Attribute und deren Werte)
  - Beziehungen des Objekts zu anderen Objekten

Klasse

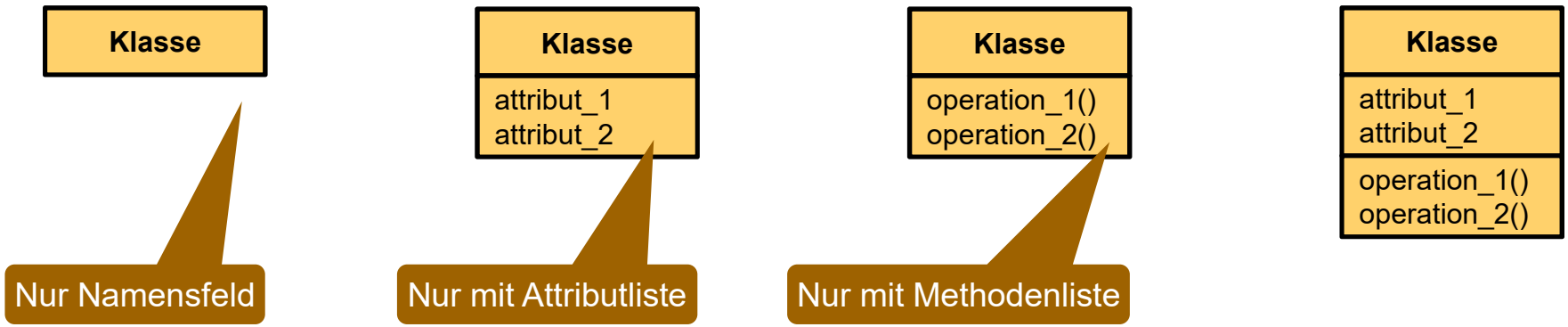
- Bedeutung
  - Vorlage / Muster für eine Menge von Objekten mit gemeinsamen Attributen, Verhalten, Beziehungen und Semantik
  - Klasse ist **nicht** eine Menge von Objekten
- Klasse ist Abstraktion
  - Muster für Objekte
  - Relevante Charakteristiken werden betont
  - Strukturierung von Daten und Verhalten
- Objekt ist Konkretisierung
  - Objekt wird von Klasse abgeleitet (Instanz einer Klasse)
  - Objekt ändert seine Klasse zur Laufzeit nicht

Objekte und Klassen im Softwaresystem

- Objektorientiertes Softwaresystem
  - Bei Erstellung / zur Übersetzungszeit
    - Besteht aus einer Sammlung von Klassen
    - Klasse ist benannte Programmeinheit
    - Klassenstruktur eines Softwaresystems ist statisch
  - Zur Laufzeit
    - Softwaresystem besteht aus Objekten, von Klassen abgeleitet
    - Objekt kapselt Daten; Zugriff über Methoden der zugehörige Klasse
    - Jede Systemaktivität ist Aktivität eines Objekts
    - Objekte bestimmen Dynamik des Programmablaufs

Klassen in der UML

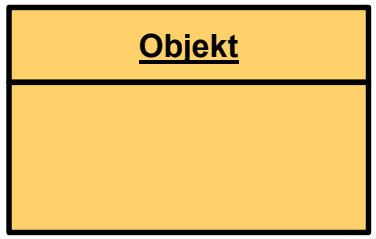
- Repräsentation
  - Rechteck mit drei Bereichen
    - Klassenname
    - Attribute
    - Methoden
  - Sind Attribute und Operationen noch nicht festgelegt oder im Moment nicht relevant, können sie weggelassen werden



In Anlehnung an: Balzert, H 2009.: Lehrbuch der Softwaretechnik, S. 132

Objekte in der UML

- Repräsentation
  - Rechteck, das in zwei Felder aufgeteilt werden kann
  - Bezeichnung eines Objekts wird unterstrichen
  - Anonyme und bestimmte (mit Namen) Objekte



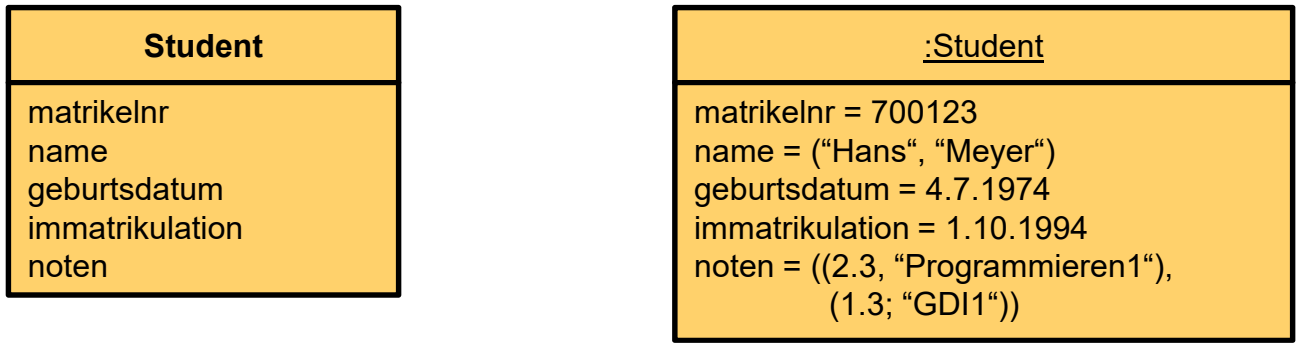
- Drei Varianten
  - Objektname und Klassenname

  - Nur Objektname (Name der Klasse aus Kontext ersichtlich)

  - Nur Klassenname (Anonymes Objekt)

Attribut (1)

- Definition
  - Attribute beschreiben die Daten, die von den Objekten einer Klasse angenommen werden können
  - Jedes Attribut ist von einem bestimmten Datentyp (primitiv oder Referenz auf anderes Objekt)
  - Alle Objekte einer Klasse besitzen dieselben Attribute
- Beispiel

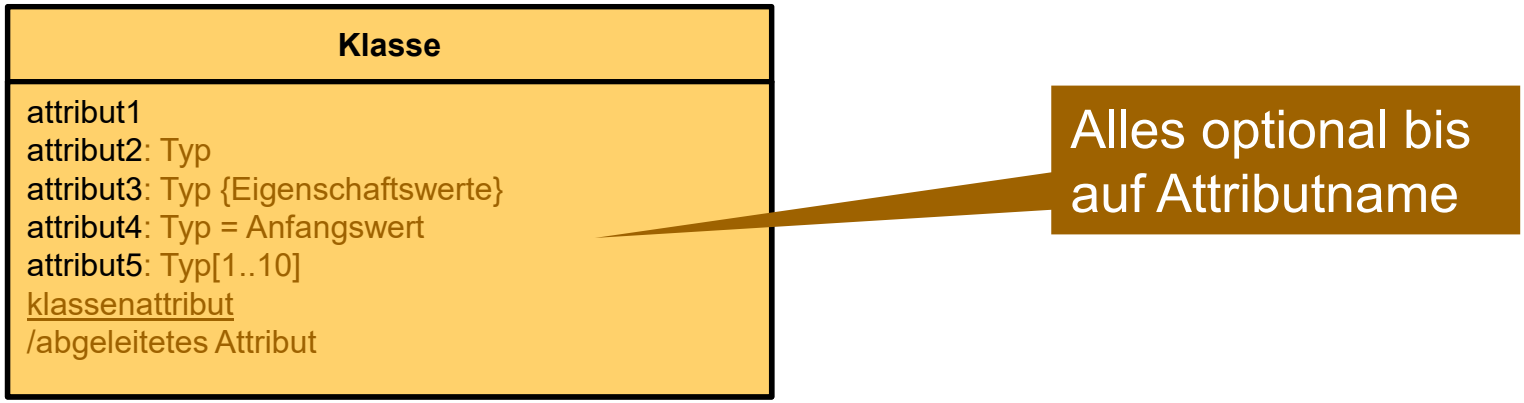


Klasse Student

Student-Objekt

Attribut (2)

