5. UML – Übersicht

Literaturempfehlung für UML

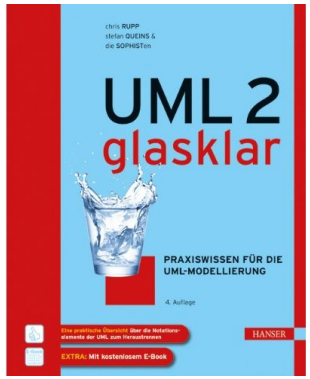
C. Rupp, S.Queins, die SOPHISTen: UML 2 glasklar: Praxiswissen für die UML-Modellierung, Hanser Verlag 2012
als E-Book an HTWG-Bibliothek verfügbar

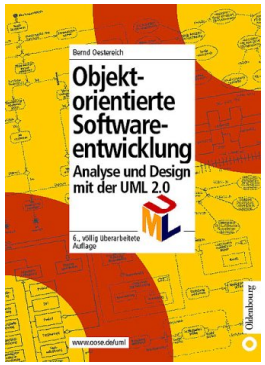
H. Balzert: Lehrbuch der Objektmodellierung – Analyse und Entwurf mit der UML 2, Spektrum Akademischer Verlag, 2005

B. Oesterreich: Objektorientierte Softwareentwicklung – Analyse und Design mit der UML 2.0, Oldenbourg Verlag, 2004

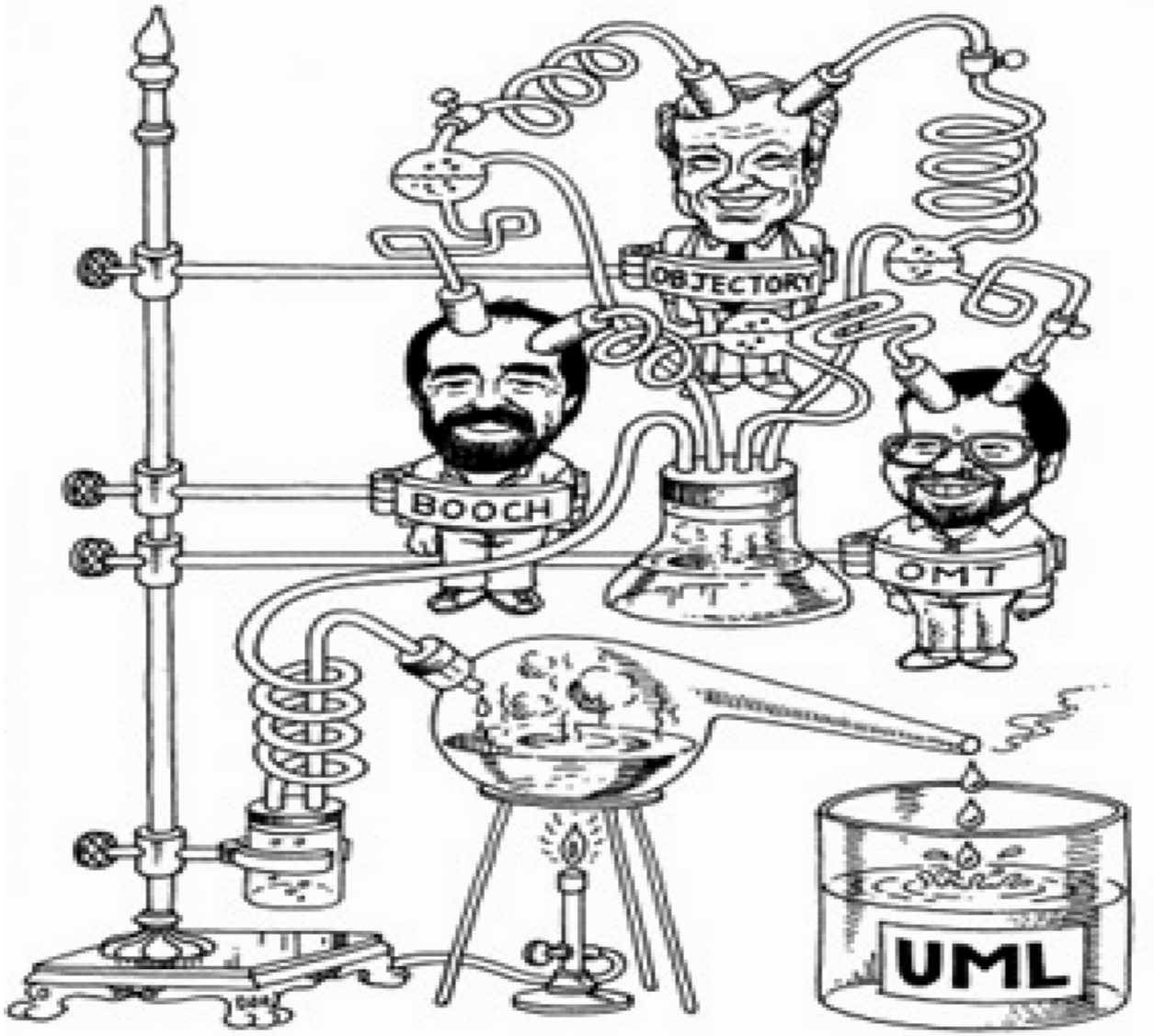
C. Kecher: UML2 – Das umfassende Handbuch, Galileo Computing, 2009

www.uml.org

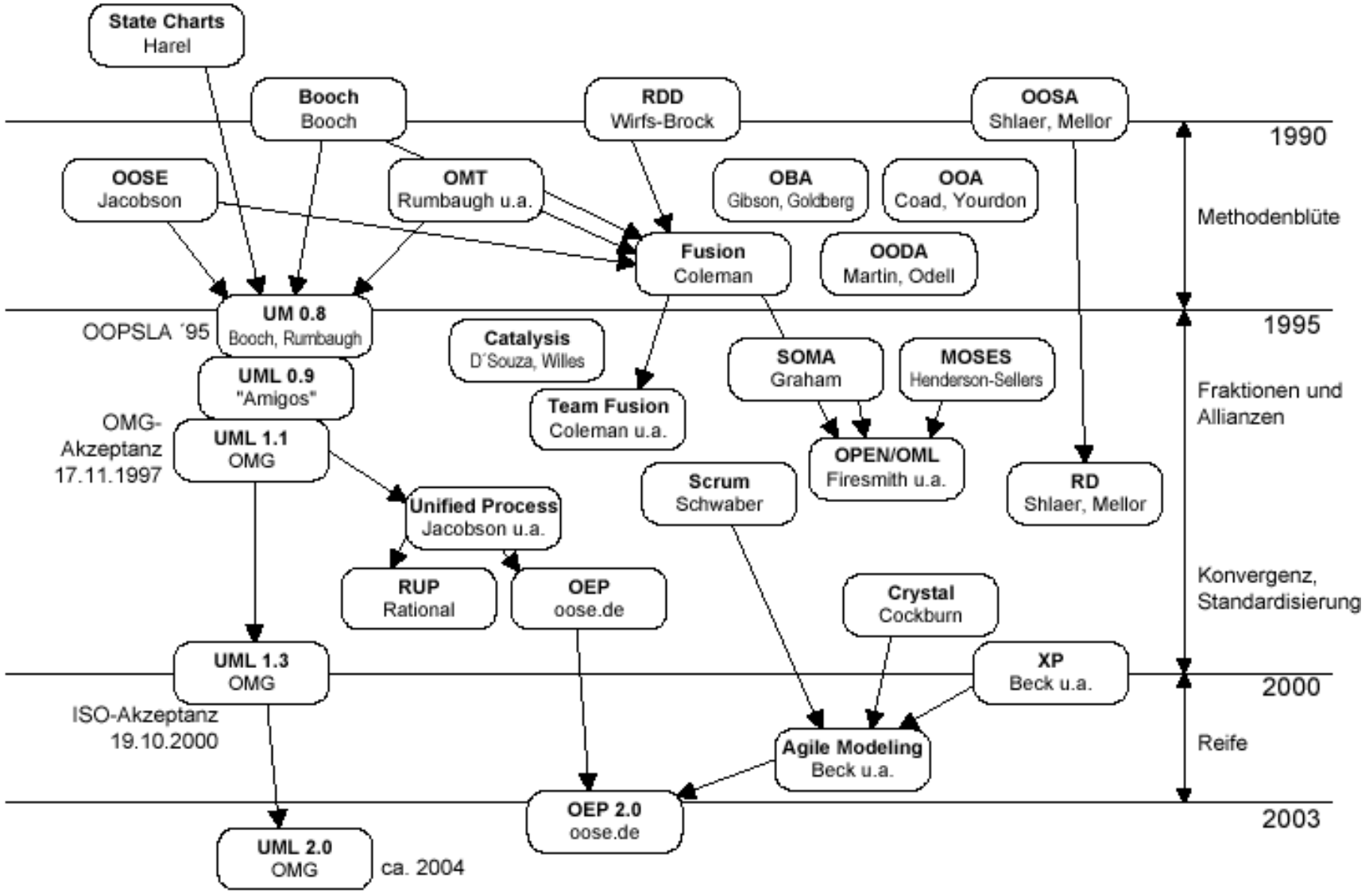




5.1 Einführung UML



Historische Entwicklung objektorientierter Methoden und der UML

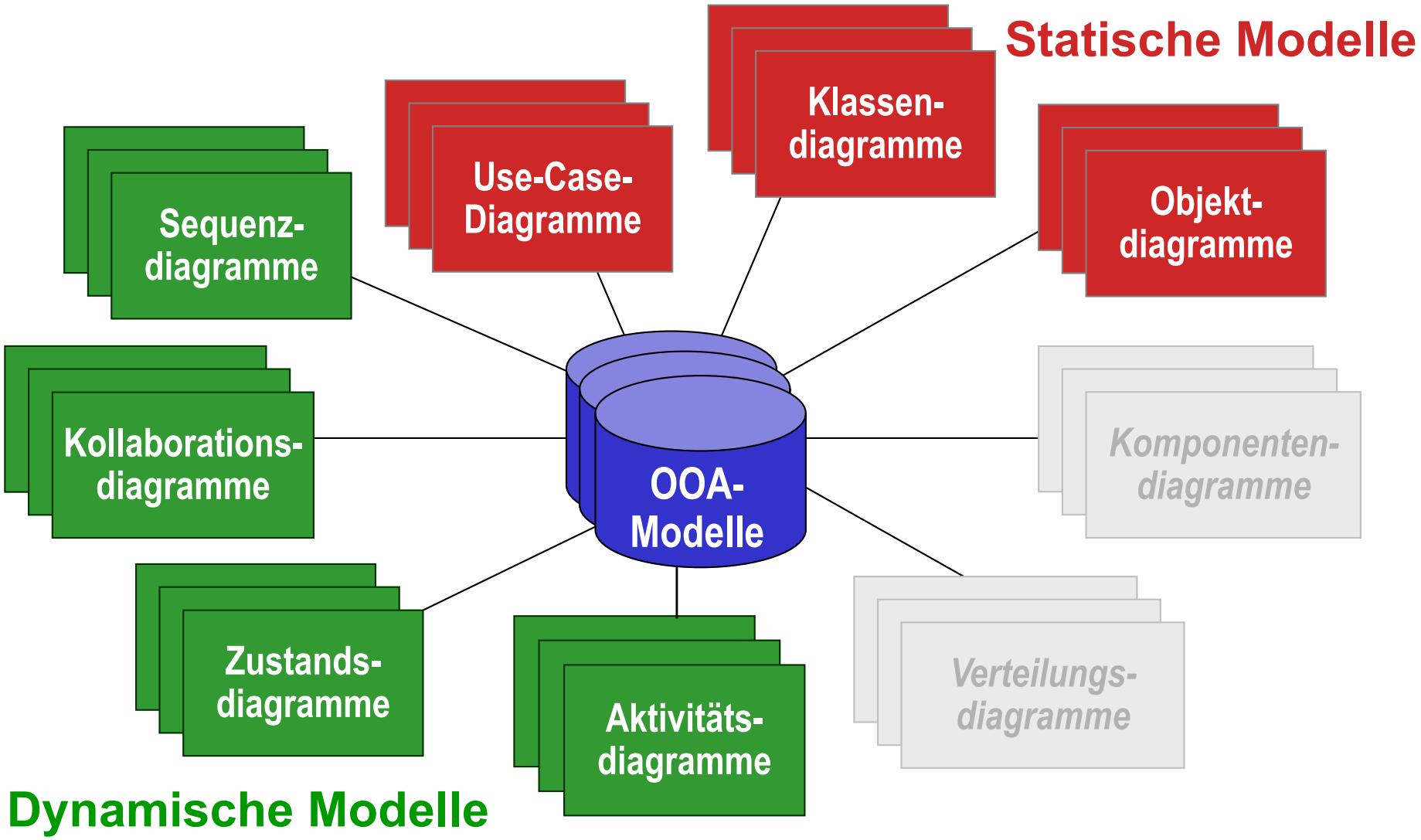


Quelle: www.oose.de

Unified Modelling Language
Historische Entwicklung

- Bis 1994: Viele konkurrierende, uneinheitliche Notationen für die Objektmodellierung
- 1994: Booch und Rumbaugh von Rational beginnen mit der Standardisierung von OO-Notationen
- 1995: Jacobson stößt dazu (UML 1.0)
- 1997: UML wird von der OMG™ als Standard akzeptiert (OMG™ = Object Management Group™)
- 2004: Grundsätzliche Überarbeitung: UML 2.0
- 2007: UML 2.1
- Aktuell: UML 2.4.1 – enthält u.a. Definition von UML-Metamodell

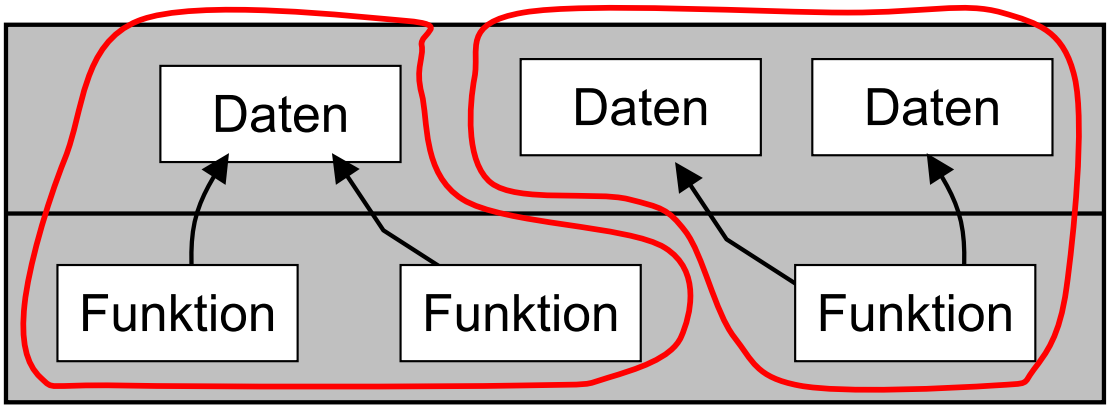
UML – Diagrammtypen



Grundkonzept der Objektorientierung

Objektorientierung = Denken in Objekten

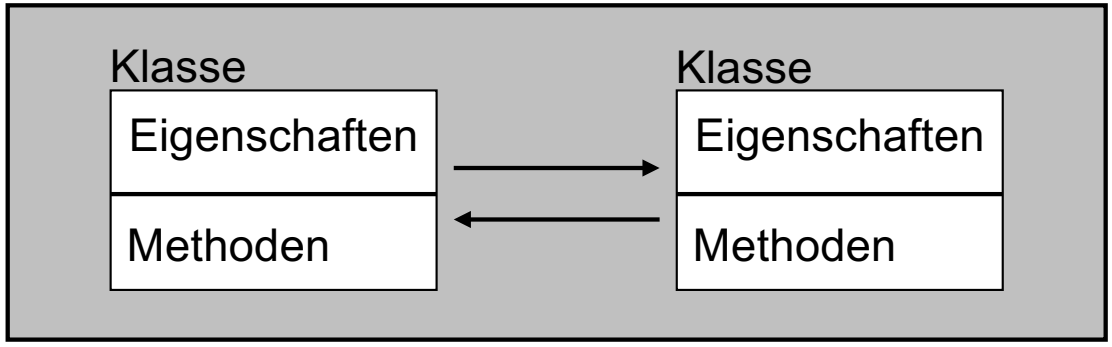
Klassisches prozedurales Konzept



Daten

Funktionen

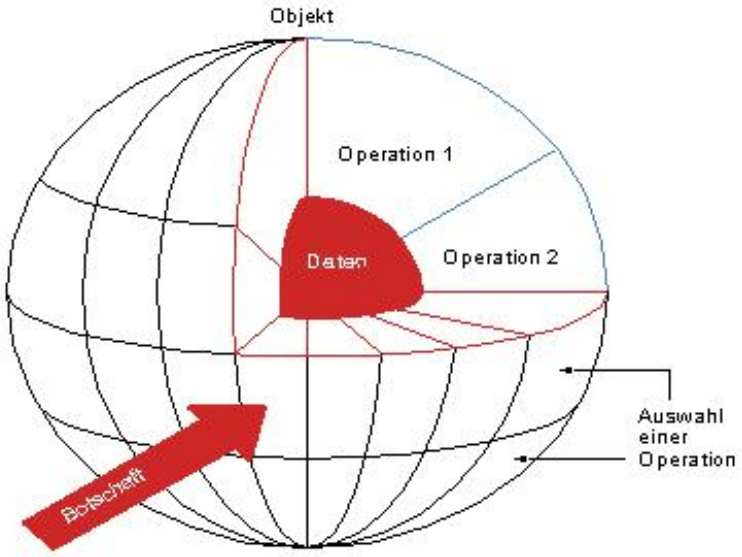
Objektorientiertes Konzept



Objekte

Kapselungsprinzip

- Kapselungsprinzip (strikte Kapselung)
  - Klassen fassen Attribute und Operationen zu einer Einheit zusammen
  - Attribute sind nur indirekt über die Operationen der Klasse zugänglich (öffentliche Schnittstelle)
- Häufig: Explizite Kapselung
  - Deklaration von Attributen oder Methoden als öffentlich bzw. privat

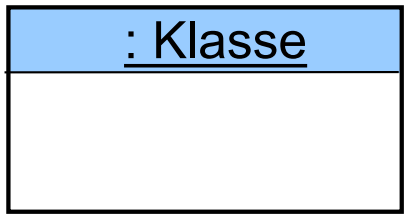


UML – Übersicht

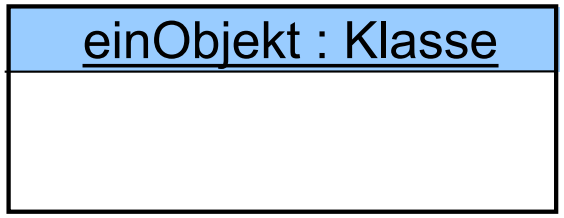
5 UML – Strukturdiagramme
- 5.1 Einführung UML
- 5.2 Objektdiagramm, Klassendiagramm
- 5.3 Komponentendiagramm

6 UML – Verhaltensmodellierung
- 6.1 Use Case-Diagramm
- 6.2 Aktivitätsdiagramm
- 6.3 Kommunikationsdiagramm
- 6.4 Sequenzdiagramm
- 6.5 Zustandsdiagramm