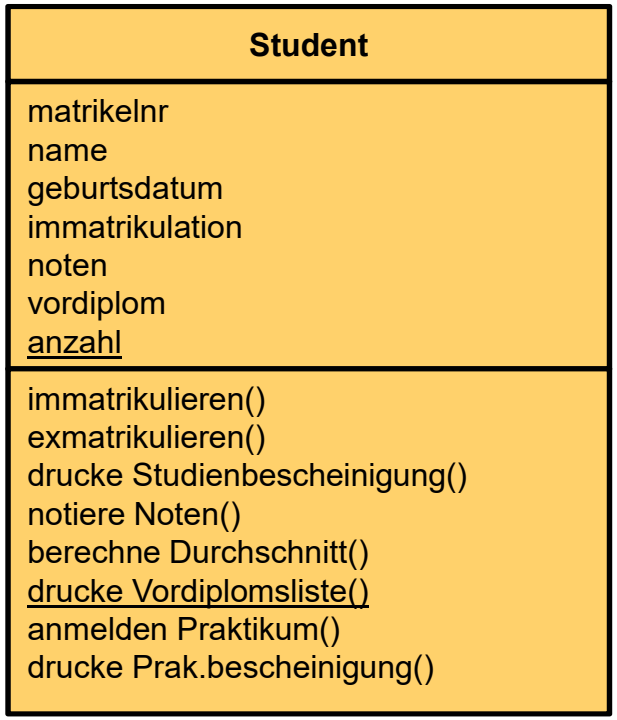
- Notation für Attribute



  - Abgeleitetes Attribut
    - Wert kann jederzeit aus anderen Attributwerten berechnet werden (darf nicht geändert werden)
  - Klassenattribut: unterstrichen
    - Ein Attributwert existiert nur für alle Objekte einer Klasse
    - Bsp.: Zähler für Anzahl „lebender“ Objekte

Operation (1)

- Definition
  - Ist eine ausführbare Tätigkeit
  - Alle Objekte einer Klasse verwenden dieselben Operationen
  - Jede Operation kann auf alle Attribute eines Objekts dieser Klasse direkt zugreifen
  - Menge aller Operationen wird als das Verhalten der Klasse bezeichnet
- Notation
  - Operationen werden analog zu den Attributen in das Klassendiagramm eingetragen



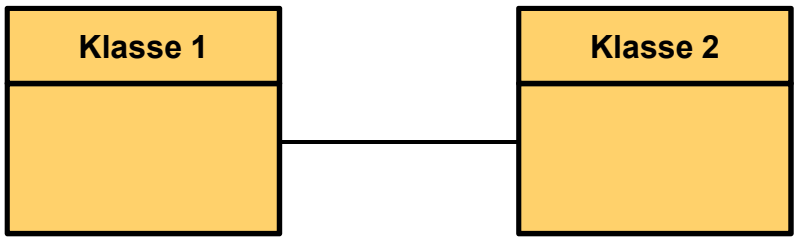
Klasse Student

Operation (2)

- Drei Arten von Operationen
  - Objektoperationen
    - Werden stets auf ein einzelnes (bereits existierendes) Objekt angewendet
  - Konstruktoroperationen
    - Erzeugt ein neues Objekt und führt entsprechende Initialisierungen und Datenerfassungen durch
  - Klassenoperationen
    - Operation, die der jeweiligen Klasse zugeordnet ist und nicht auf ein einzelnes Objekt der Klasse angewendet werden kann
    - Operation manipuliert Klassenattribut
    - Operation bezieht sich auf alle oder mehrere Objekte der Klasse
    - Unterstrichen dargestellt

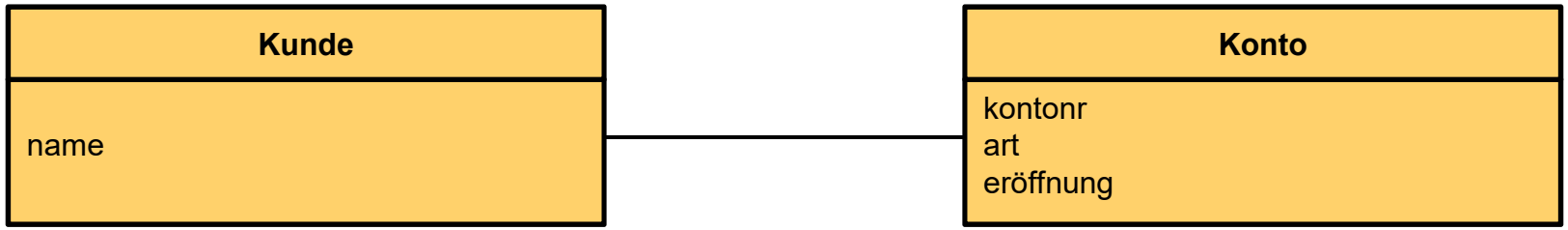
Assoziation (1)

- Definition
  - Beschreibung der Beziehungen zwischen den Objekten von Klassen
  - Objektbeziehungen sind Exemplare einer Assoziation
  - Objektbeziehung
    - Mindestens zwei Objekte müssen beteiligt sein
    - Möglich: beide Objekte gehören zur selben Klasse (reflexive Assoziation)
- Notation
  - Linie zwischen zwei Klassen im Klassendiagramm

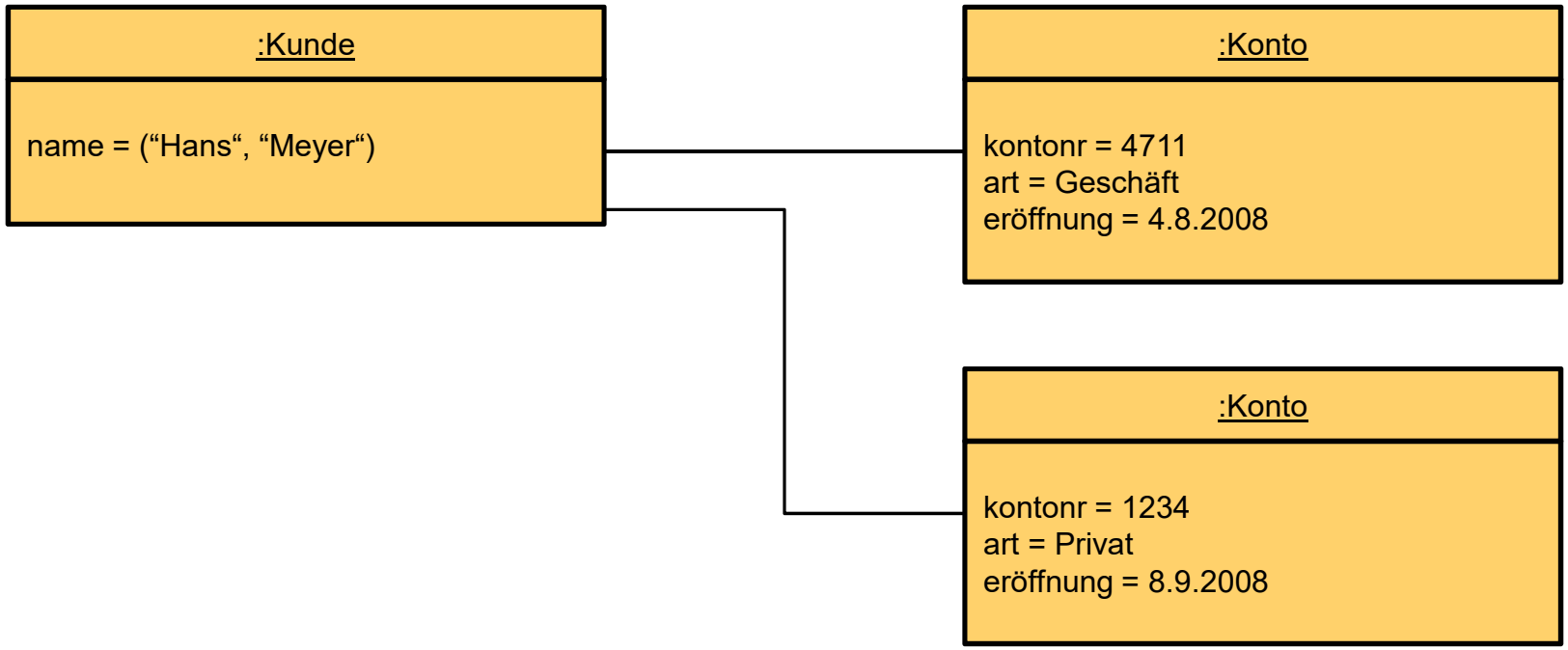


Beispiel Assoziation

- Klassendiagramm



- Objektdiagramm



Objektdiagramm – Beispiel

Identifizieren Sie anhand der folgenden Beschreibung Objekte und deren Objektbeziehungen und stellen Sie diese in einem Objektdiagramm dar.

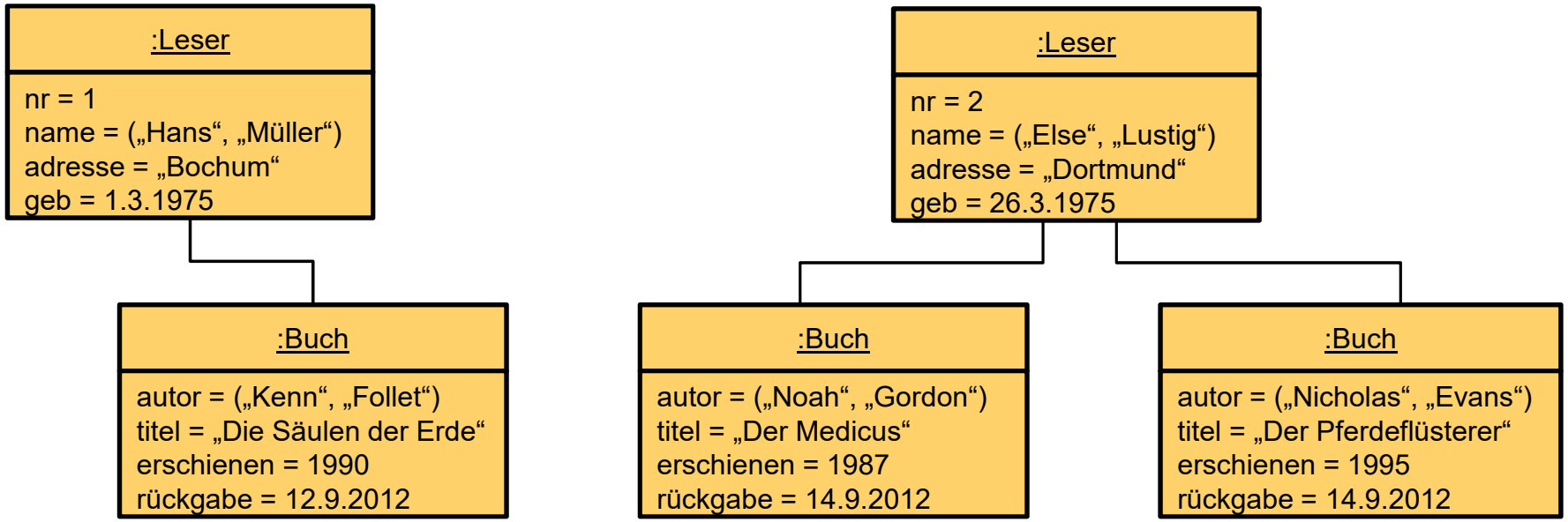
In einer Bibliothek sind die Regale voller Bücher. Dort stehen beispielsweise

- Kenn Follett, Die Säulen der Erde, 1990
- Noah Gordon, Der Medicus, 1987
- Nicholas Evans, Der Pferdeflüsterer, 1995

Für jeden Leser werden Namen, Adresse und Geburtsdatum gespeichert. Außerdem erhält jeder Leser eine Nummer.

Hans Müller, geb. am 1.3.1975 und wohnhaft in Bochum, leiht sich „Die Säulen der Erde“ aus. Spätestens am 12.9.2012 muss er es zurückgeben. Dieses Rückgabedatum wird ins Buch eingetragen.

Else Lustig aus Dortmund, geb. am 26.3.1975, leiht sich „Der Medicus“ und „Der Pferdeflüsterer“ aus. Beide Bücher muss sie am 14.9.2012 zurückgeben.



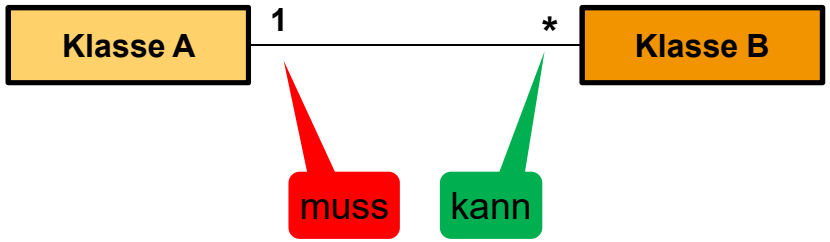
Assoziation (2)

- Multiplizitäten
  - Ein Objekt kann zu keinem anderen Objekt,
  - zu genau einem anderen Objekt oder
  - zu vielen Objekten eine Beziehung haben.
  - Spezifizieren die Wertigkeit, d.h. wie viele andere Objekte ein bestimmtes Objekt kennen kann oder muss.