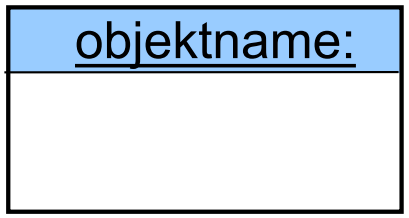
5.2 Objektdiagramm



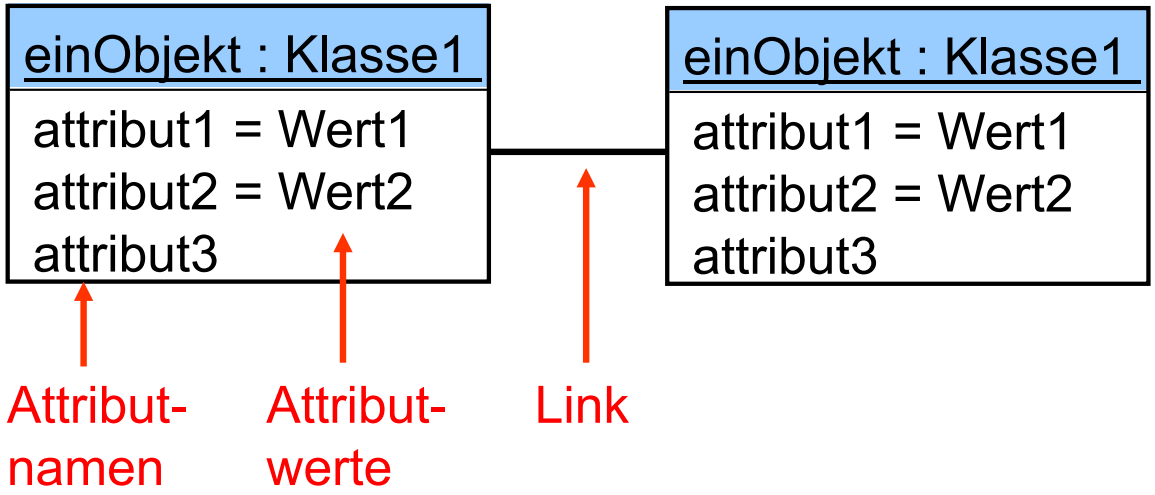
Namenloses Objekt
der Klasse "Klasse"



Objekt "einObjekt"
der Klasse "Klasse"

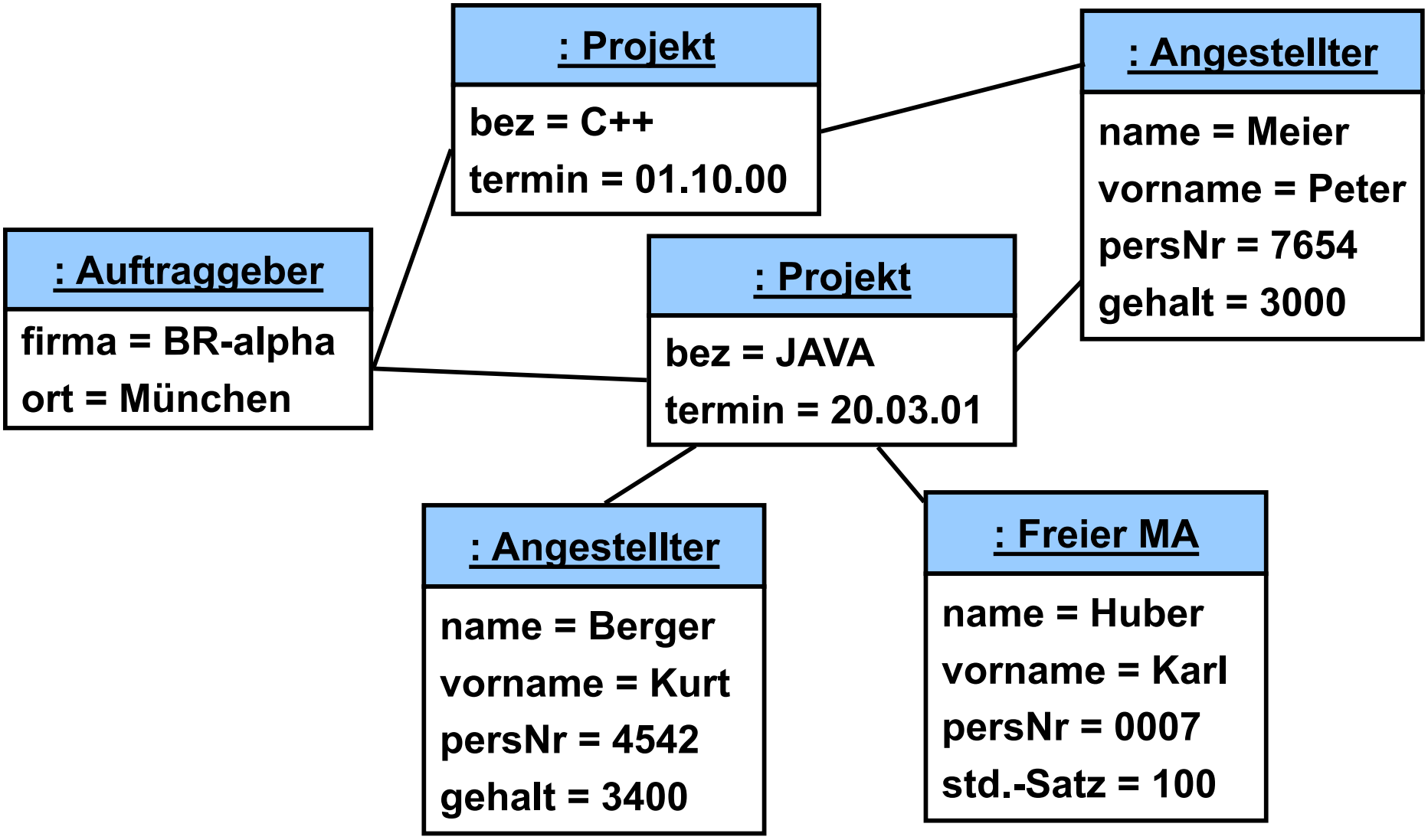


Objekt “objektname"
(Klasse ergibt sich
aus dem Kontext)



Zur Unterscheidung von der Klassen-Notation wird Objektname unter-strichen und beginnt mit einen Kleinbuchstaben

Objektdiagramm
Beispiel



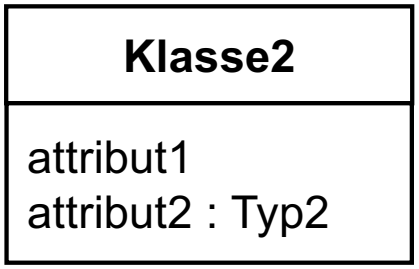
Klassendiagramm

- **Das** zentrale UML-Diagramm: Darstellung einer Menge inhaltlich zusammenhängender Klassen
  - Klassen mit Namen
  - Attribute ggfs. mit Datentypen
  - Operationen ggfs. mit Signaturen
  - Assoziationen zwischen Klassen (ggfs. als Aggregation oder Komposition)
  - Namen und Multiplizitäten (Kardinalitäten) der Assoziationen
  - Rollen von Klassen in Assoziationen
  - …

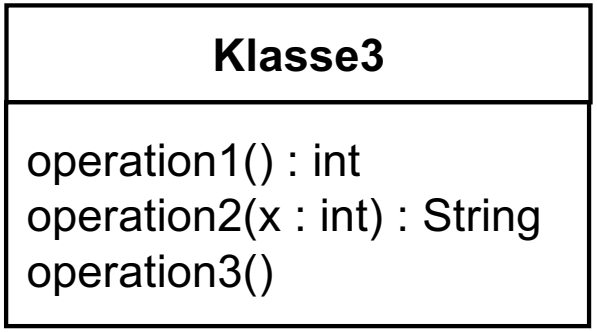
UML-Notation: Klassen

Minimaldarstellung

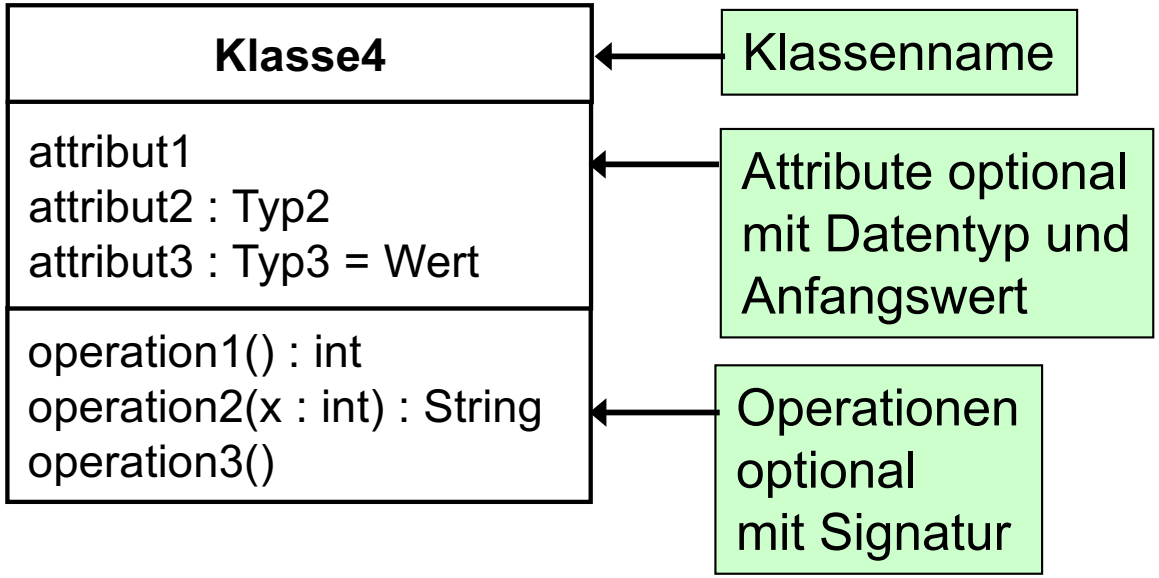
Nur Attribute



Nur Operationen

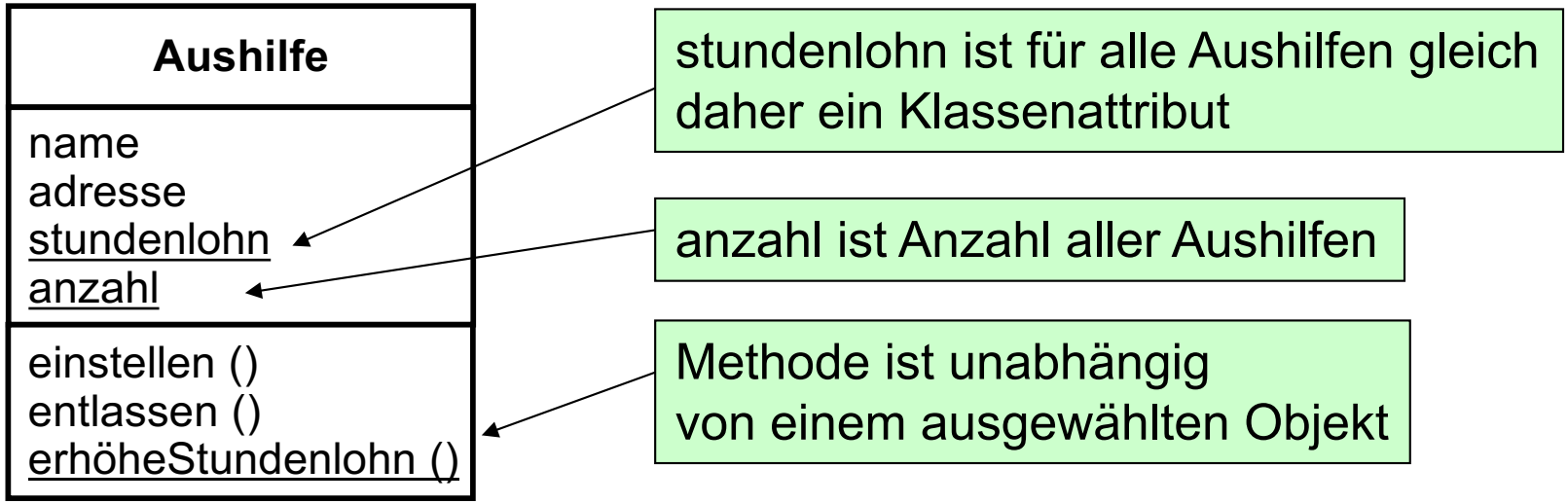


Vollständige Darstellung



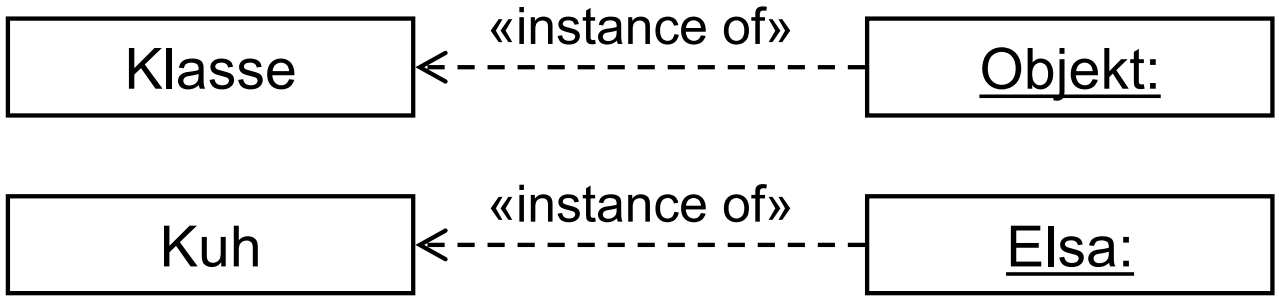
Klassenattribute – Objektattribute

- Klassenattribute
  - Klassenattribute gehören nicht einem einzelnen Objekt, sondern sind Attribut einer Klasse
  - Alle Objekte dieser Klasse können auf ein solches gemeinsames Klassenattribut zugreifen
- Klassenoperationen
  - Operation, die der jeweiligen Klasse zugeordnet ist



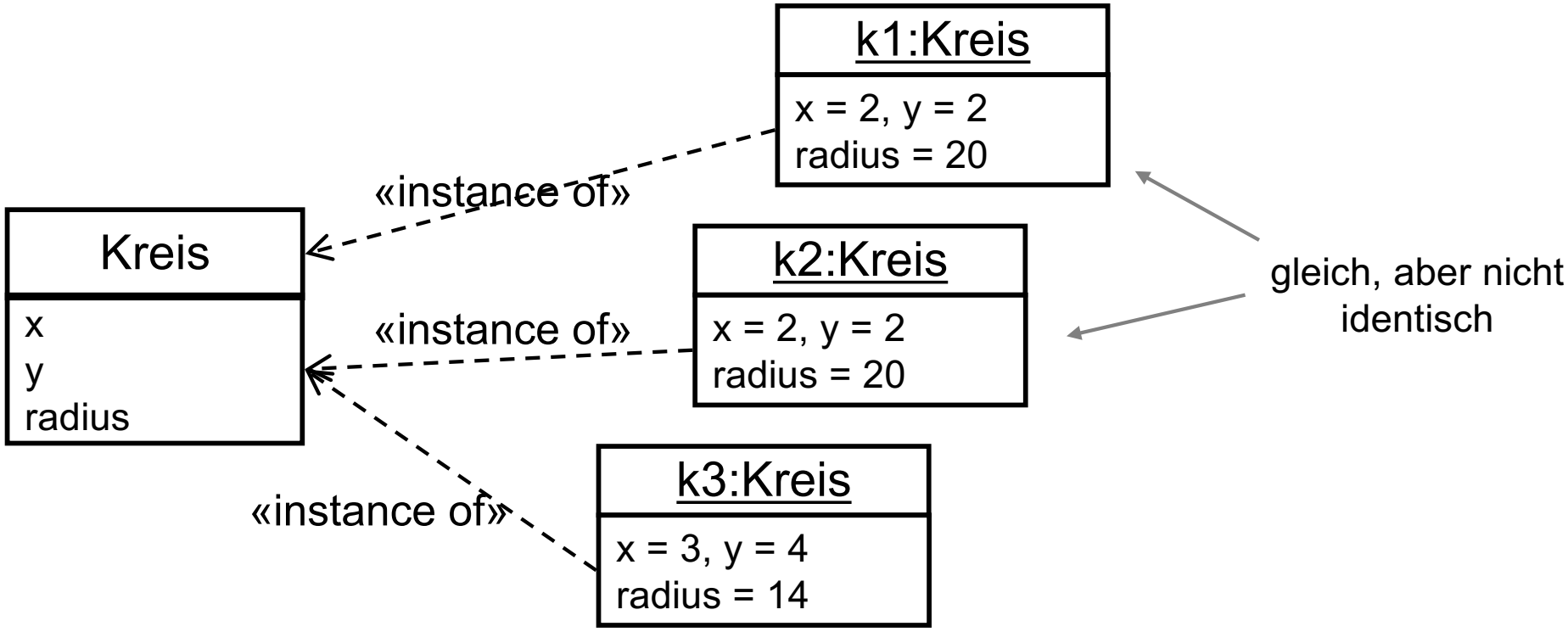
Objekt-Klassen-Beziehung

- Objekt-Klassen-Prinzip
  - Klassen beschreiben Struktur und Verhalten einer Menge gleichartiger Objekte.
  - Objekt ist eine Instanz einer Klasse, die sich entsprechend dem Protokoll ihrer Klasse verhält
  - Andere Bezeichnungen: Instanz, Instantiierung, Exemplar, Objekt