- Notation
  - An jedem Ende der Assoziationslinie angegeben
  - Multiplizität, die zu einer Klasse A gehört, wird bei der Klasse B eingetragen und umgekehrt

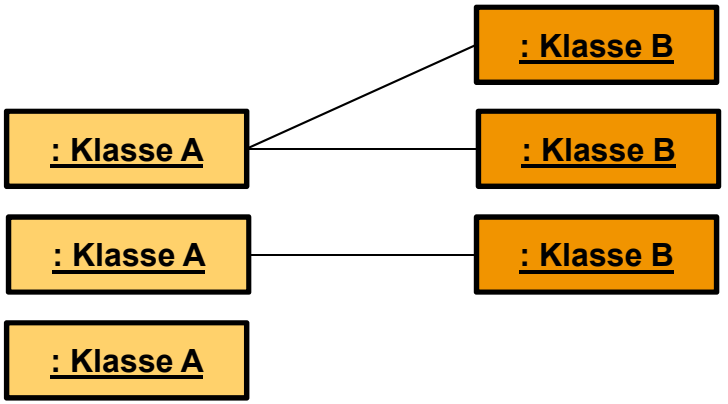
Assoziation (3)

- Beispiele für mögliche Multiplizitäten

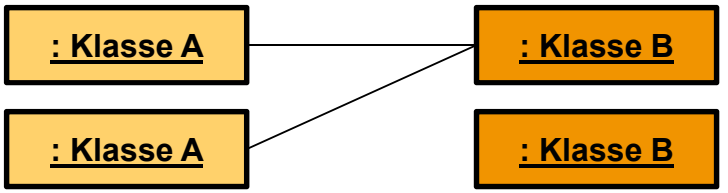


  - Objekte der Klasse A können Beziehungen zu 0, 1 oder mehreren Objekten der Klasse B haben (Notation: *).
  - Objekte der Klasse B müssen eine Beziehung zu genau einem Objekt der Klasse A haben (Notation: 1).

Mögliche Objektkonstellationen



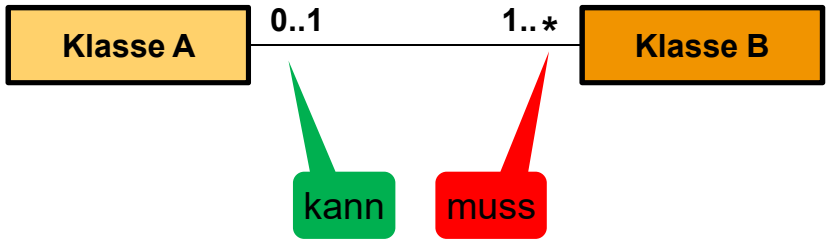
Unmögliche Objektkonstellationen



Quelle: Balzert, H. (2009): Lehrbuch der Softwaretechnik, S. 159

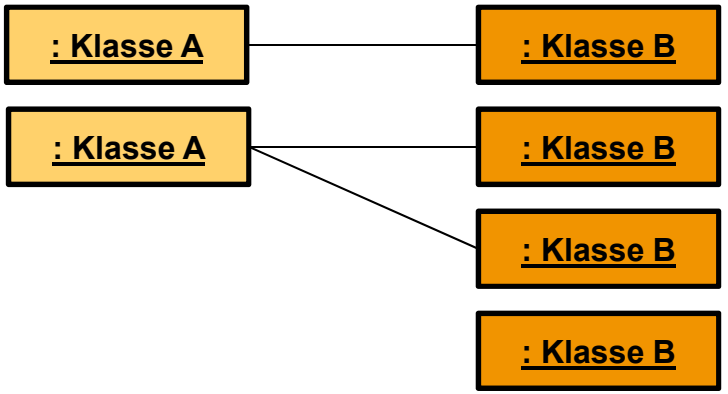
Assoziation (4)

- Beispiele für mögliche Multiplizitäten

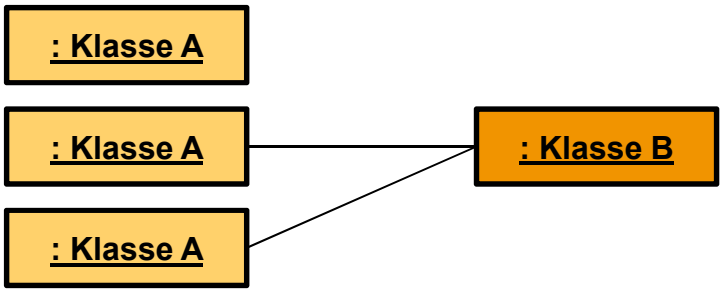


  - Objekte der Klasse A müssen zu 1 oder mehreren Objekten der Klasse B Beziehungen haben (Notation: 1..*).
  - Objekte der Klasse B können zu keinem oder zu genau 1 Objekt der Klasse A eine Beziehung haben (Notation: 0..1).

Mögliche Objektkonstellationen



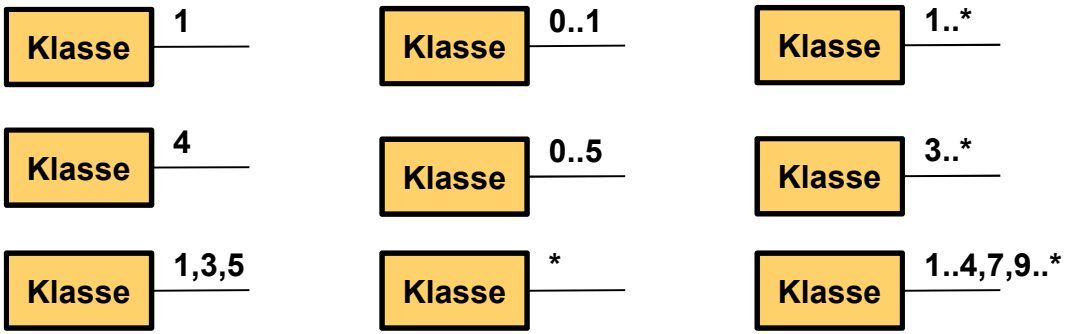
Unmögliche Objektkonstellationen



Quelle: Balzert, H. (2009): Lehrbuch der Softwaretechnik, S. 159

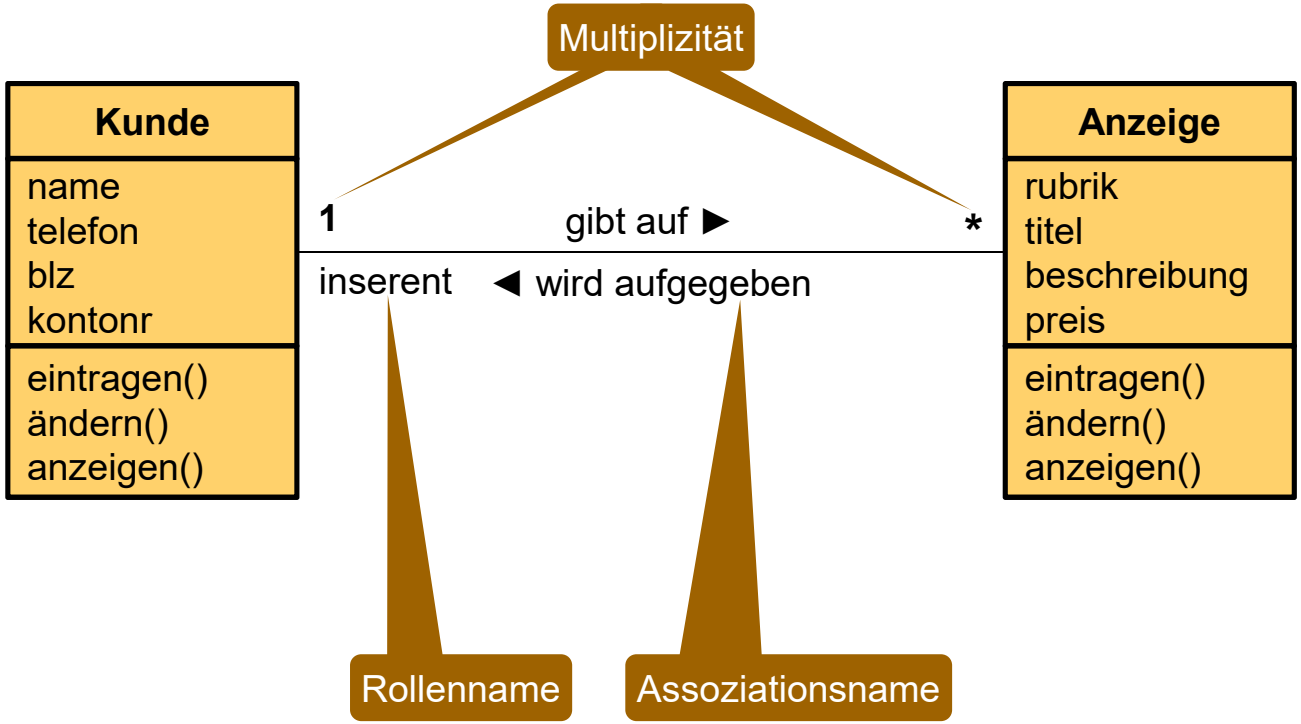
Assoziation (5)