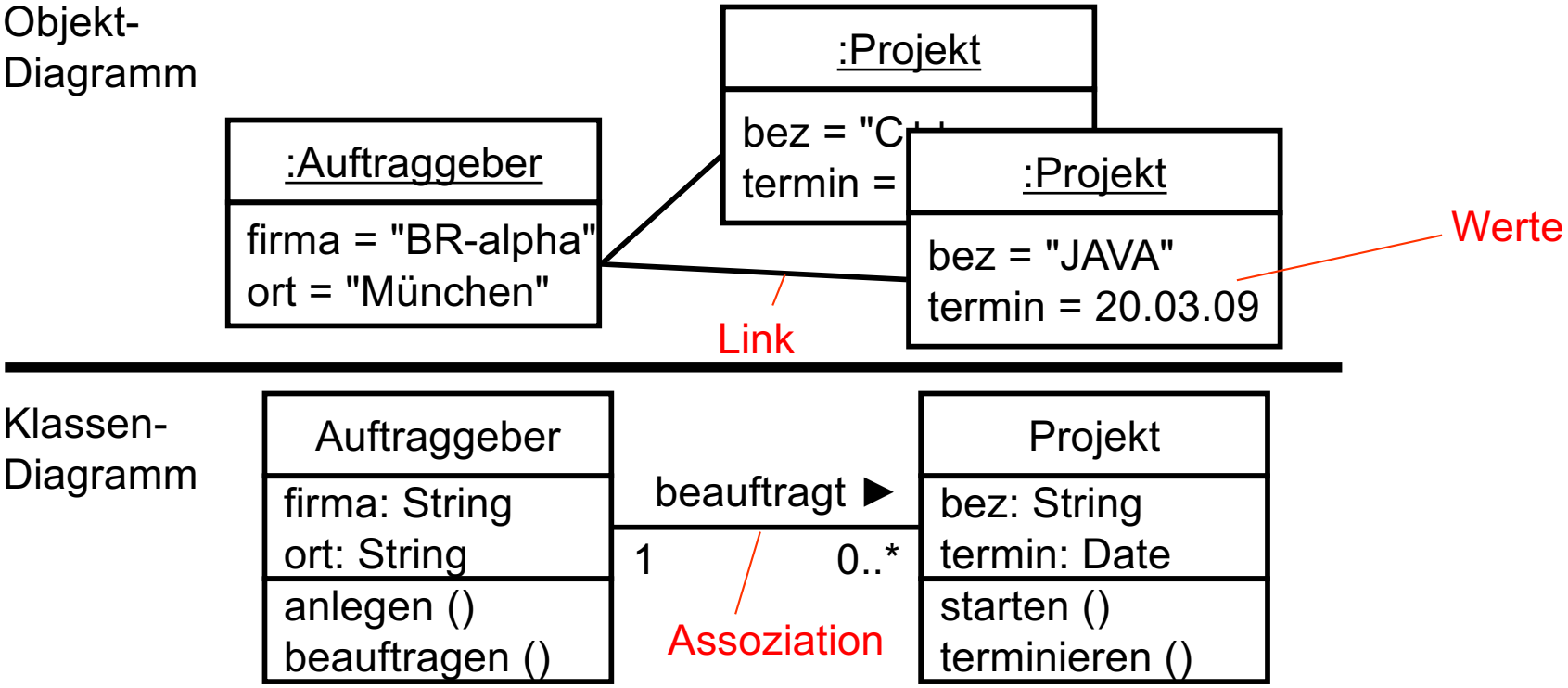
Objektidentität

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



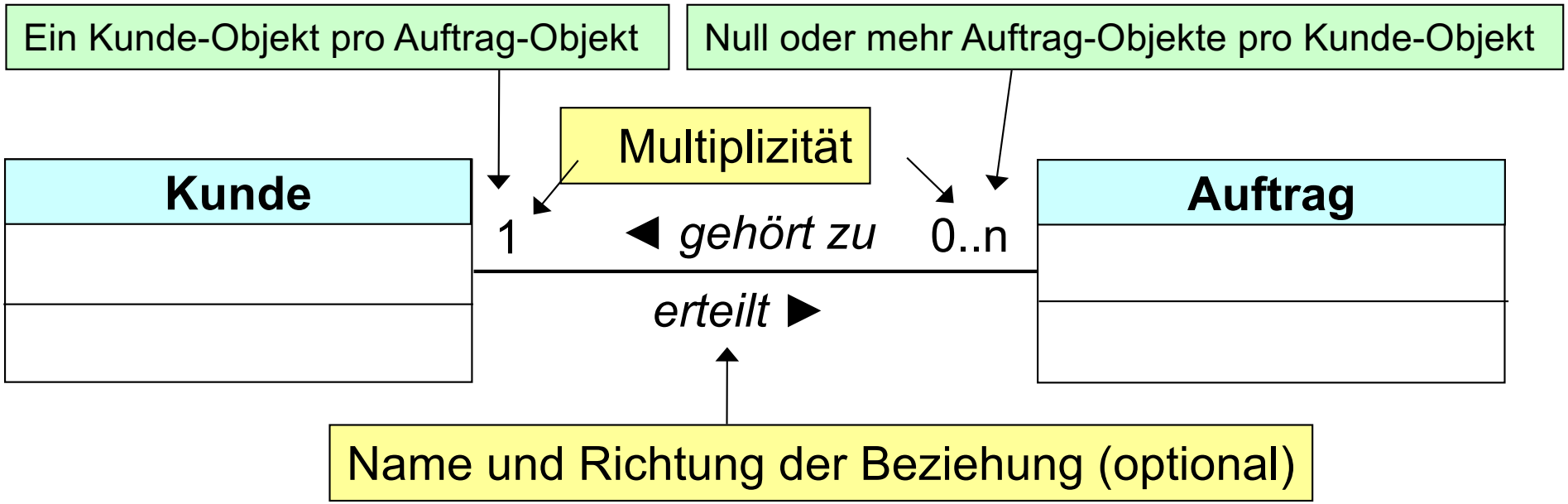
Instanzen in UML

- Drei Arten von Instanzen
  - Objekt als Instanz einer Klasse
  - Link als Instanz einer Assoziation
  - Wert als Instanz eines Attributs

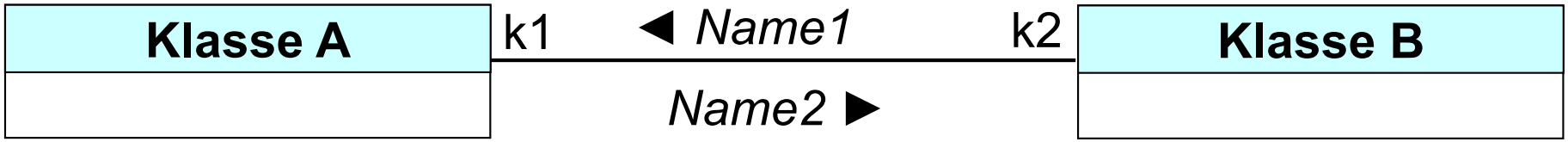


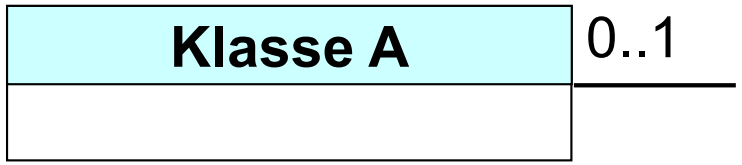
Assoziationen

- Assoziation
  - Quantifizierte Beziehung zwischen einer Anzahl von Objekten zweier Klassen
- In UML ausgedrückt durch beschriftete und gewichtete Linien zwischen den Klassen

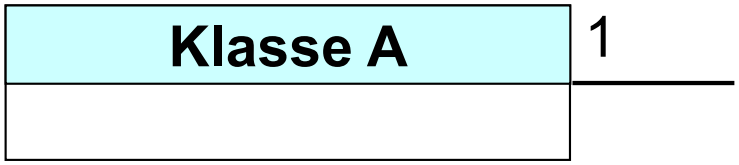


Multiplizitäten

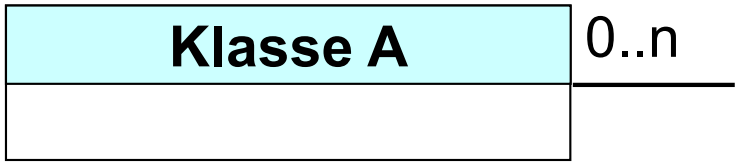




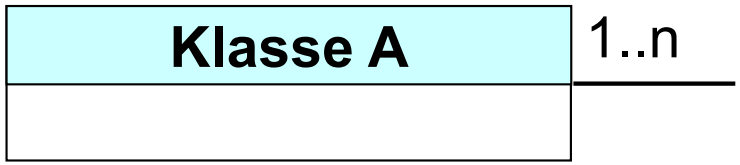
kann-Beziehung, einfach



muss-Beziehung, einfach

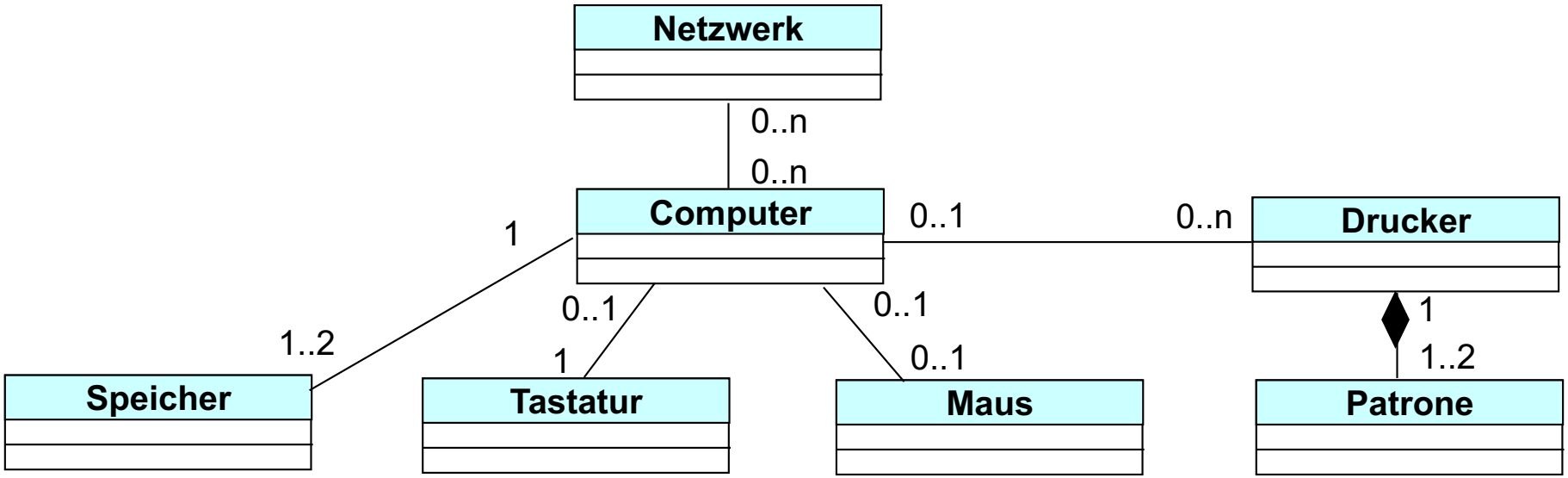


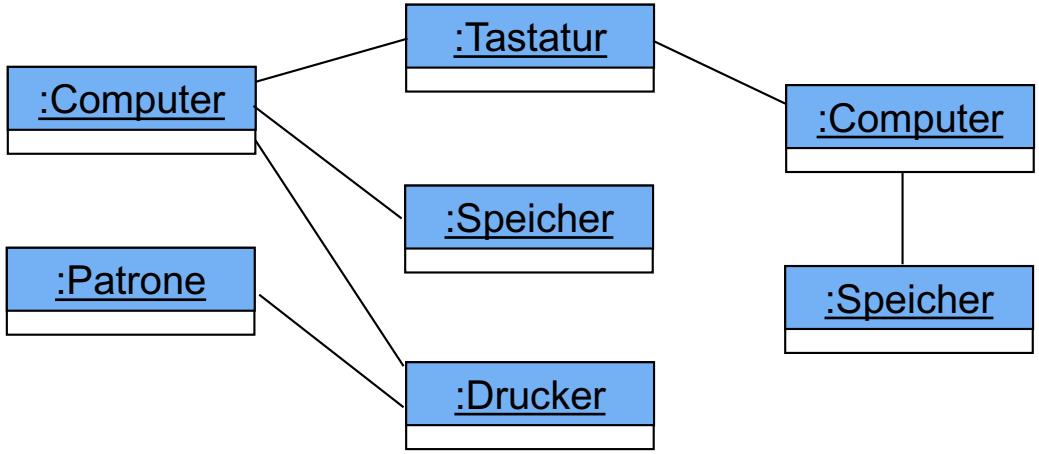
kann-Beziehung, mehrfach
(auch Konstante statt *n* möglich)



muss-Beziehung, mehrfach
(auch Konstante statt *n* möglich)

Klassen- vs. Objektdiagramme

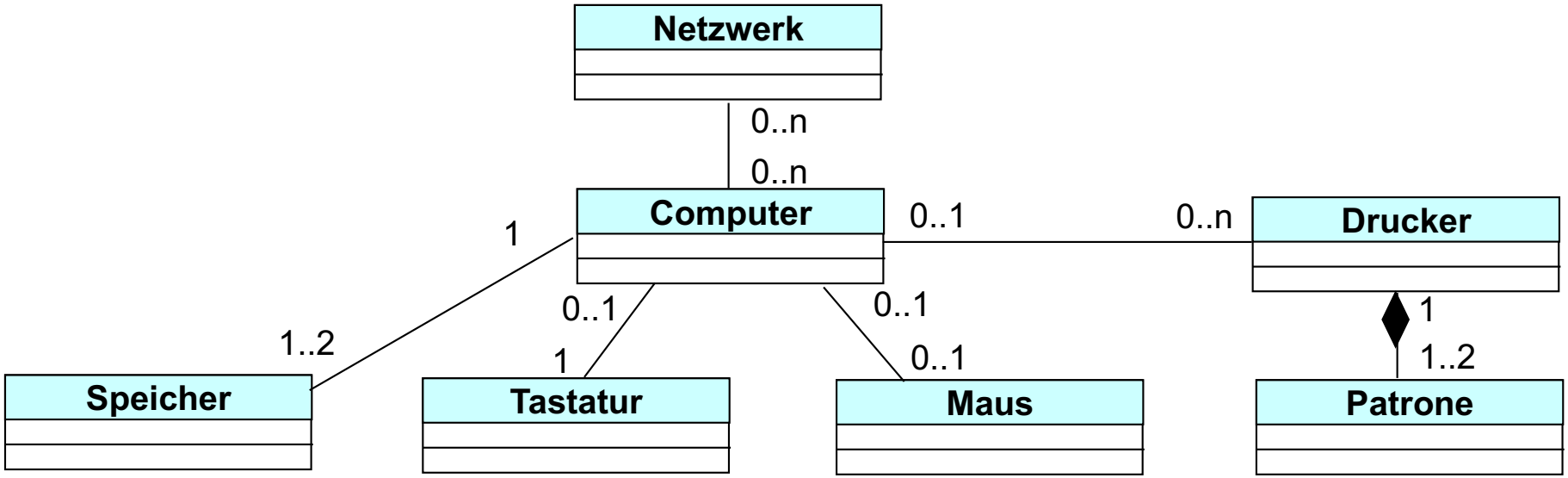


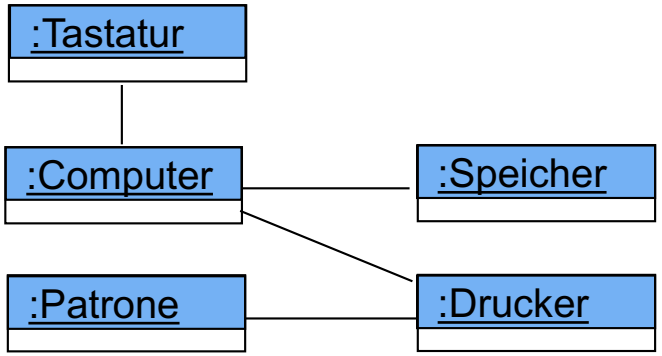


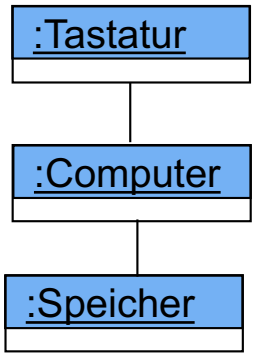
Korrekt? **NEIN!**

Eine Tastatur für nur einen Computer!

Klassen- vs. Objektdiagramme



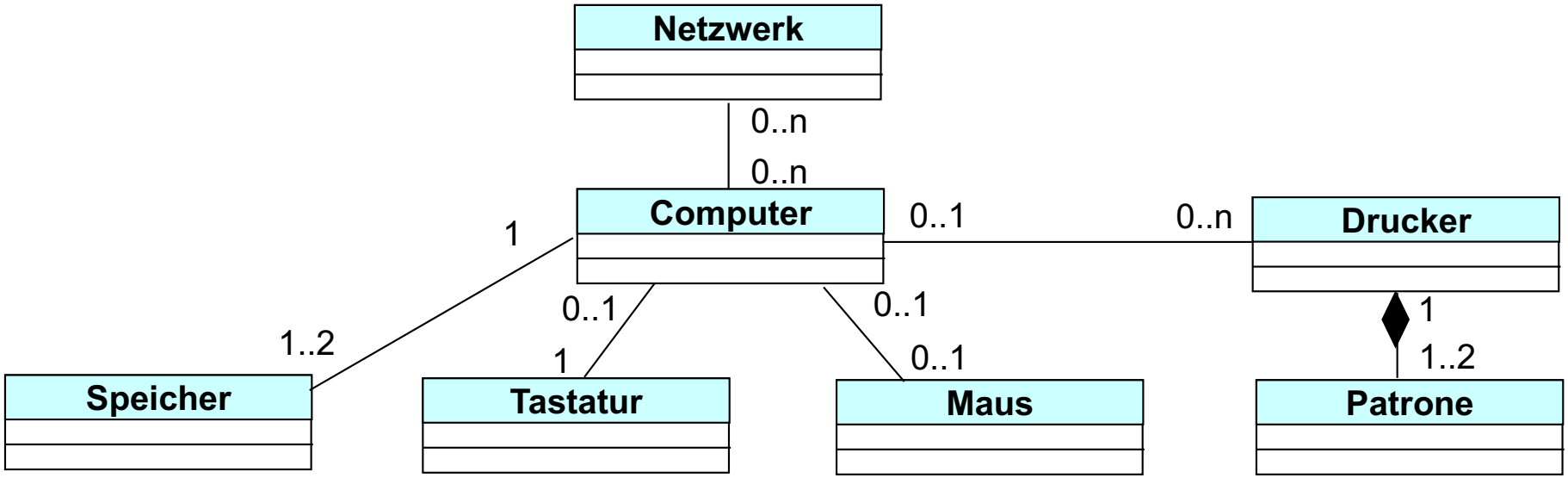


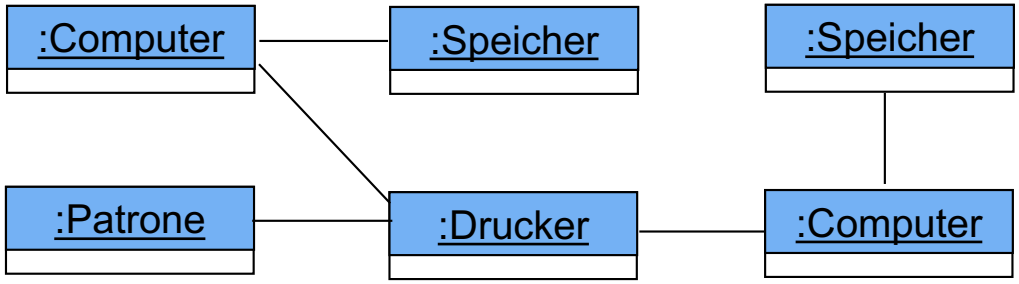


Korrekt? **ja**

Alle Verbindungen o.k.

Klassen- vs. Objektdiagramme

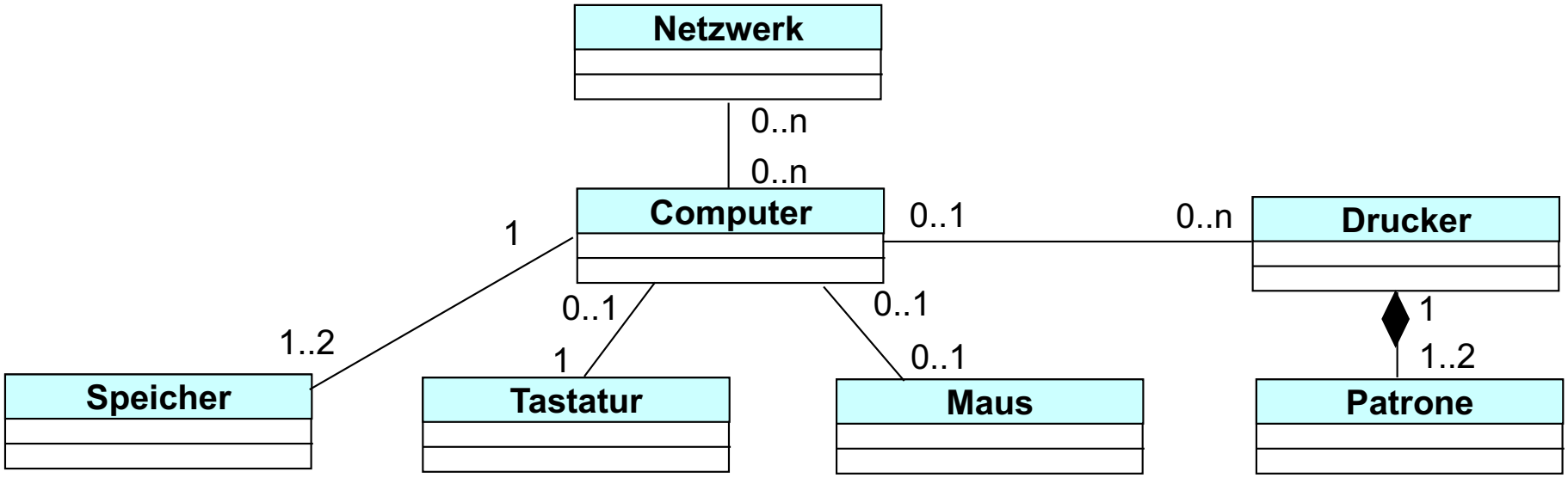


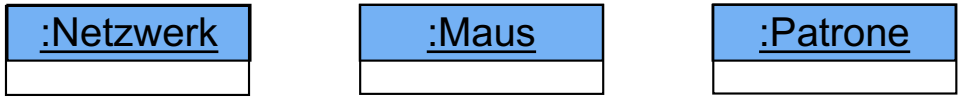


Korrekt? **NEIN!**

- Pro Computer ein Drucker
- Computer braucht Tastatur

Klassen- vs. Objektdiagramme

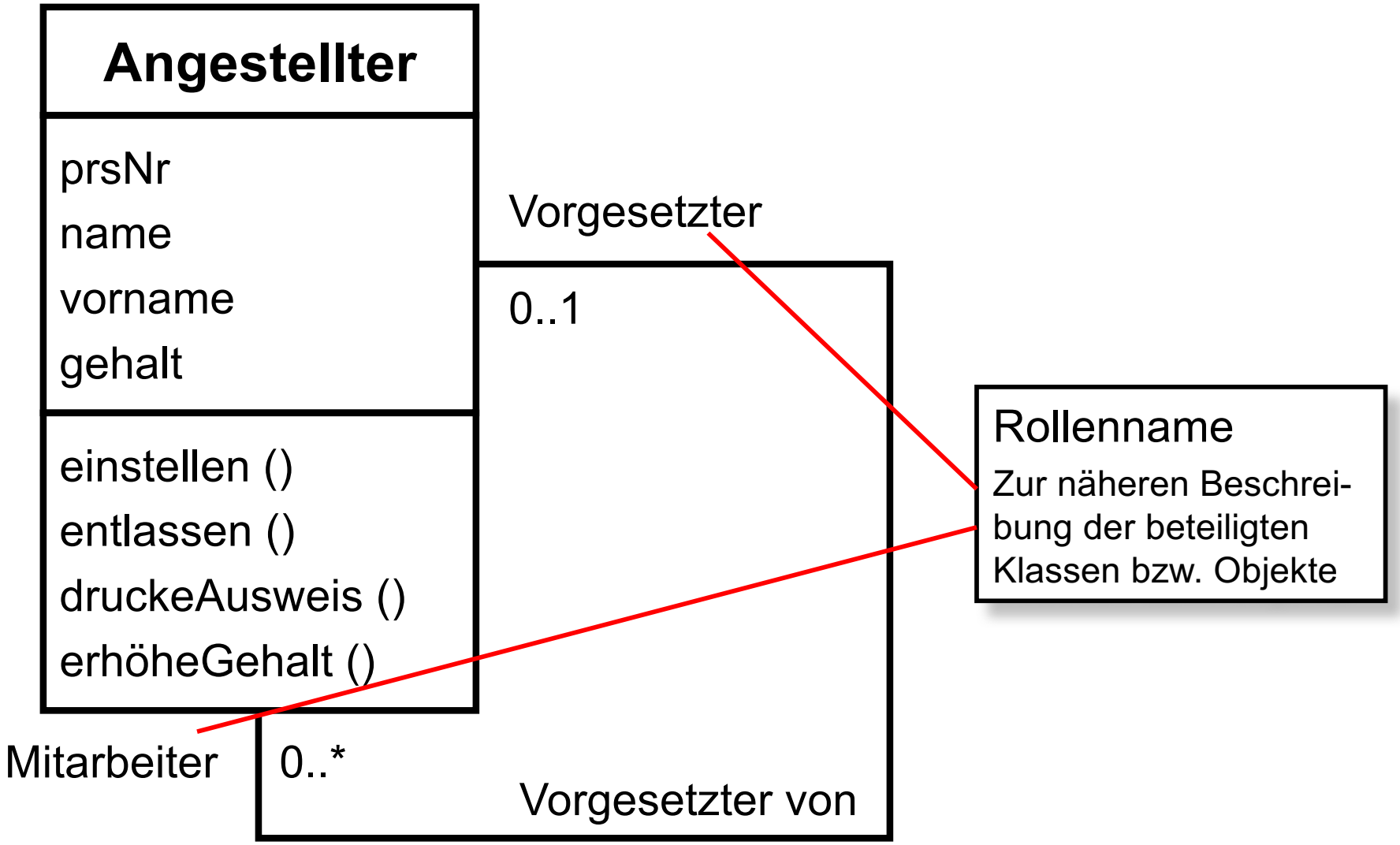




Korrekt? **NEIN!**

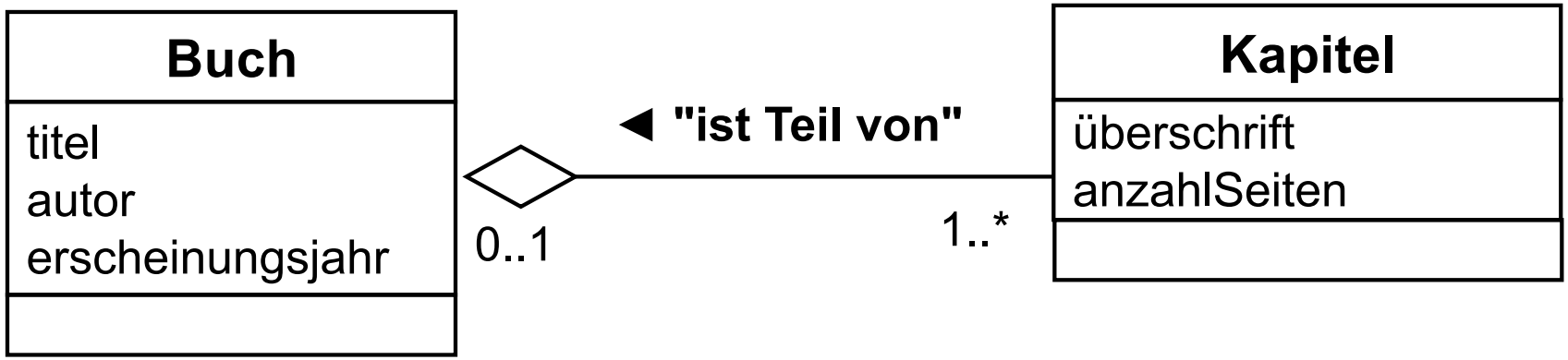
Patrone braucht Drucker

Reflexive Assoziation



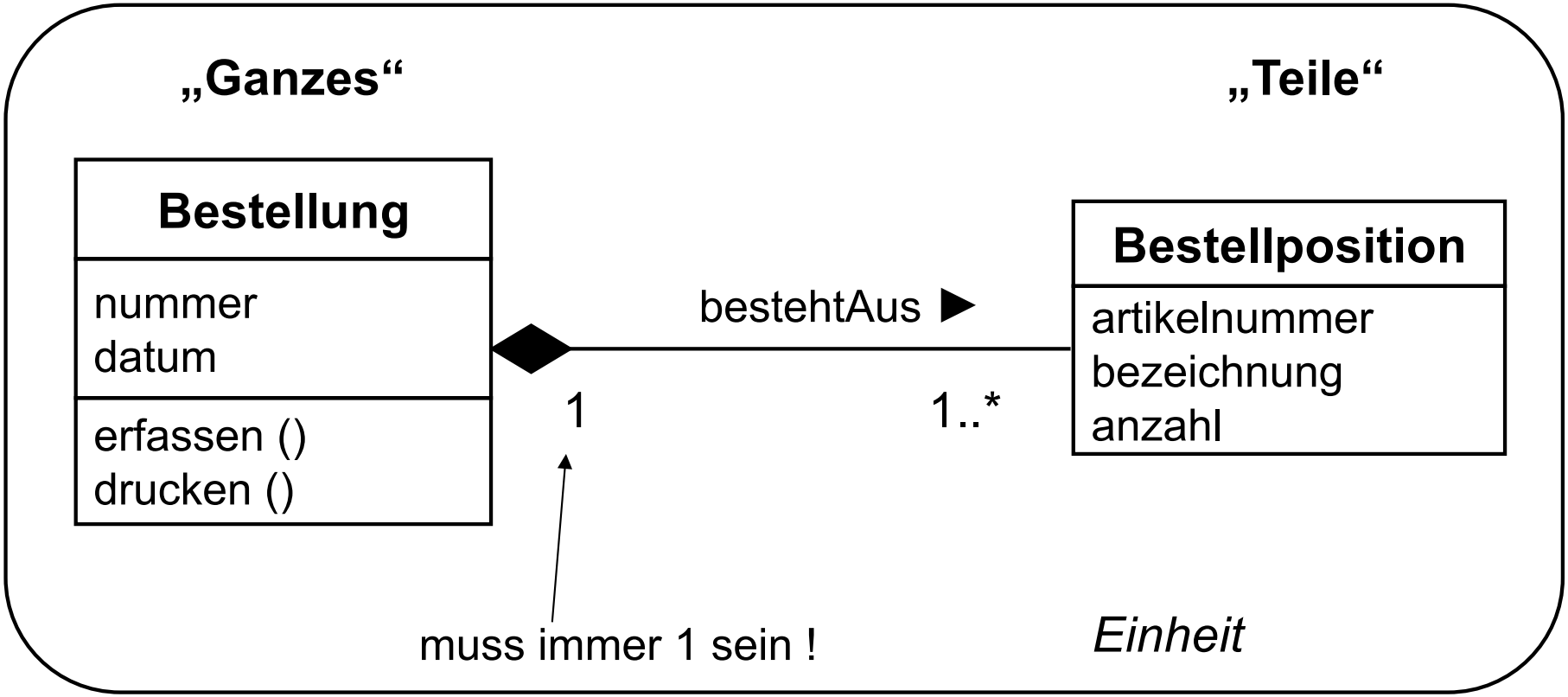
Aggregation

- Aggregation
  - Spezielle Form der Assoziation
  - Bedeutung: "besteht aus" oder "ist-Teil-von"-Beziehung



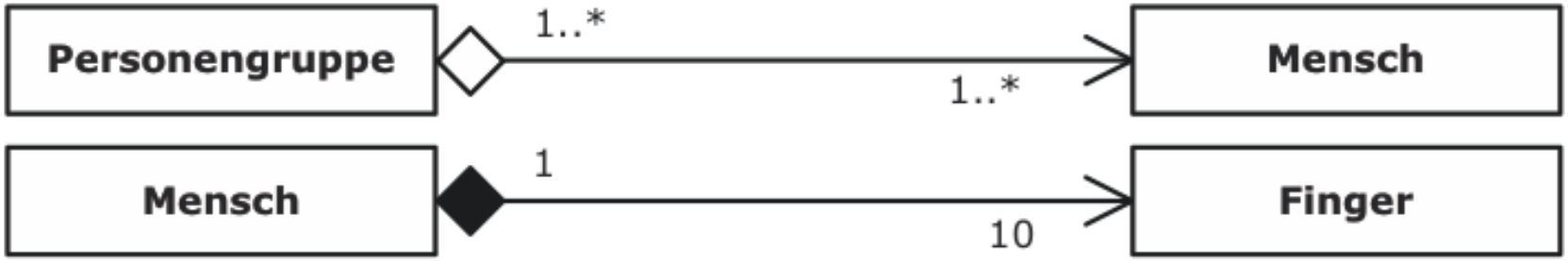
Komposition

- Komposition
  - Starke Form der Aggregation
  - Ganzes und Teile sind untrennbar verbunden (existenzabhängig)



Aggregation und Komposition

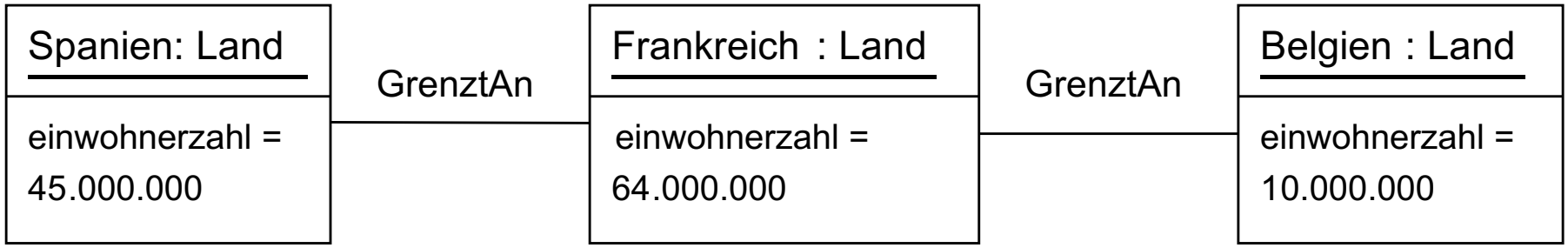
- Unterschiede Komposition zu Aggregation
  - Physische Inklusion der Teile im Ganzen
  - Teile und Ganzes bilden eine Einheit



Quelle: C. Rupp. S. Queins & die SOPHISTen: UML 2 glasklar

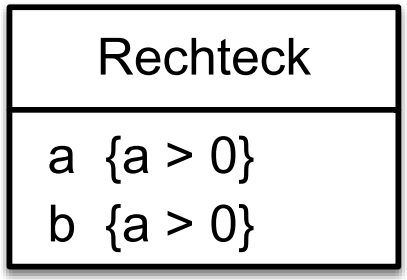
Aufgabe zu Objekt- und Klassendiagramm

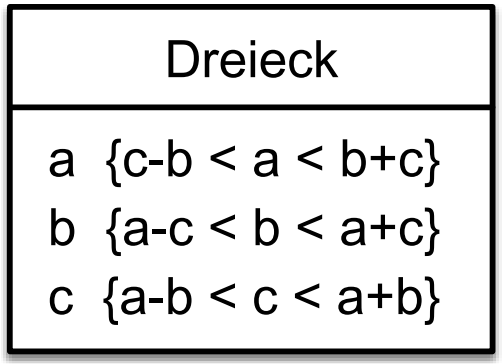
Zeichnen Sie ein UML-Klassendiagramm zu folgendem Objektdiagramm. Geben Sie dabei Multiplizitätsangaben an.



Zusicherungen / Constraint

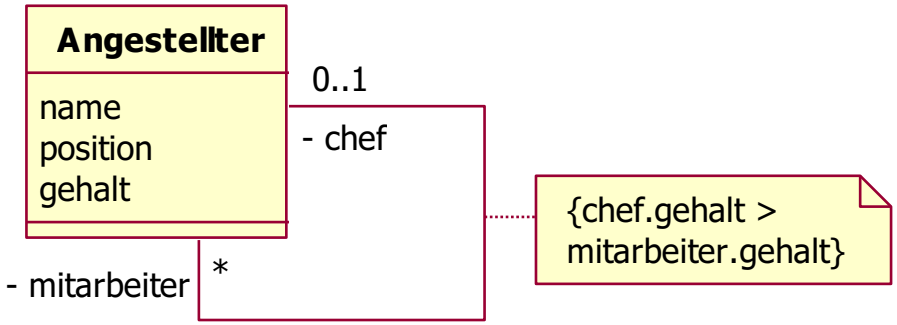
- Einschränkung / Zusicherung / Constraint
  - Ausdruck, der die möglichen Inhalte, Zustände oder die Semantik eines Modellelementes einschränkt
  - Ein Constraint muss stets erfüllt sein
- Weitere Beispiele für Constraints
  - Rechtwinkliges Dreieck
  - Mindestlohn ist 4000€ bei 20-jähriger Betriebszugehörigkeit
  - Erziehungsberechtigter ist Pflicht, wenn Alter unter 18 Jahre

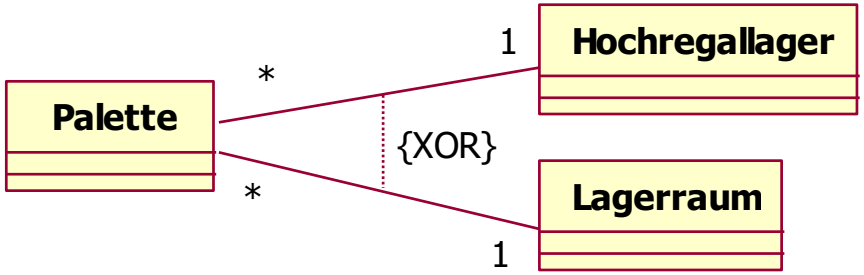




Zusicherungen / Constraint

- Komplexere Constraints

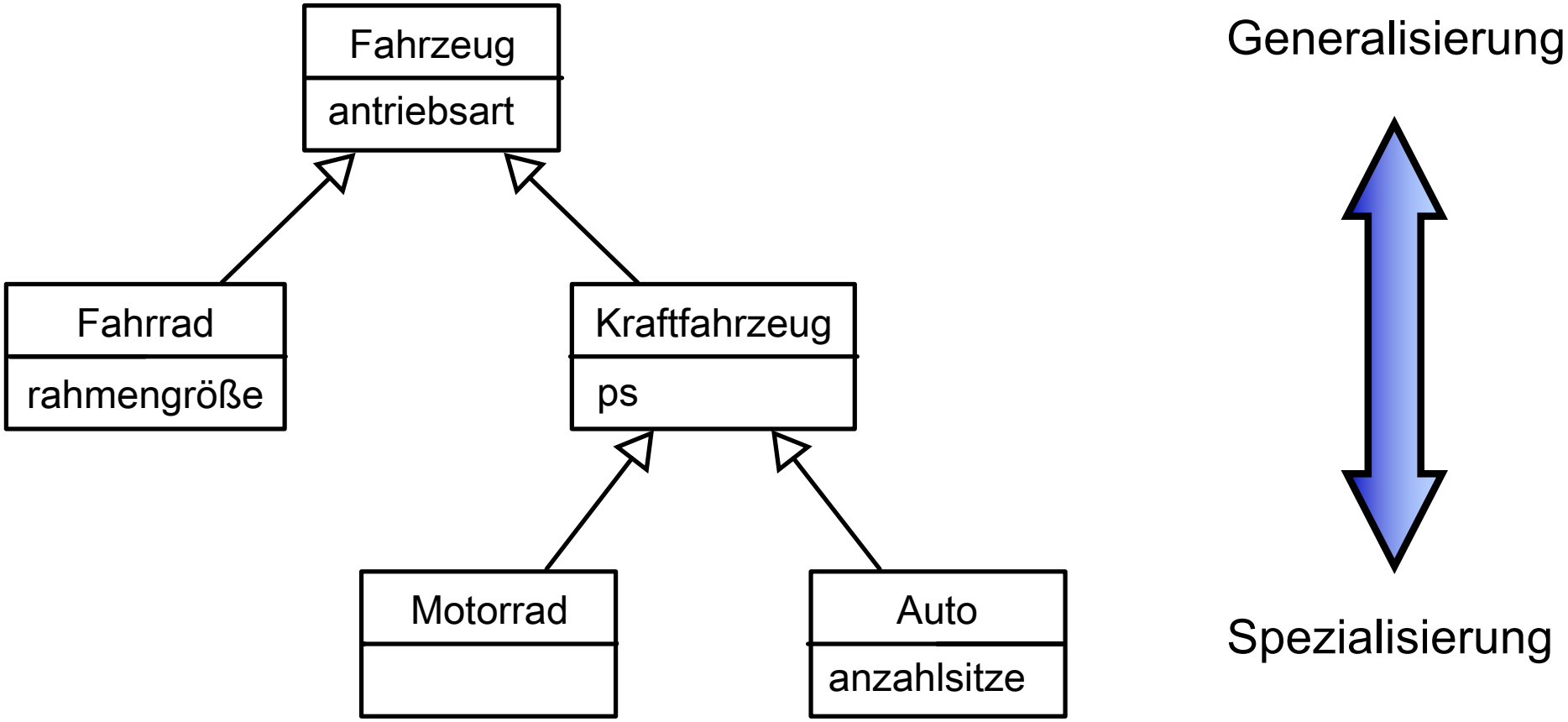




- Object Constraint Language (OCL)
  - Formale Sprache um weitere Semantik zu UML-Modellen hinzuzufügen