- Multiplizitäten
  - Beispiele



- Assoziationsname
  - Assoziationen können benannt werden – Beschreibung der Semantik
  - Assoziationsname beschreibt i.A. nur eine Richtung der Assoziation, wobei ein schwarzes Dreieck die Leserichtung angibt
- Assoziationsrolle
  - Enthält Informationen über die Bedeutung einer Klasse
  - Wird jeweils bei der Klasse (Ende der Assoziation) geschrieben, deren Bedeutung in der Assoziation sie näher beschreibt

- Assoziationsnamen und –rollen



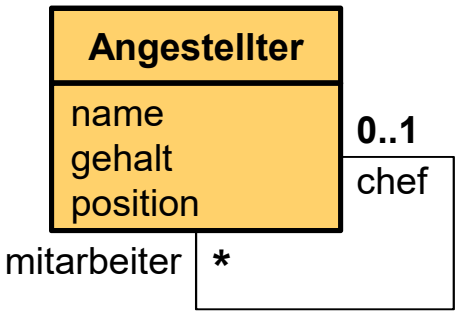
In Anlehnung an: Balzert, H. (2009): Lehrbuch der Softwaretechnik, S. 160

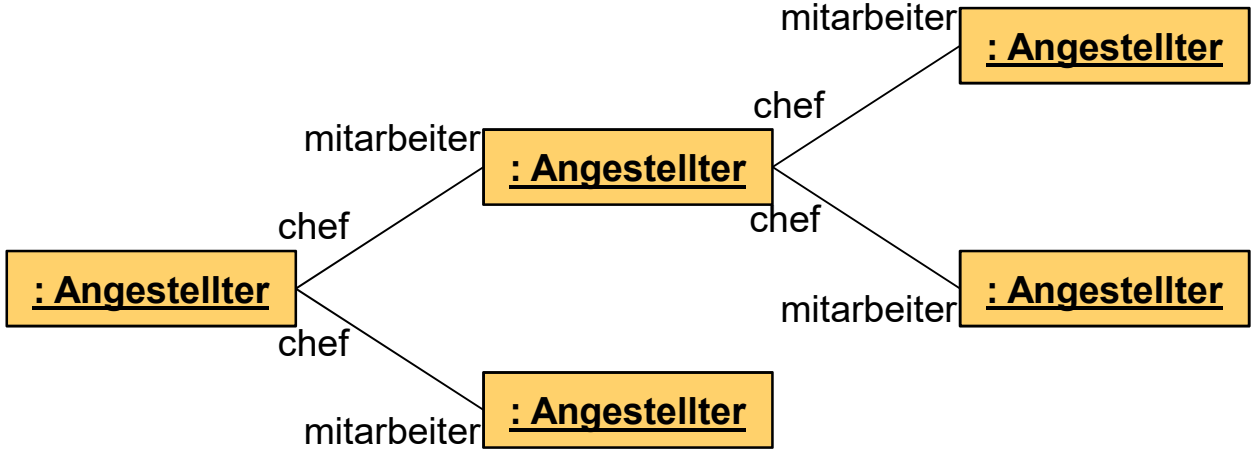
Aufgabe

- Lesen Sie folgende Assoziationen!

Assoziation (7)

- Reflexive Assoziation
  - Beziehung zwischen zwei Objekten der gleichen Klasse
  - Rollennamen! → damit die Verständlichkeit gewährleistet ist

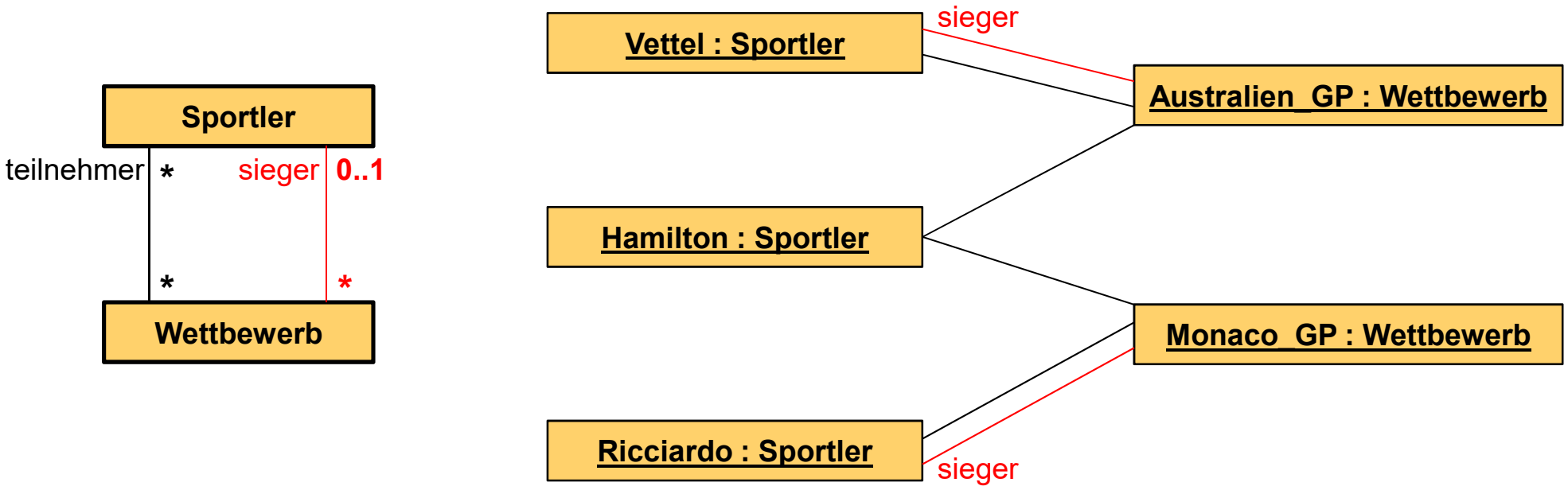




Quelle: Balzert, H. (2009): Lehrbuch der Softwaretechnik, S. 161

Assoziation (8)

- Mehrere Assoziationen zwischen zwei Klassen
  - Rollennamen oder Assoziationsnamen müssen angegeben werden!