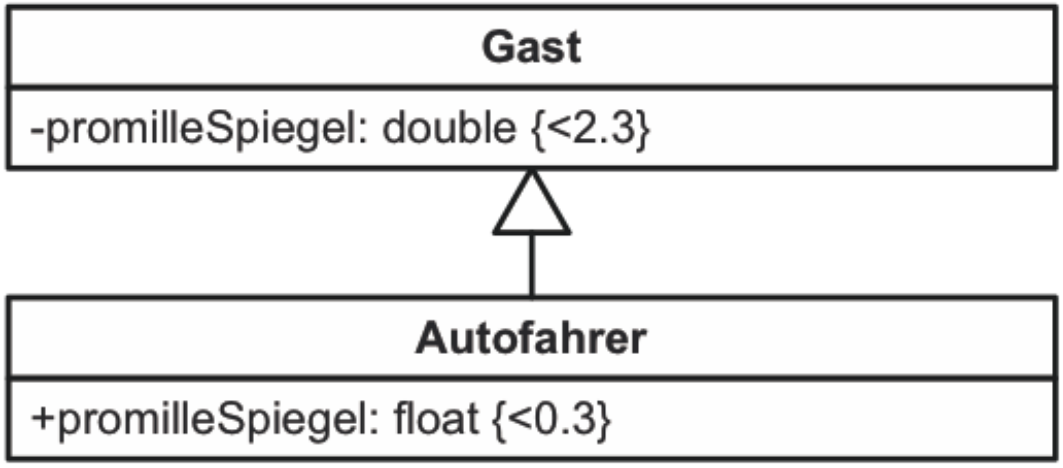
Spezialisierung/Generalisierung

- Spezialisierung / Generalisierung (Vererbung)
  - Beziehung zw. einer allgemeinen und spezialisierten Klasse



Generalisierung

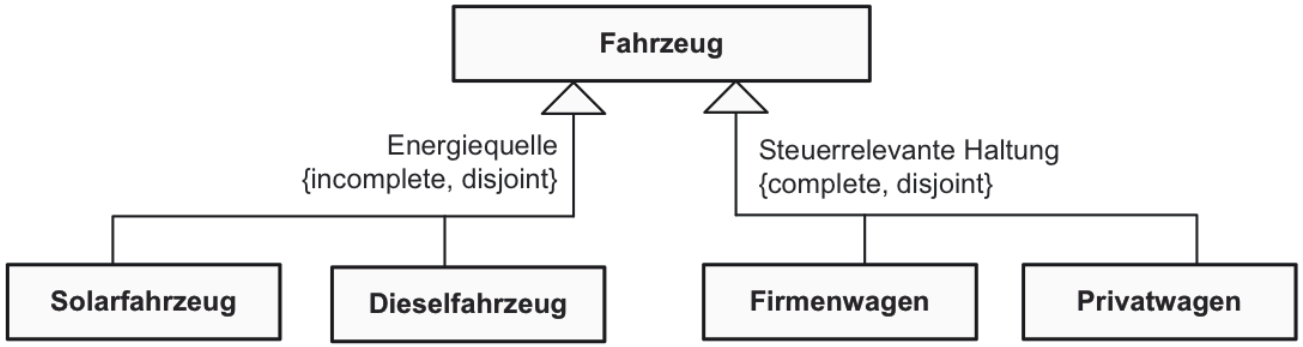
- Substituierbarkeit (Ersetzbarkeitsprinzip)
  - Instanzen von Subklassen können jederzeit anstelle von Instanzen der Superklassen eingesetzt werden
  - Bedingungen von Superklassen dürfen eingeschränkt werden



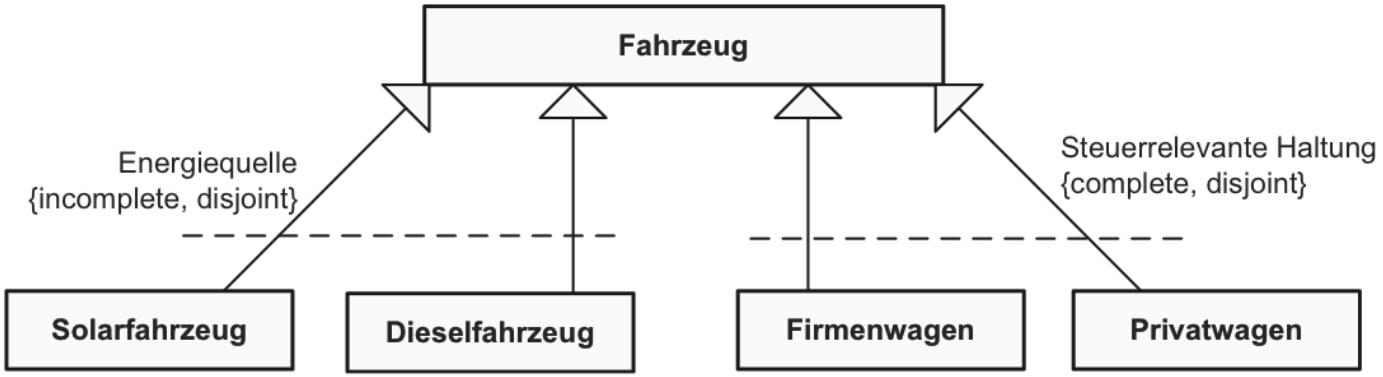
Quelle: C. Rupp. S. Queins & die SOPHISTen: UML 2 glasklar

Generalisierungsmenge

- Generalisierungsmenge (Generalization Set)
  - Zusammenfassung von Subtypen nach einem bestimmten Schema



  - Alternativnotation



Quelle: C. Rupp. S. Queins & die SOPHISTen: UML 2 glasklar

Generalisierungsmenge – Begriffe

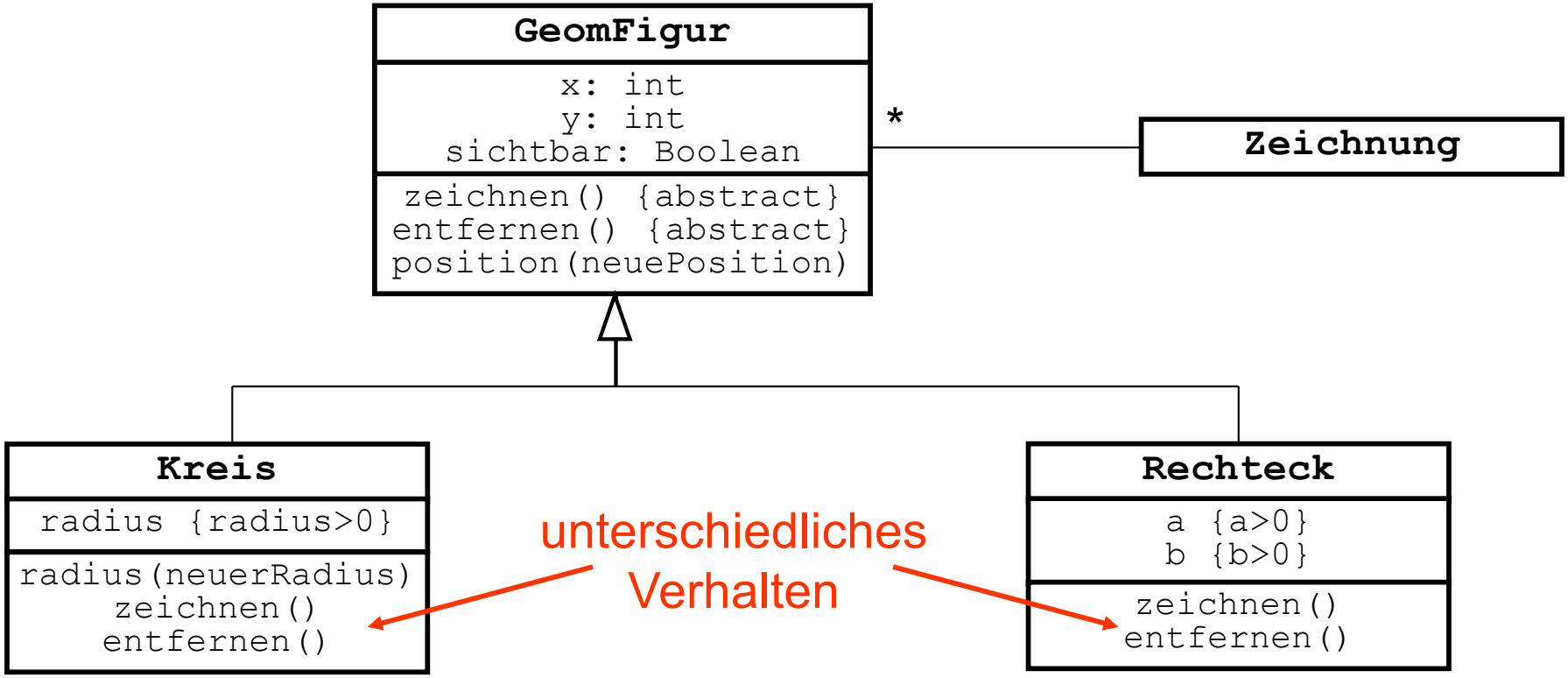
- Diskriminator
  - Unterscheidungsmerkmal, der die hierarchische Strukturierung beschreibt
- Schlüsselworte bei Generalisierungseigenschaft
  - **Complete**: Elemente der Generalisierungsmenge stellen alle sinnvoll denkbaren Spezialisierungen dar, Beispiel: Person – Mann und Frau
  - **Incomplete**: Gegenteil von complete, Beispiel: es fehlt Fahrzeug mit Gasantrieb
  - **Disjoint**: Keine Instanz eines Subtyps ist gleichzeitig Instanz eines anderen Subtyps, Beispiel: dieselgetriebenes Solarfahrzeug
  - **Overlapping**: Es kann Instanzen geben, die Ausprägungen von mehr als einem Subtyp sind

Beispiele

- Definieren Sie das Klassendiagramm inkl. Spezialisierungshierarchie für folgende Beispiele:
  - Person, Angestellter, Student, Hilfskraft
  - Computer, Laptop, PC, Festplatte, Akku
  - Fahrzeug, Auto, Motorrad, Fahrrad, Tandem, Motor
  - Buch, Roman, Kinderbuch, Fachbuch, Informatikbuch, Krimi, Autor

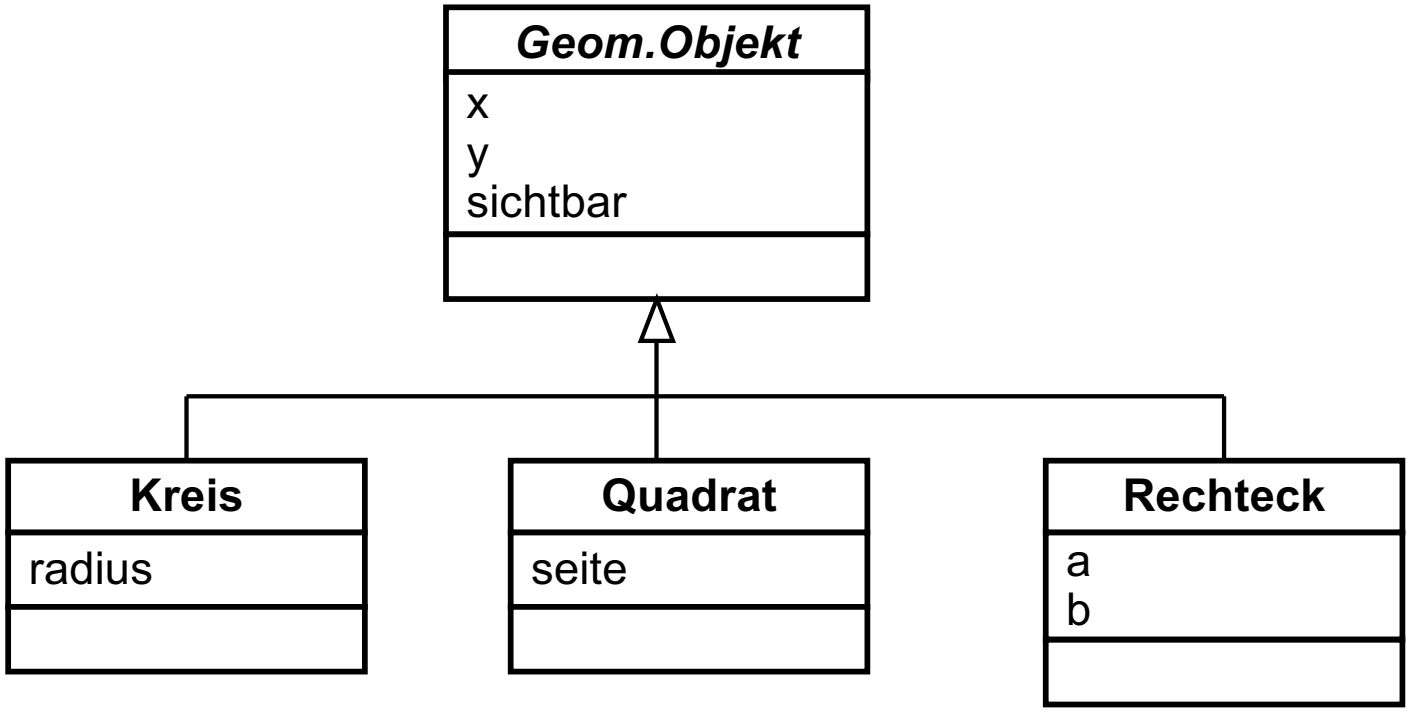
Polymorphismus

- Polymorphie-Prinzip
  - Polymorphie heißt, dass eine Operation sich in unterschiedlichen Klassen unterschiedlich verhalten kann
  - Einer der Eckpfeiler der Objektorientierung



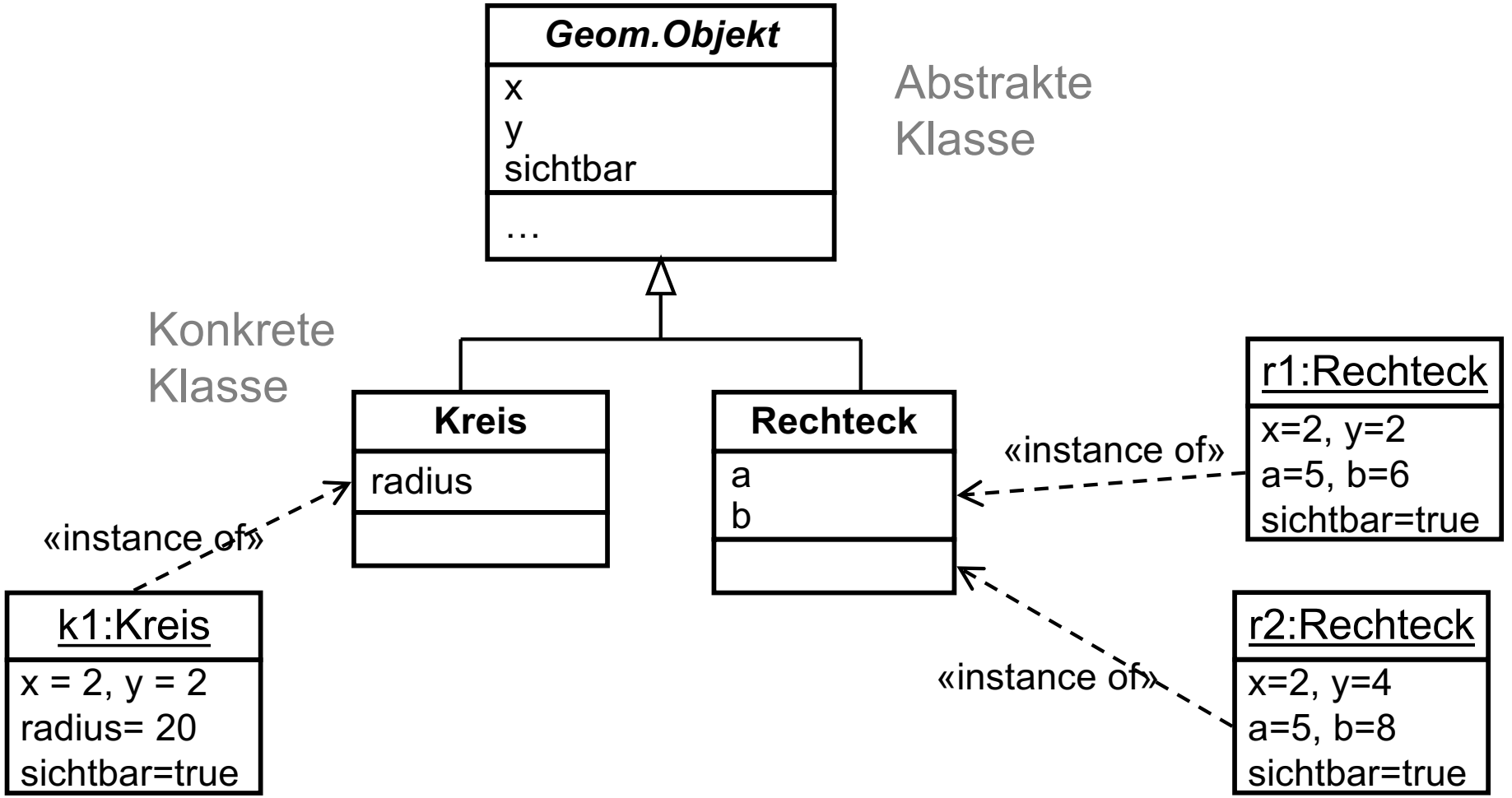
Abstrakte Klassen

- Abstrakte Klassen
  - Von abstrakten Klassen können keine Objekte erzeugt werden
  - Kennzeichnung durch kursiv geschriebenen Namen, alternativ zusätzlich durch Angabe von `{abstract}`