In Anlehnung an: Balzert, H. (2009): Lehrbuch der Softwaretechnik, S. 161

„Hierarchische“ Assoziation

- Eine Assoziation besteht zwischen zwei gleichrangigen Klassen

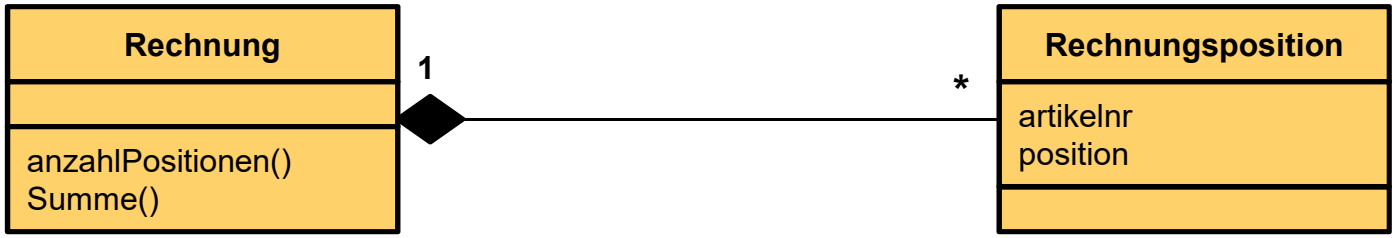
- Zwei Spezialfälle
  - Aggregation
  - Komposition

Aggregation

- Liegt vor, wenn zwischen den Objekten der beteiligten Klassen eine Rangordnung gilt, die sich durch „ist Teil von“ bzw. „besteht aus“ beschreiben lässt
  - Ganzes-Teile-Beziehung (whole-part)
  - Ganzes (Aggregat) übernimmt bestimmte Aufgaben für seine Teile
  - Beispiel

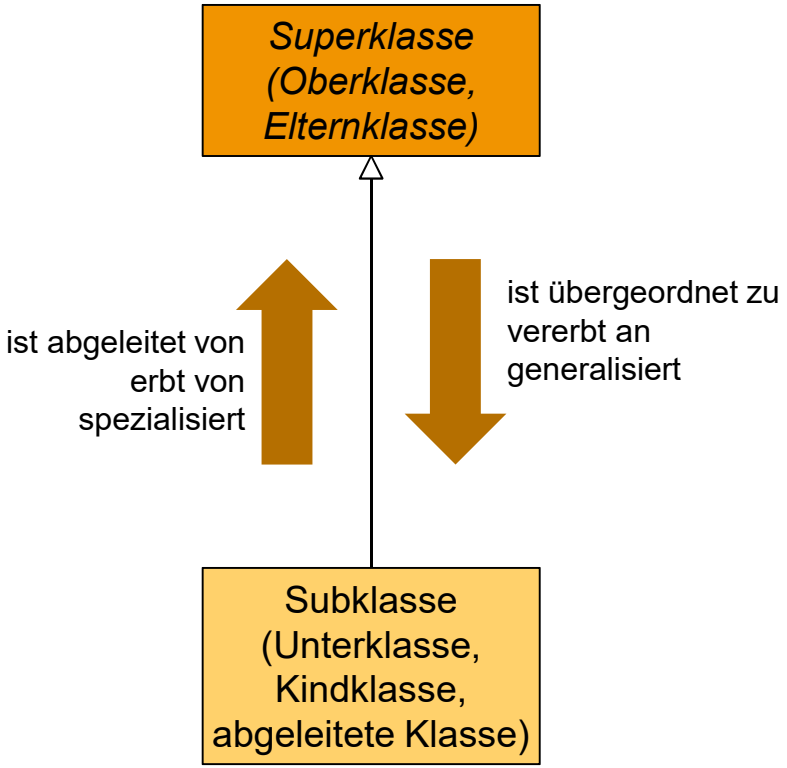
Komposition

- Starke Form der Aggregation
  (auch eine „ist Teil von“-Beziehung)
  - Teile sind vom Ganzen existenzabhängig
    (wird das Ganze kopiert/gelöscht, dann werden alle seine Teile kopiert/gelöscht)
  - Jedes Teil kann nur einem Kompositionsobjekt angehören
  - Beispiel



Vererbung

- Grundidee
  - Beschreibt Ähnlichkeit zwischen Klassen
  - Spezielle Beziehung: „is-a“; keine Multiplizität
  - Definiert gemeinsame Eigenschaften / Verhalten zentral
  - Spezialisierte Klassen basieren darauf, beschreiben nur Unterschiede
- Unterklasse erbt von Oberklasse
  - Operationen (Verhalten)
  - Attribute (möglichen Zustände)
  - Semantik (Substitutionsprinzip)
    - Statt Objekt der Oberklasse kann immer auch ein Objekt der Unterklasse verwendet werden

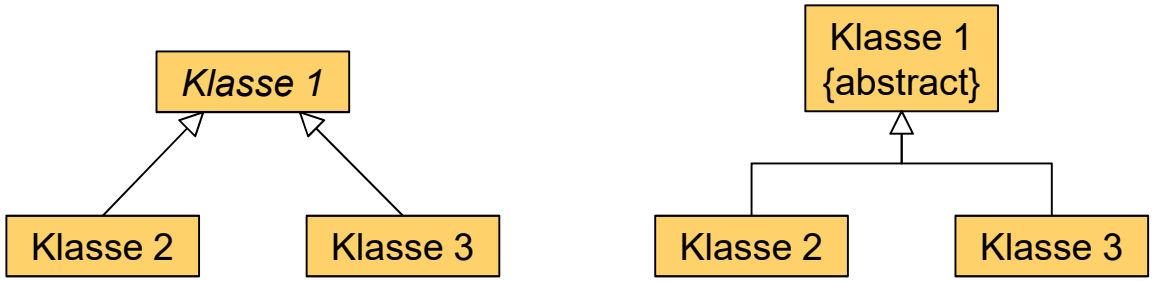


Abstrakte Klasse (1)

- Definition
  - Klasse, die nicht instanziiert werden kann
- Eigenschaften
  - Mindestens eine abstrakte Operation
    - Definiert nur Methodensignatur ohne Implementierung
    - Spezifiziert lediglich die Schnittstelle
  - Abgeleitete Klassen müssen alle abstrakten Operationen der Oberklasse implementieren
  - Zusätzlich in abstrakter Klasse auch konkrete Methoden möglich

Abstrakte Klasse (2)

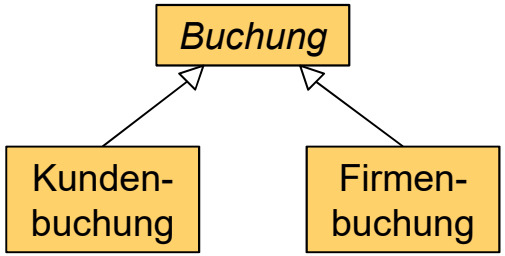
- Notation in UML
  - Schlüsselwort `{abstract}`
  - Oft *kursive Schrift*
  - Bei Handschrift: besser Schlüsselwort verwenden