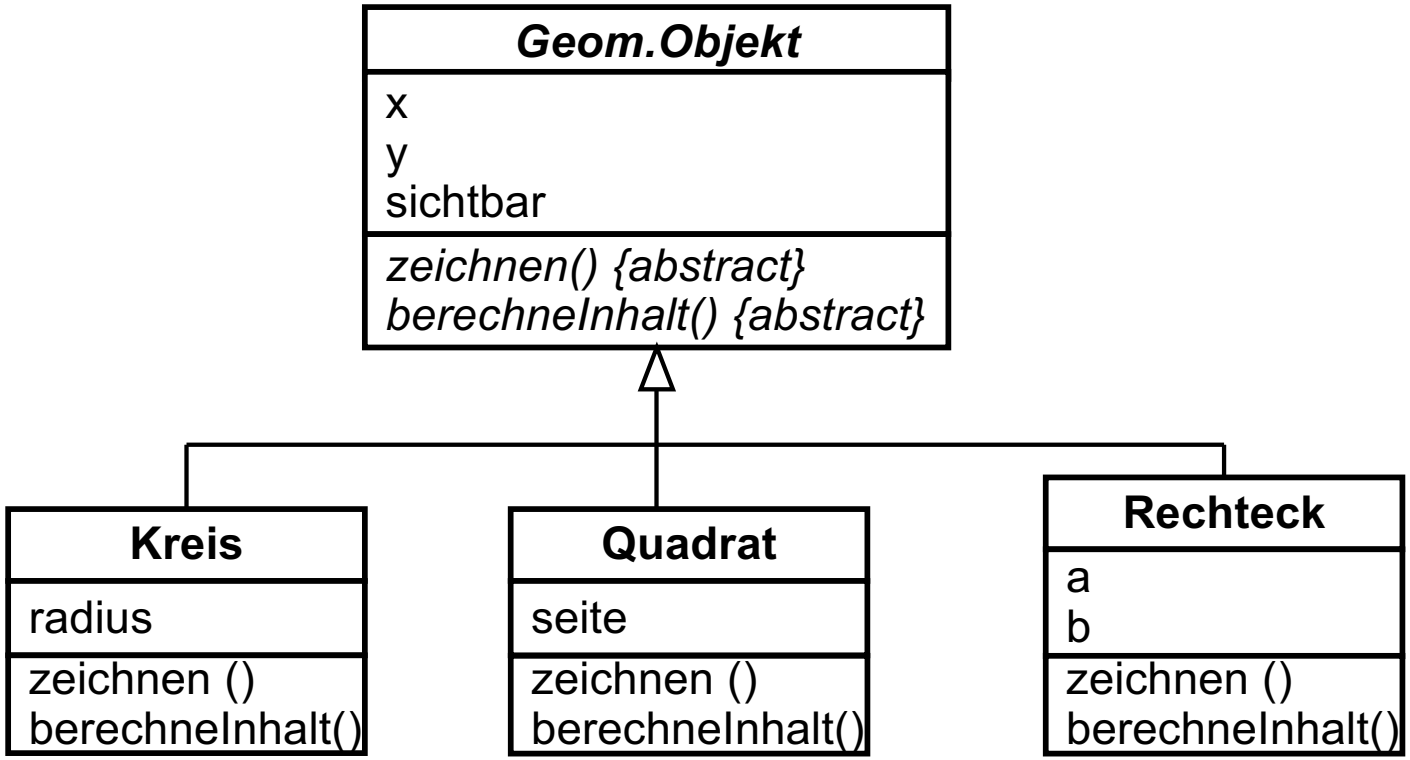
Abstrakte Klassen

- Zu abstrakten Klassen existieren keine Instanzen



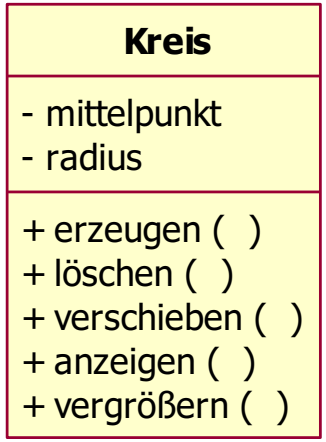
Abstrakte Methoden

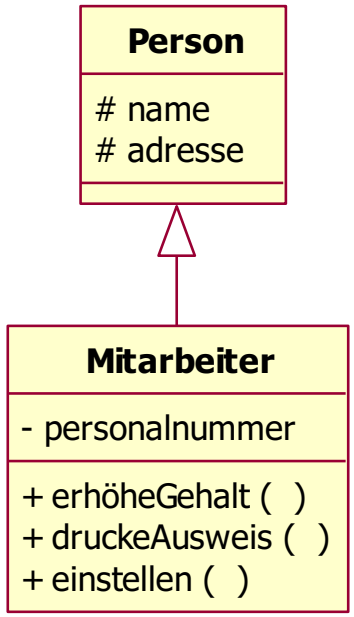
- Abstrakte Methoden
  - Kennzeichnung durch Angabe von `{abstract}`
  - Abstrakte Methoden können nur in abstrakten Klassen vorkommen
  - Implementierung in einer Unterklasse



Attribute - Sichtbarkeit

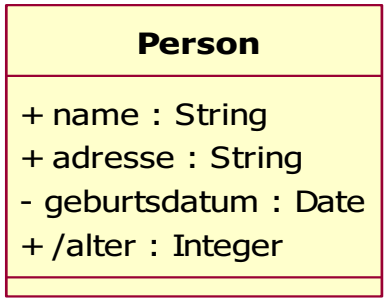
- Zugriffsmöglichkeiten
  - \+ *public*: für alle sichtbar und benutzbar
  - \# *protected*: Die Klasse selbst, ihre Unterklassen sowie die als friend deklarierten Klassen haben Zugriff
  - – *private*: Nur die Klasse selbst und die als friend deklarierten Klassen kommen an private Attribute heran
  - ~ *package*: Nur die Klasse im gleichen Paket haben Zugriff auf private Attribute





Attribute

- Abgeleitete Attribute
  - Attribut, das aus Werten anderer Attribute berechnet wird
  - Attributwert kann nicht direkt geändert werden
  - Kennzeichnung durch „/“
- readonly
  - Attributwert darf nicht geändert werden, z.B. kontonr {readOnly}
- Optionale und obligatorische Attribute
- Bag, Sequence, Union
  - Attributwert besteht aus einer Menge von Werten

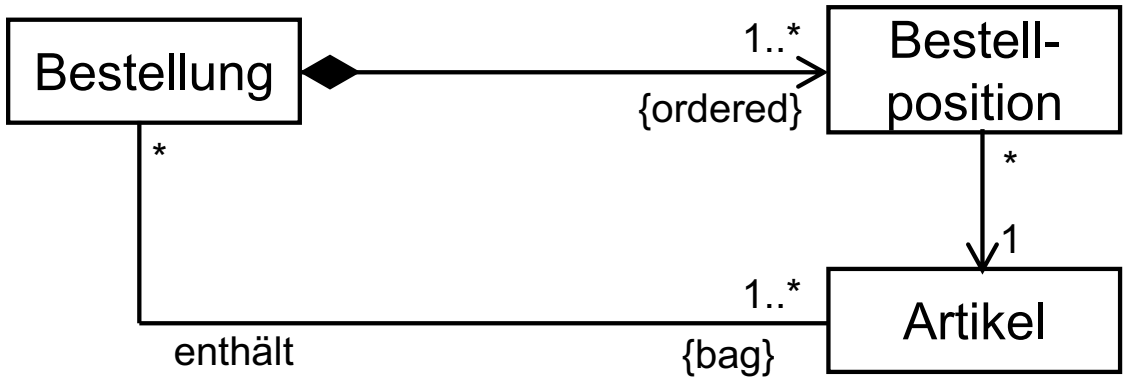


Operationen

- Abstrakte Operation
  - Besteht nur aus Signatur
- Sichtbarkeit
  - Vergl. Attribute
- Readonly-Operation
  - Keine Veränderung der Attributwerte
- Notation Signatur
  - `Sichtbarkeit name (Parameterliste): Ergebnistyp {Eigenschaftswert}`

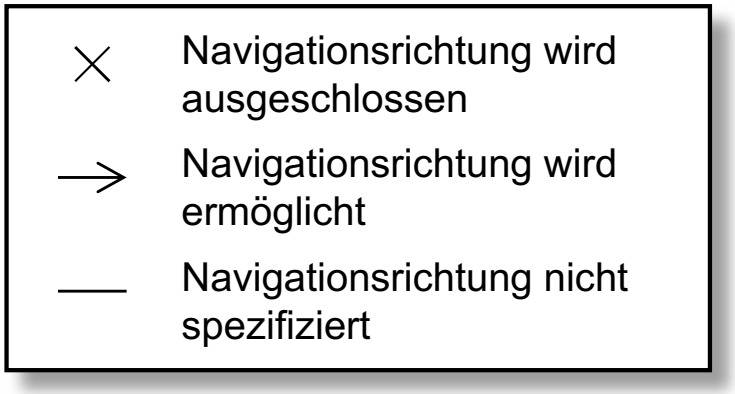
Assoziationen

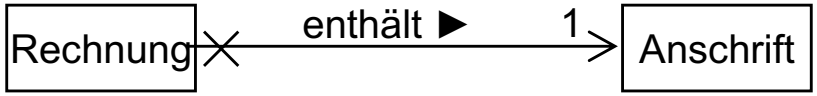
- Änderbarkeit
  - `readonly`
  - `frozen`
  - `addonly`
- Weitere Eigenschaftswerte
  - `ordered`: definiert Ordnung auf Menge der Objektbeziehungen
  - `bag`: definiert, dass ein Objekt mehrmals in der Menge der Objektbeziehungen vorkommen darf
  - `sequence`: definiert Kombination von `ordered` und `bag`



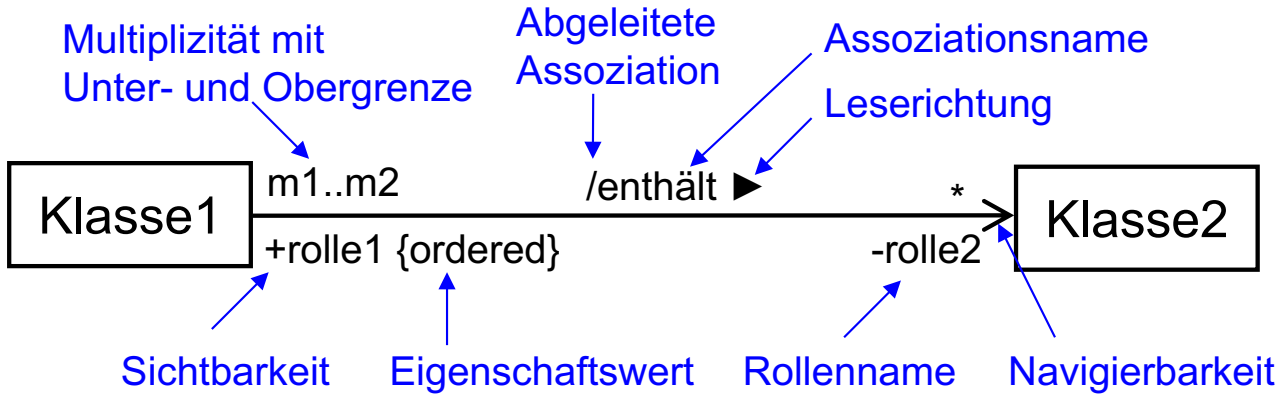
Assoziationen

- Navigierbarkeit
  - Spezifikation in welcher Richtung navigiert werden kann



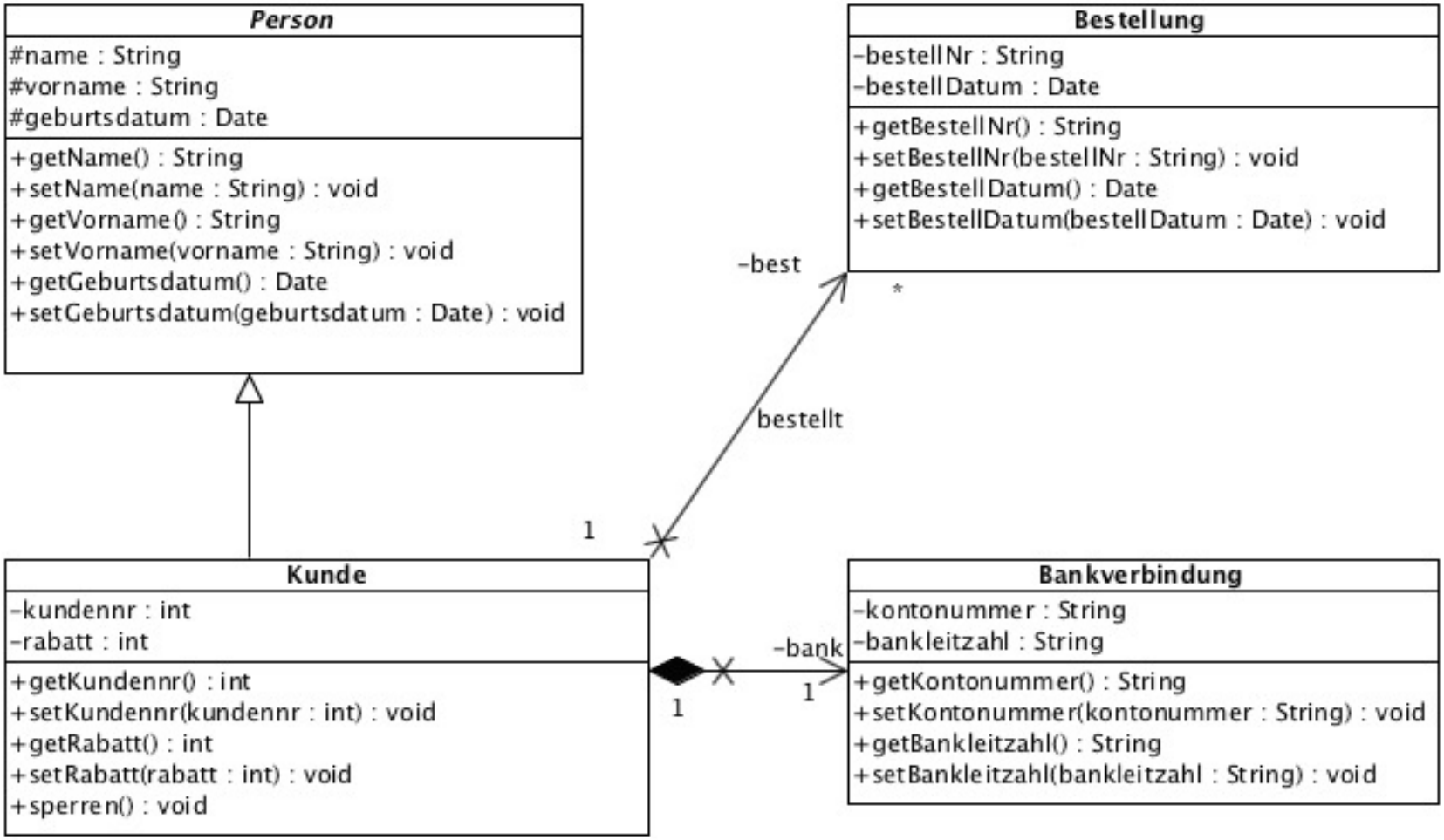


- Notation für Assoziation



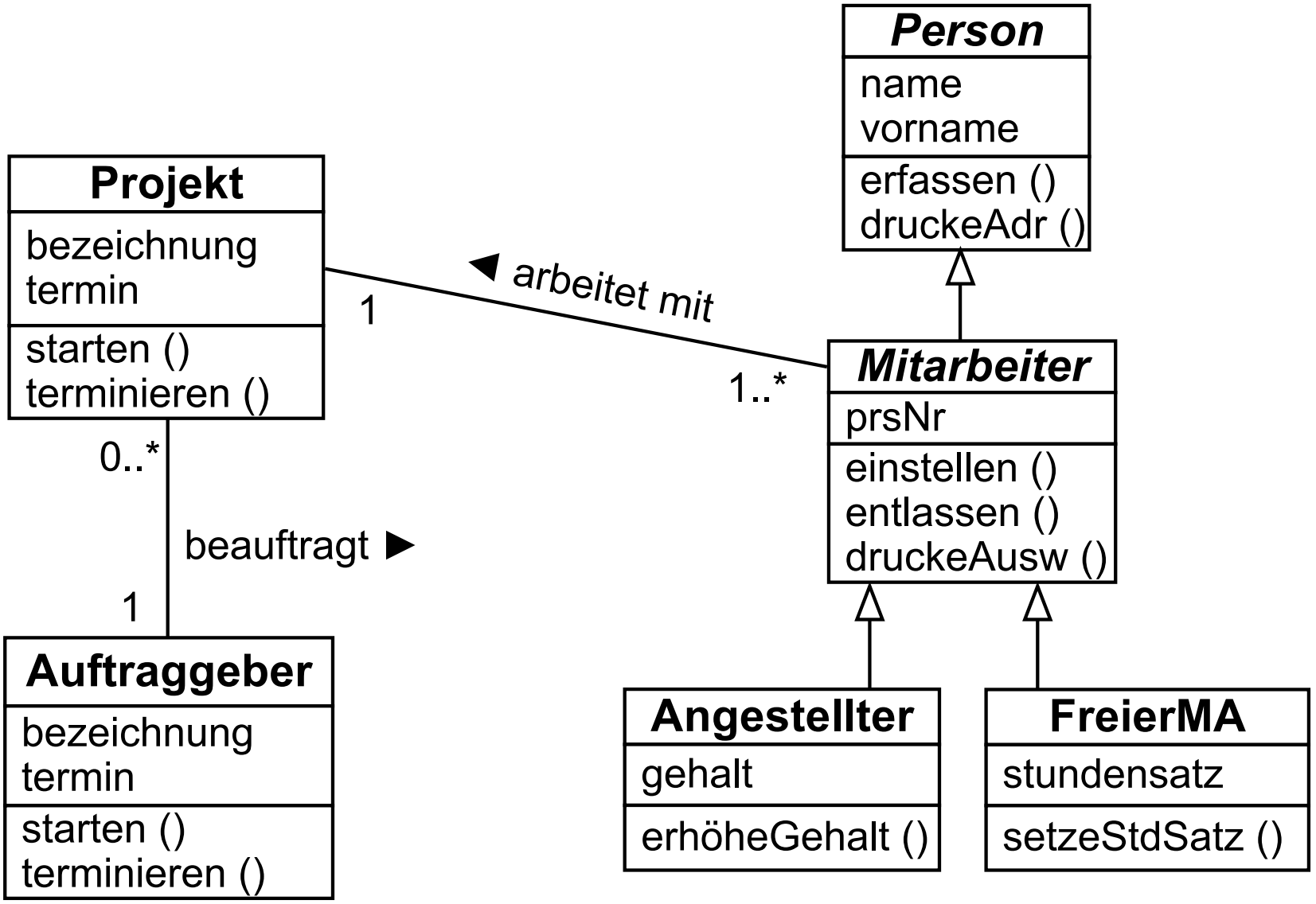
Generierung Programmcode

- Welcher Java-Code wird aus der Klasse „Kunde“ generiert?