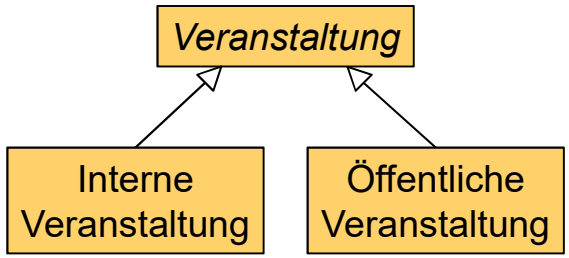


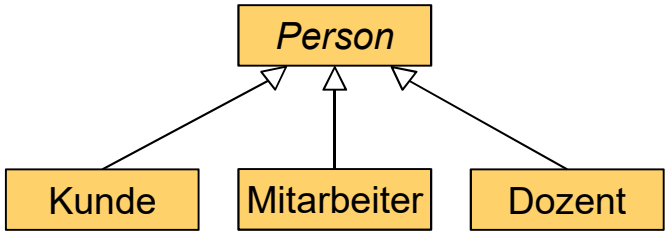
Quelle: Balzert, H. (2009): Lehrbuch der Softwaretechnik, S. 151

Vererbungsstrukturen

- Beispiele







Quelle: Balzert, H. (2009): Lehrbuch der Softwaretechnik, S. 152

Wie findet man Klassen? (1)

- Anforderungsspezifikation analysieren
  - Grammatische Untersuchung
  - Konzentration auf Substantive und Verben

- Beispiel
  - Aufgabenstellung:
    Zu entwickeln ist ein einfacher Grafischer Editor
  - Problemstellung (Vorstudie / Grobkonzept):
    Mit Hilfe eines grafischen Editors können Zeichnungen angefertigt werden. Es werden verschiedene Grafiksymbole wie Linie, Rechteck, Kreis usw. zur Verfügung gestellt, die selektiert und in die Zeichnung kopiert werden können.
    Der Editor erlaubt das Abspeichern und Laden erstellter Zeichnungen.

Wie findet man Klassen? (2)

- Substantive = Klassen
  - Aufgabenstellung:
    Zu entwickeln ist ein einfacher Grafischer Editor
  - Problemstellung (Vorstudie / Grobkonzept):
    Mit Hilfe eines grafischen Editors können Zeichnungen angefertigt werden.
    Es werden verschiedene Grafiksymbole wie Linie, Rechteck, Kreis usw.
    zur Verfügung gestellt, die selektiert und in die Zeichnung kopiert werden können.
    Der Editor erlaubt das Abspeichern und Laden erstellter Zeichnungen.

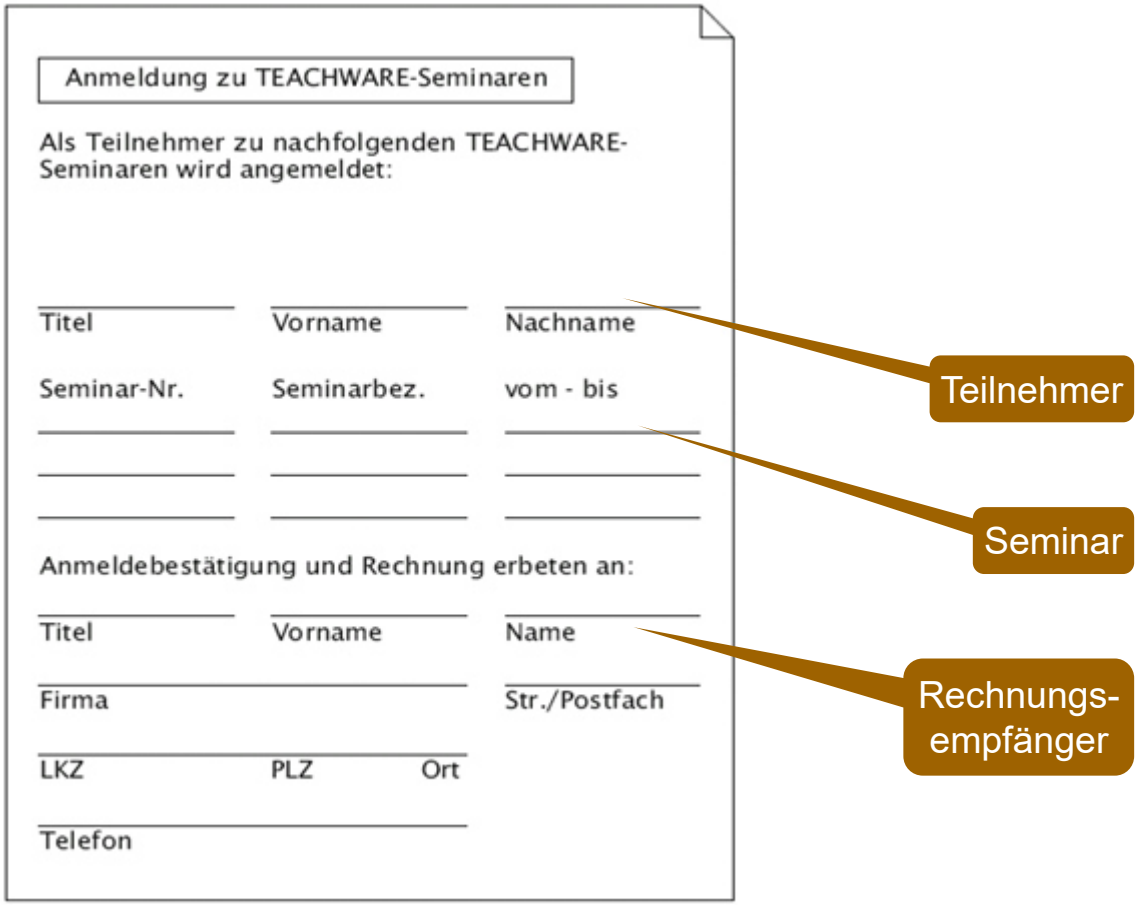
Wie findet man Klassen? (3)

- Verben = Methoden / Funktionen

  - Aufgabenstellung:
    Zu entwickeln ist ein einfacher Grafischer Editor
  - Problemstellung (Vorstudie / Grobkonzept):
    Mit Hilfe eines grafischen Editors können Zeichnungen angefertigt werden.
    Es werden verschiedene Grafiksymbole wie Linie, Rechteck, Kreis usw.
    zur Verfügung gestellt, die selektiert und in die Zeichnung kopiert werden können.
    Der Editor erlaubt das Abspeichern und Laden erstellter Zeichnungen.

Wie findet man Klassen? (4)

- Dokumentenanalyse
  - Formulare und Listen
  - Software-Altsysteme: z.B. Benutzerhandbücher, Bildschirmmasken
  - Technische Systeme: reale Objekte



Quelle: Balzert, H. (2009): Lehrbuch der Softwaretechnik, S. 141

Wie findet man Klassen? (5)

- CRC-Karten
  - CRC = **C**lass-**R**esponsibilities-**C**ollaboration
    (Klasse- Verantwortlichkeiten- Beteiligte)
  - CRC-Karten = Karteikarten (an einer Pinnwand)
  - Idee: Analysten / Entwickler füllen Karten aus und diskutieren
  - Sind als Ergänzung zum OOA-Modell zu verstehen
  - Informationen werden auf einer höheren Abstraktionsebene dargestellt als im Klassendiagramm

Wie findet man Klassen? (6)

- Beispiel für eine CRC-Karte

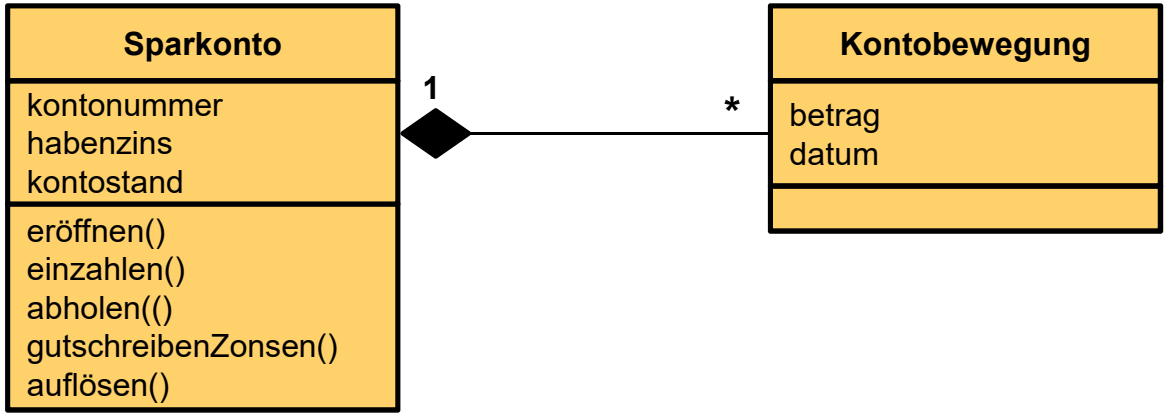
Klasse: ***Konto***	
Basisklasse: **-**	
Verantwortlichkeiten:	Kollaboration:
einzahlen*:* Es wird ein Betrag auf das Konto eingezahlt. Der entstehende Umsatz wird in die Liste der Umsätze des Kontos aufgenommen. **auszahlen*:*** Es wird ein Betrag abgehoben. Analog zum Einzahlen wird ein Umsatz erzeugt. **erstelleKontoauszug*:*** Erstellt einen Kontoauszug mit allen Buchungen seit dem letzten Kontoauszug	***Kontoauszug*** ***Umsatz***

Wie findet man Klassen? (7)

- Vergleich CRC-Karte und Klassendiagramm

Klasse: *Sparkonto*	
Verantwortlichkeiten:	***Kollaboration:***
• *Verwaltet ein Sparkonto* • *Delegiert Aufgaben an Kontobewegung*	• *Kontobewegung*

- Unterschiedliche Abstraktionsebene



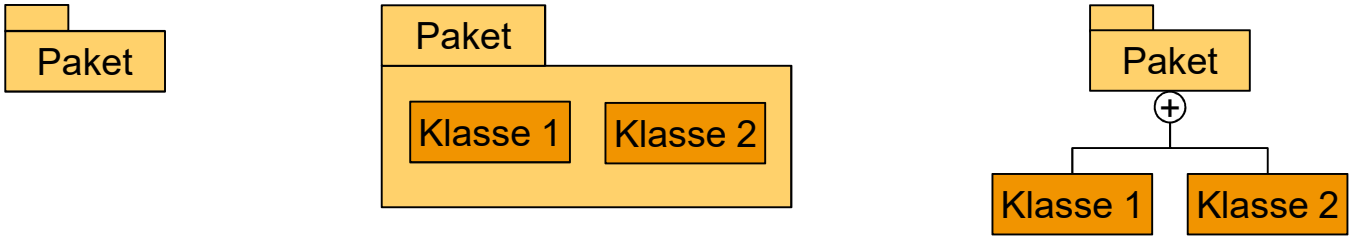
Quelle: Balzert, H. (2009): Lehrbuch der Softwaretechnik, S. 175

Paket (1)

- Bei umfangreichen Softwareentwicklungen entstehen viele Klassen und Diagramme

- Paket
  - Strukturierungsmechanismus, das von den Details abstrahiert und die übergeordnete Struktur verdeutlicht
  - UML: gruppiert Modellelemente (z.B. Klassen) und Diagramme

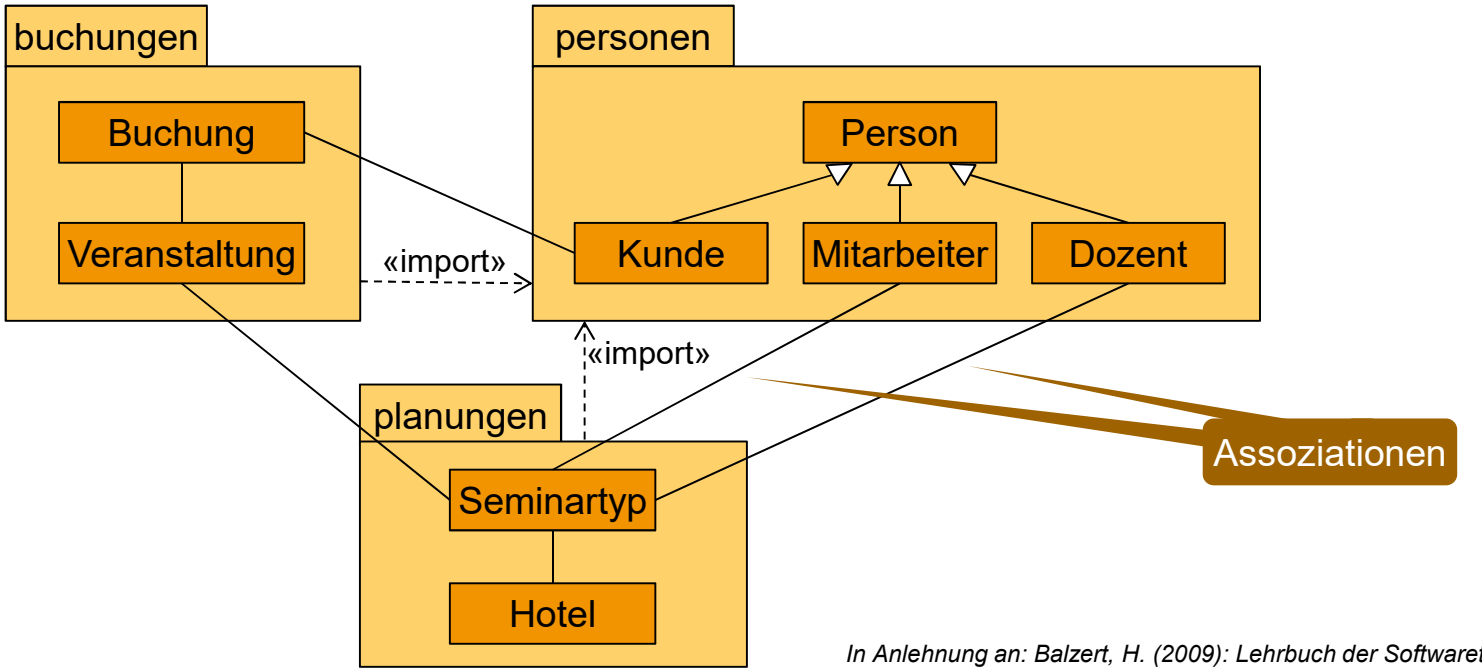
  - Notationen



Quelle: Balzert, H. (2009): Lehrbuch der Softwaretechnik, S. 145

Paket (2)

- Beispiel Paketdiagramm (SemOrg)



In Anlehnung an: Balzert, H. (2009): Lehrbuch der Softwaretechnik, S. 147

- Paketdiagramme dienen auch dazu, verschiedene Architekturschichten klar zu trennen, z.B.
  - Benutzungsoberfläche
  - Fachkonzept
  - Datenbankzugriffe