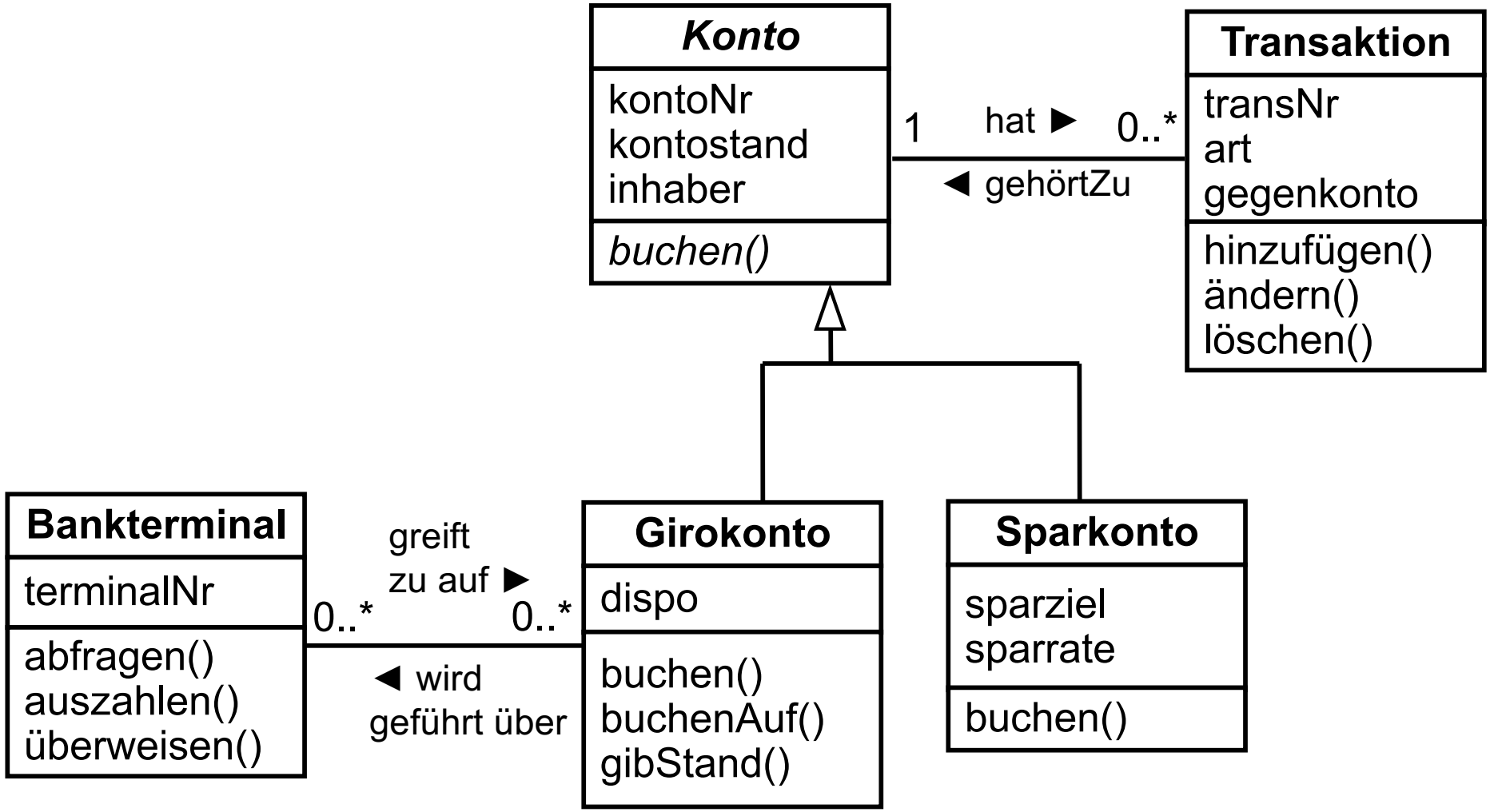


Klassendiagramm

Beispiel Projektverwaltung

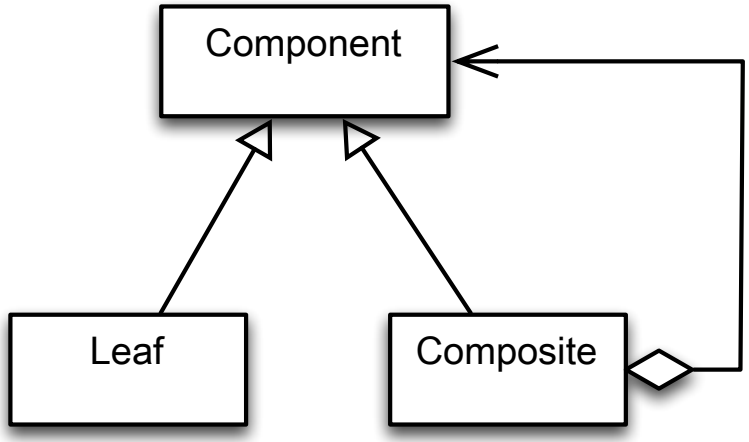


Klassendiagramm
Beispiele

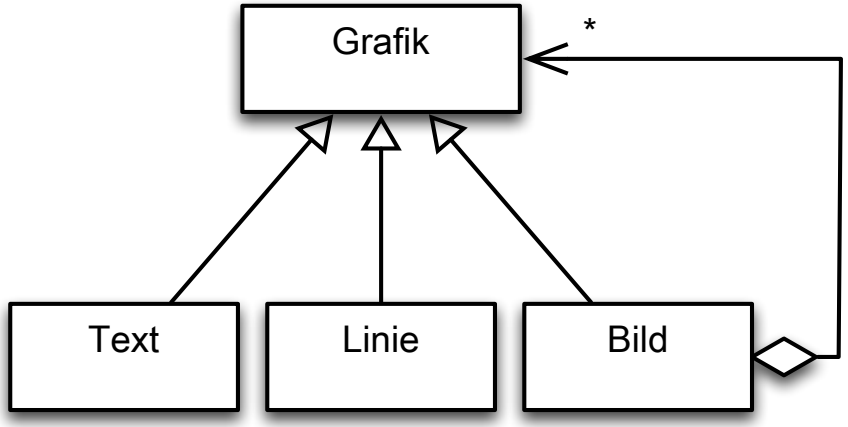


Klassendiagramm
Beispiele

- Design Pattern Composite

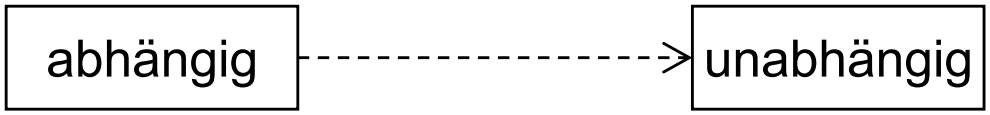


  - Konkretes Beispiel



Abhängigkeitsbeziehung

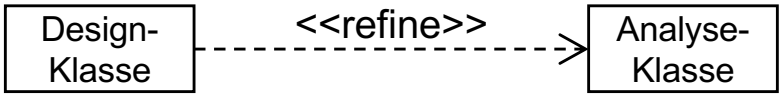
- Definition
  - Änderung in einem (unabhängigem) Element macht eine Änderung in dem anderen (abhängigen) Element notwendig
- Beispiele für Verfeinerungsbeziehungen
  - Beziehung zwischen Analyse- und Design-Version
  - Beziehung zwischen einer sauberen Implementierung und einer optimierten, aber eventuell diffizilen Variante
  - Beziehung zwischen einer Schnittstellenklasse und einer Klasse, die diese Schnittstelle umsetzt
  - Beziehung zwischen zwei unterschiedlich granulierten Elementen



Arten von Abhängigkeitsbeziehungen

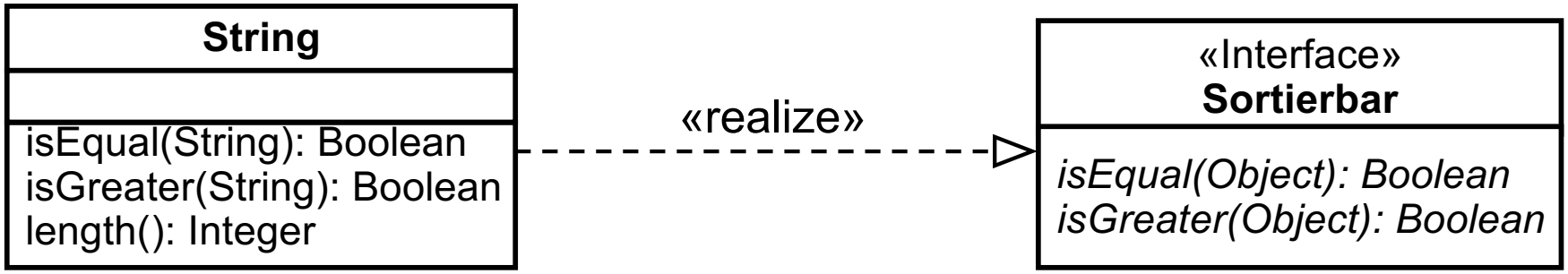
Schlüsselwort	Bedeutung
<<call>>	Das abhängige Element ruft eine Operation des unabhängigen auf.
<<create>>	Das abhängige Element erzeugt Exemplare des unabhängigen Elements.
<<derive>>	Das abhängige Element ist vom unabhängigen Element abgeleitet.
<<instantiate>>	Das abhängige Element ist eine Instanz des unabhängigen Elementes.
<<permit>>	Das abhängige Element hat die Erlaubnis private Eigenschaften des unabhängigen Elements zu verwenden
<<realize>>	Das abhängige Element implementiert beispielsweise eine Schnittstelle oder ein abstraktes Element.
<<refine>>	Das abhängige Element befindet sich auf einem konkreteren semantischen Niveau als das unabhängige Element.
<<trace>>	Das abhängige Element führt zum unabhängigen Element, um die semantische Abhängigkeiten nachverfolgen zu können.
<<use>>	Das abhängige Element benutzt das unabhängige Element für seine Implementierung.

Beispiel:



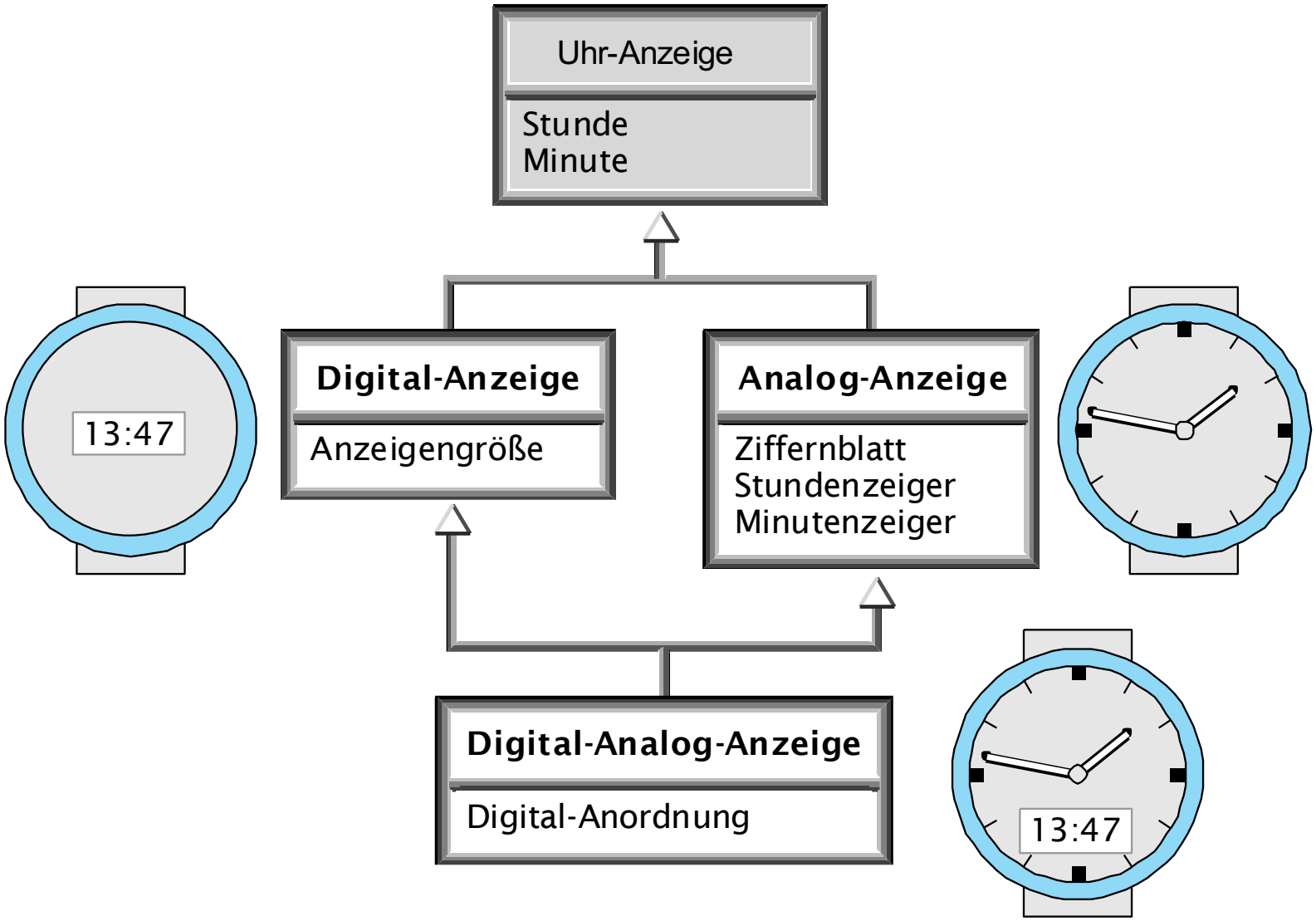
Interfaces, Schnittstellen

- Beschreibung
  - Spezifikationen des externen Verhaltens von Klassen
  - Beinhalten eine Menge von Signaturen für Operationen
  - Klassen, die diese Schnittstelle bereitstellen wollen, müssen diese Operationen implementieren
  - Schnittstellen haben gewöhnlich keine Attribute
  - Schnittstellen können durch Generalisierungsbeziehungen erweitert werden



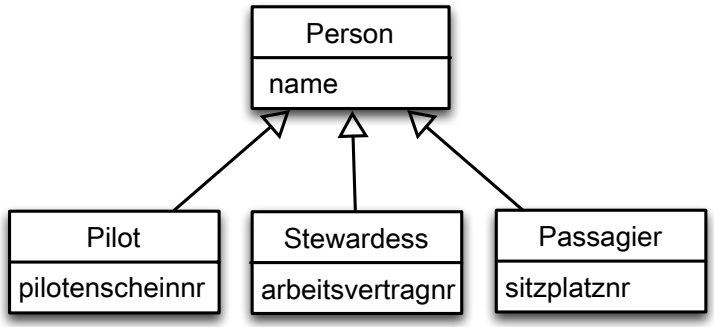
Klassendiagramm

Beispiel Mehrfachvererbung

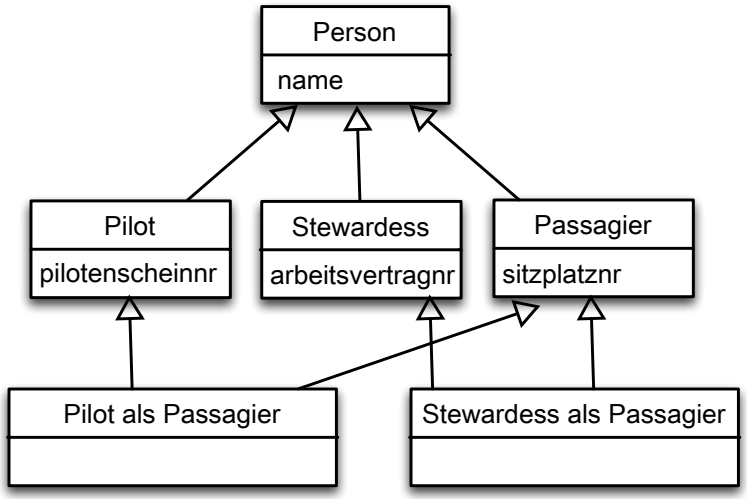


Verwendung der Spezialisierung

- Probleme
  - Was passiert, wenn Stewardess als Passagier mitfliegt ?

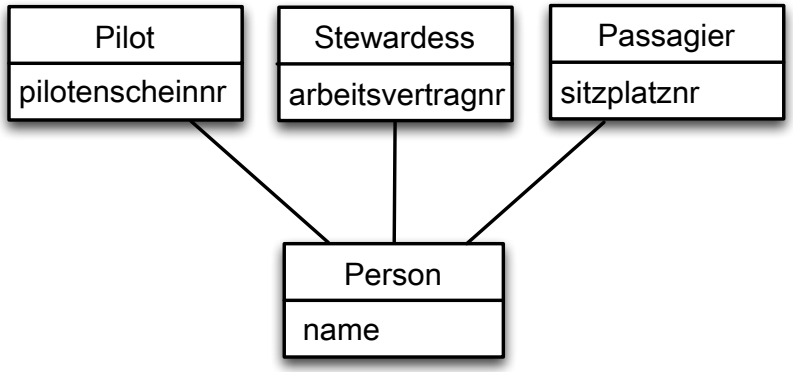


  - Schlechte Verwendung der Spezialisierung (Explosion der Klassen)

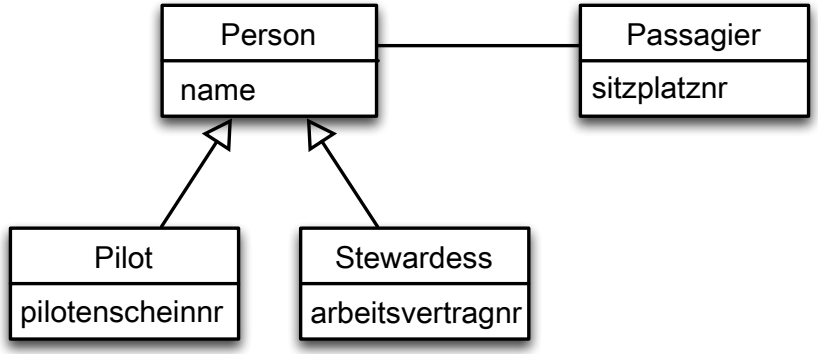


Alternativen zur Spezialisierung

- Verwendung der Delegation
  - Passagier erhält eine Referenz auf die betreffende Person
  - Operationen, die die Person betreffen, werden an Klasse Person delegiert



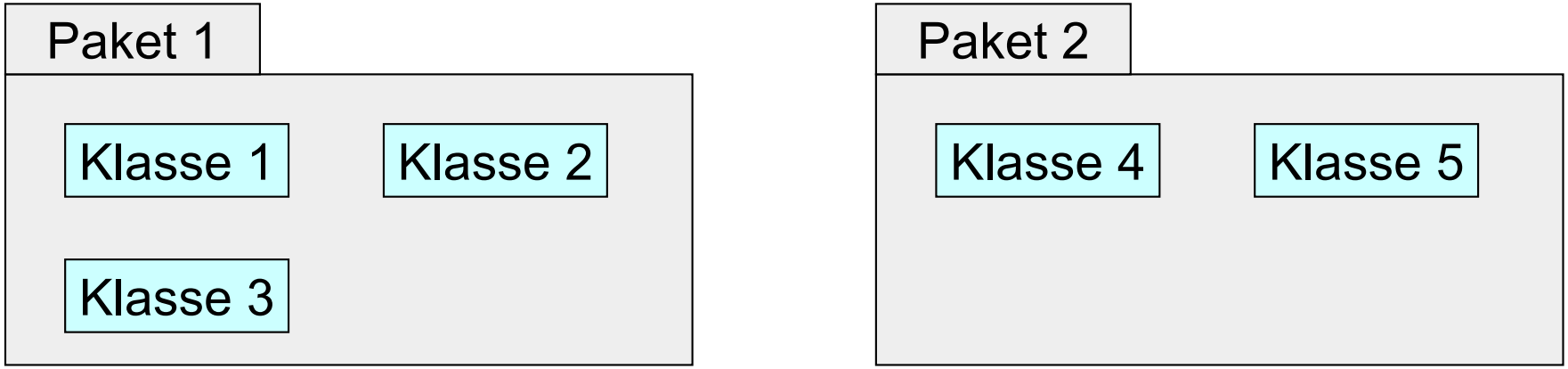
  - Mischform



Eventuell Umbenennung
Passagier → Reservierung

Paketdiagramme

- Pakete sind Ansammlungen von Modellelementen beliebigen Typs
  - Kleinere überschaubare Einheiten
  - Definition eines Namensraums, innerhalb dessen die Namen der enthaltenen Elemente eindeutig sein müssen
  - Hierarchische Gliederung möglich
  - Zitieren einer Klasse in einem anderen Paket:
    `Paketname::Klassenname`
- Unerlässlich bei großen Projekten!

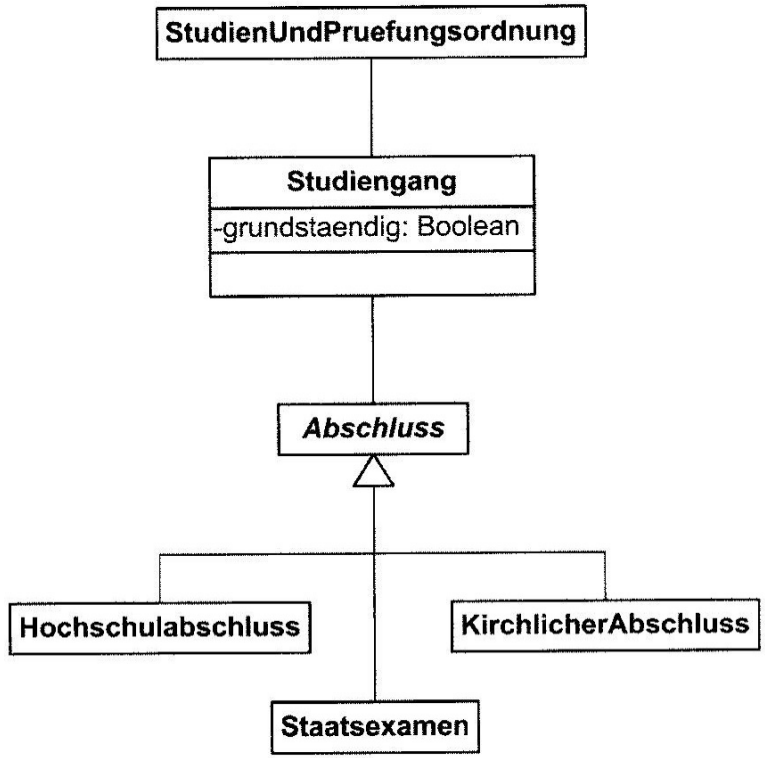


Formalisierung mit UML

§ 30

Studiengänge

(1) Ein Studiengang ist ein durch Studien- und Prüfungsordnungen geregeltes, auf einen bestimmten Abschluss (Hochschulabschluss, Staatsexamen, kirchlicher Abschluss) ausgerichtetes Studium. Grundständige Studiengänge sind Studiengänge, die zu einem ersten Abschluss im Sinne des Satzes 1 führen.



Probleme

- Es geht verloren, wann ein Studiengang grundständig ist
- Darf es pro Studiengang beliebig viele SPOs geben?
- Darf eine SPO mehrere Studiengänge regeln?
- Ist Aufzählung der Abschlussarten vollständig?

Quelle: Drachenfels: Informatik Spektrum 39 3 2016

Aufgabe zu Klassendiagramm

Erstellen Sie ein UML-Klassendiagramm anhand folgender Beschreibung

Eine Tagung (z.B. Softwaretechnik-Tagung in Hamburg) ist zu organisieren. Für jeden Teilnehmer der Tagung werden der Name, die Adresse und der Status (Student, Mitglied, Nichtmitglied) gespeichert. Jeder Teilnehmer kann sich für ein oder mehrere halbtägige Tutorien, die zusätzlich zum normalen Tagungsprogramm angeboten werden, anmelden. Für jedes Tutorium werden dessen Nummer, die Bezeichnung sowie das Datum gespeichert. Alle Tutorien kosten gleich viel. Damit ein Tutorium stattfindet, müssen mind. 10 Anmeldungen vorliegen. Jedes Tutorium wird von genau einem Referenten angeboten. Für jeden Referenten werden dessen Name, Adresse und Firma gespeichert. Ein Referent kann sich auch für ein oder mehrere Tutorien anderer Referenten anmelden und kann bei diesen kostenlos zuhören. Diese Anmeldung zählen bei der Ermittlung der Mindestanmeldungen nicht mit. Ein Teilnehmer kann nicht gleichzeitig Referent sein. Ein Referent kann mehrere Tutorien anbieten. Ein Teilnehmer kann sich in der Tagungsanmeldung auch für einige Rahmenprogramme (z.B. Besuch eines Musicals) eintragen lassen. Für jedes Rahmenprogramm werden dessen Bezeichnung, das Datum und die Kosten gespeichert.

Quelle: Balzert: Lehrbuch der Objektmodellierung

Aufgabe zu Paketbildung

Von den folgenden Klassen gehört jede zu einem Paket. Gruppieren Sie die aufgeführten Klassen in Pakete. Wählen Sie für jedes Paket einen aussagefähigen Namen.

Artikel
Bankverbindung Kunde
Bestellposten
Bestellung
Kunde
Lager
Lagerplatz
Lagerverwalter
Lieferant
Reklamation
Rücksendung
Stornierung
Zahlungsart bei Bestellung

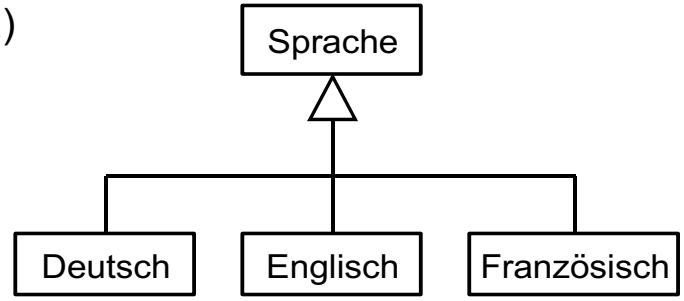
Quelle: Balzert: Lehrbuch der Objektmodellierung

Kontrollfragen

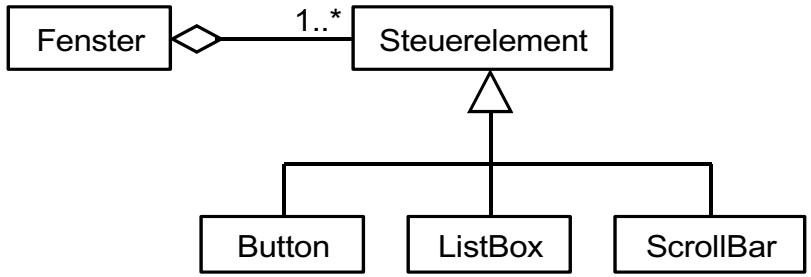
Systemanalyse Klausur WS 2004/05

Sind die folgenden UML-Klassendiagramme sinnvoll modelliert? Begründen Sie jeweils Ihre Entscheidung.

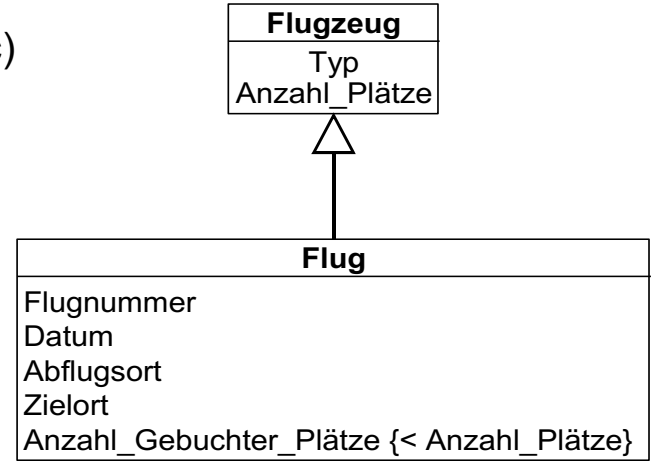
a)



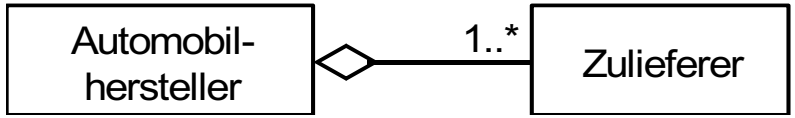
b)



c)



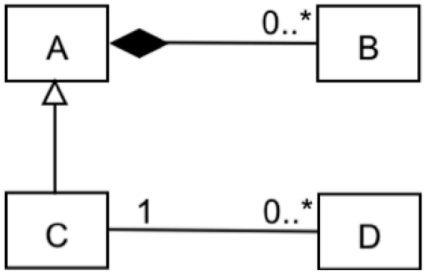
d)



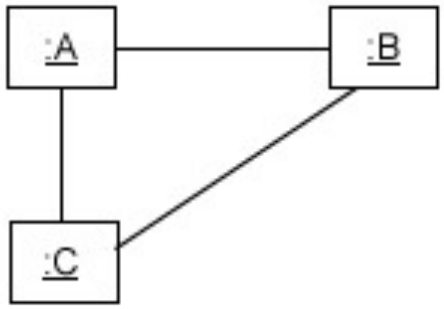
Kontrollfragen

Systemmodellierung Klausur SS 2009

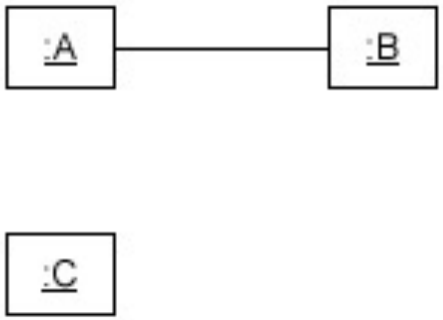
Gegeben ist folgendes Klassendiagramm. Überprüfen Sie, ob es sich bei den Objektdiagrammen a) bis f) um korrekte Instanzen des Klassendiagramms handelt. Falls nicht, begründen Sie kurz, warum die Instanzbeziehung verletzt ist.



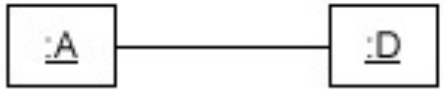
a)



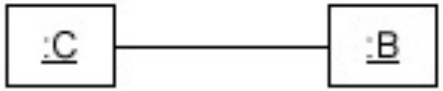
b)



c)



d)



e)

f)

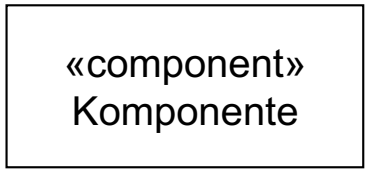
UML – Übersicht

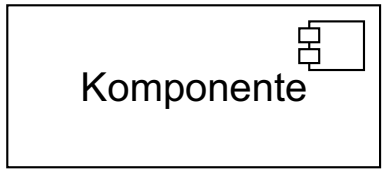
5 UML – Strukturdiagramme
- 5.1 Einführung UML
- 5.2 Objektdiagramm, Klassendiagramm
- 5.3 Komponentendiagramm

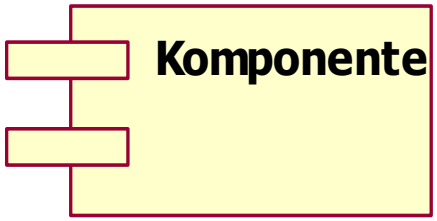
6 UML – Verhaltensmodellierung
- 6.1 Use Case-Diagramm
- 6.2 Aktivitätsdiagramm
- 6.3 Kommunikationsdiagramm
- 6.4 Sequenzdiagramm
- 6.5 Zustandsdiagramm

5.3 Komponentendiagramm

- Komponente = Softwarebaustein
  - Ausführbare und austauschbare Software-Einheit mit definierten Schnittstellen und eigener Identität
  - Definierte Schnittstellen
  - Geheimnisprinzip, Austauschbarkeit (Substituierbarkeit)
  - Besteht gewöhnlich aus einer Menge von Klassen
- Komponentenbasierte Entwicklung
  - Schnittstellen von ausgelieferten Komponenten dürfen nicht mehr modifiziert werden, interne Realisierung das geändert werden
  - Komponentenmodelle: JavaBeans, EJB, CORBA, COM, .NET

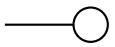






Komponentendiagramm

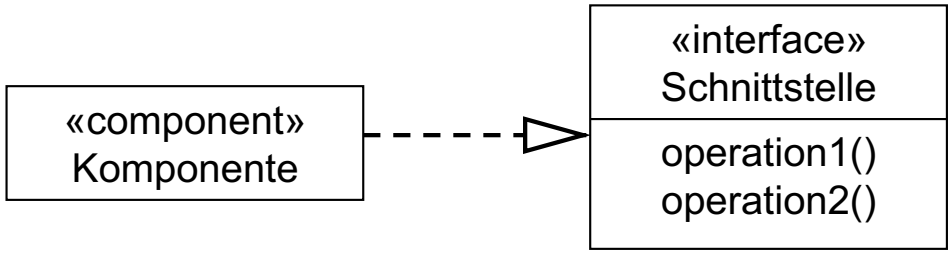
- Darstellung der Schnittstellen
  - Zur Verfügung gestellte Schnittstellen
  - Genutzte Schnittstellen

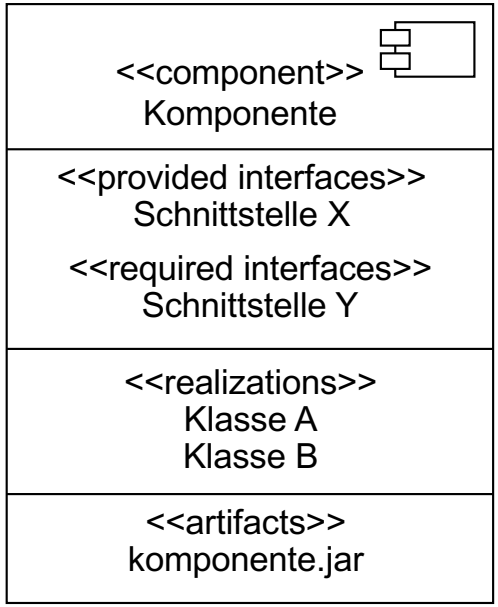


angebotene Schnittstelle

genutzte Schnittstelle

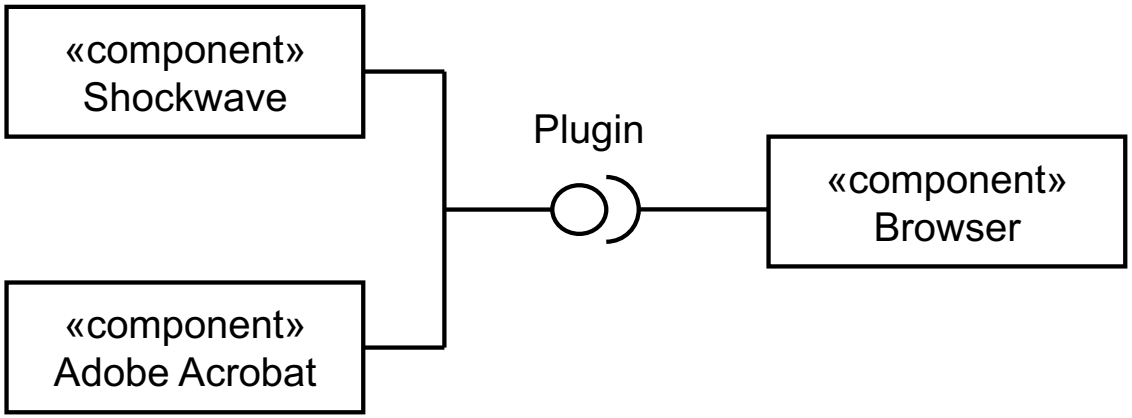
Port





Komponentendiagramm

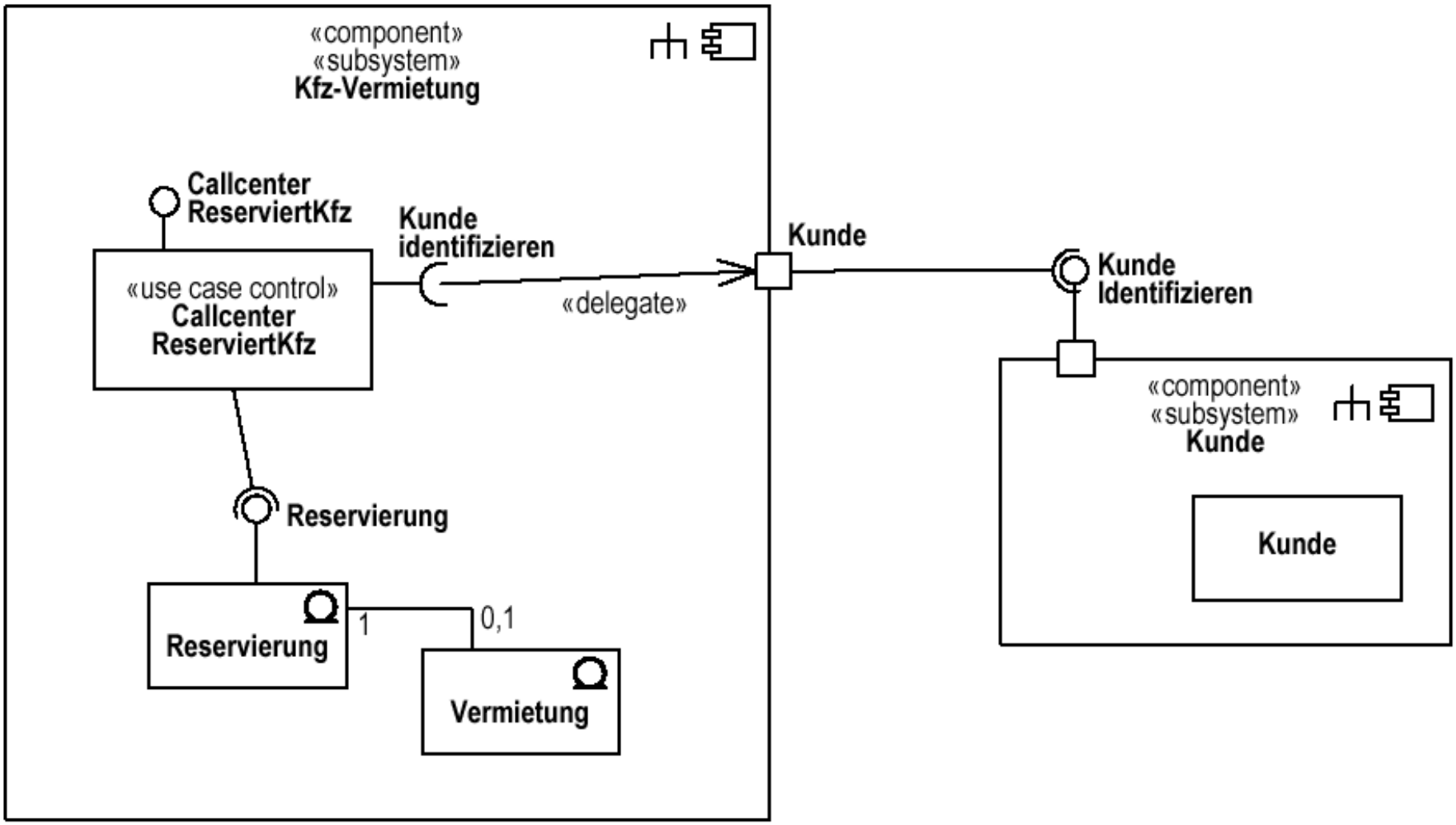
- Komponentendiagramm
  - Dokumentation der Abhängigkeiten



Komponenten

- Komponentenstruktur
  - Komponenten können intern aus mehreren Klassen bestehen (*Classifiern*)
  - Interne Sicht oder White-Box-Sicht
- Port
  - Gruppierung von bereitgestellten und geforderten Schnittstellen

Kompositionsstrukturdiagramm
Beispiel



Quelle: www.oose.de

Erstellung eines Komponentendiagramms

- Erstellen Sie ein Komponentendiagramm mit den folgenden Komponenten den den jeweils angegebenen Schnittstellen, die diese Schnittstellen zur Verfügung stellen und benutzen
  - Zimmerbelegung: bietet Zimmeranfrage und Buchung
  - Preisberechnung: bietet Preisliste und Sonderangebote
  - Zimmerreservierung: bietet Stornierung und Reservierung, benutzt Zimmeranfrage, Buchung, Preisliste und Sonderangebote
  - OnlineReservierung: benutzt Reservierung