UML - Klassendiagramme

Problemstellung

- ✓ Ausgangssituation: Sie …
  - kennen 2-3 Programmiersprachen
  - haben bisher allein oder in sehr kleinen Gruppen (<5) gearbeitet
- ✓ Neue Herausforderungen: Sie sollen …
  - mit Kollegen zusammenarbeiten, die andere Programmiersprachen nutzen
  - mit Kunden kommunizieren, die nichts von Informatik verstehen
  - komplexe, evt. verteilte Systeme realisieren, mit einer Vielzahl von Klassen, Subsystemen, verschiedensten Technologien, …
  - Entwürfe auf einer höheren Abstraktionsebene als Quellcode kommunizieren und dokumentieren
- ✓ Lösungsweg: Abstraktion des zu realisierenden Systems
  - Erst modellieren, dann programmieren!

Probleme im traditionellen SW-Prozess

- ✓ Software wird umfangreicher, nicht kleiner
- ✓ Windows NT 5.0 ~ 40 Millionen Zeilen Code
  - Kein Programmierer kann diese Menge Code alleine bewältigen.
  - Code ist oft von Entwicklern, die an der Entwicklung nicht mitgewirkt haben, nicht direkt zu verstehen
- ✓ Wir brauchen einfachere Repräsentationen für komplexe Systeme
  - Modellierung ist ein Mittel um mit Komplexität umzugehen
- ✓ Nutzen der UML
  - Verbesserte Kommunikation mit Anwendern
  - Verbesserte Kommunikation mit Programmierern

Unified Modeling Language(UML)

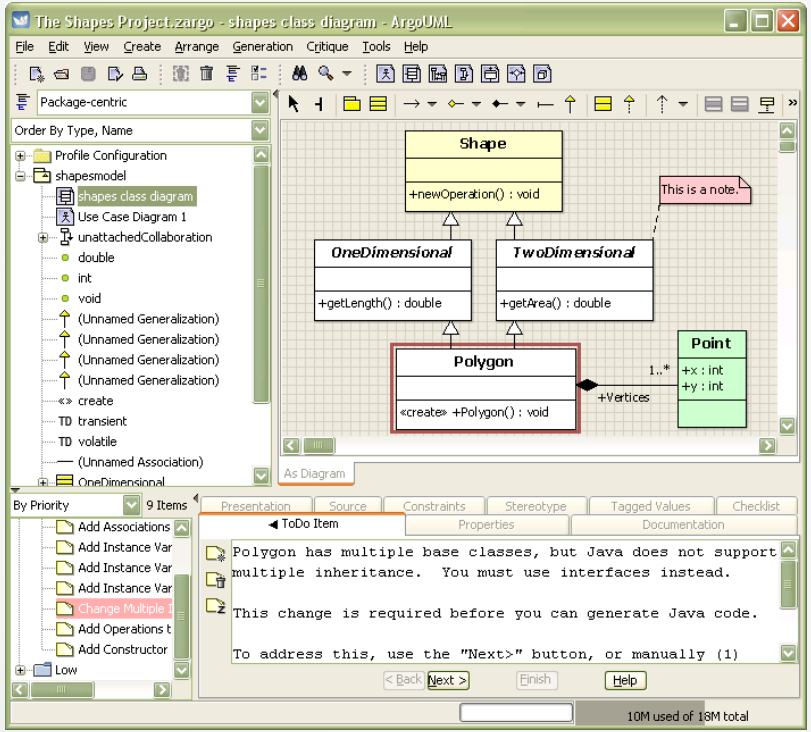
- ✓ Standardisierte graphische Notation für alle Aktivitäten der Softwareentwicklung
  - Nutzen für Anforderungserhebung, Entwurf, Implementierung, Einsatz, …
  - Besondere Unterstützung für objektorientierte Modellierung
  - Standardisiert durch die OMG (Object Management Group:www.omg.org)
- ✓ Bietet zahlreiche Diagrammtypen für verschiedene Sichten von Software
  - statische Sichten auf die Struktur
  - dynamische Sichten auf das Verhalten
- ✓ Werkzeugunterstützung für…
  - Erstellung von UML Diagrammen
  - Code-Generierung aus Diagrammen (Forward Engineering)
  - Diagramm-Generierung aus Code (Reverse Engineering)
  - Nahtlose Synchronisation von Diagrammen und Code (Round-Trip Engineering)

UML Versionen

- ✓ UML 1 (1997)
  - OMG-Spezifikation eines Modellierungsstandards
  - OMG (Object Management Group): Verband führender Softwareunternehmen und Standardisierungsgremium für
  - objektorientierte Systementwicklung
- ✓ UML 2.0 (2005)
  - Neue Diagrammtypen
  - Modifikation bestehender Diagrammtypen
  - Idee: Model-Driven Software Engineering(MDSE): Ablösung von Quellcode als Hauptartefakt
  - Ersetzen von durch Modelle, aus denen Software generiert werden

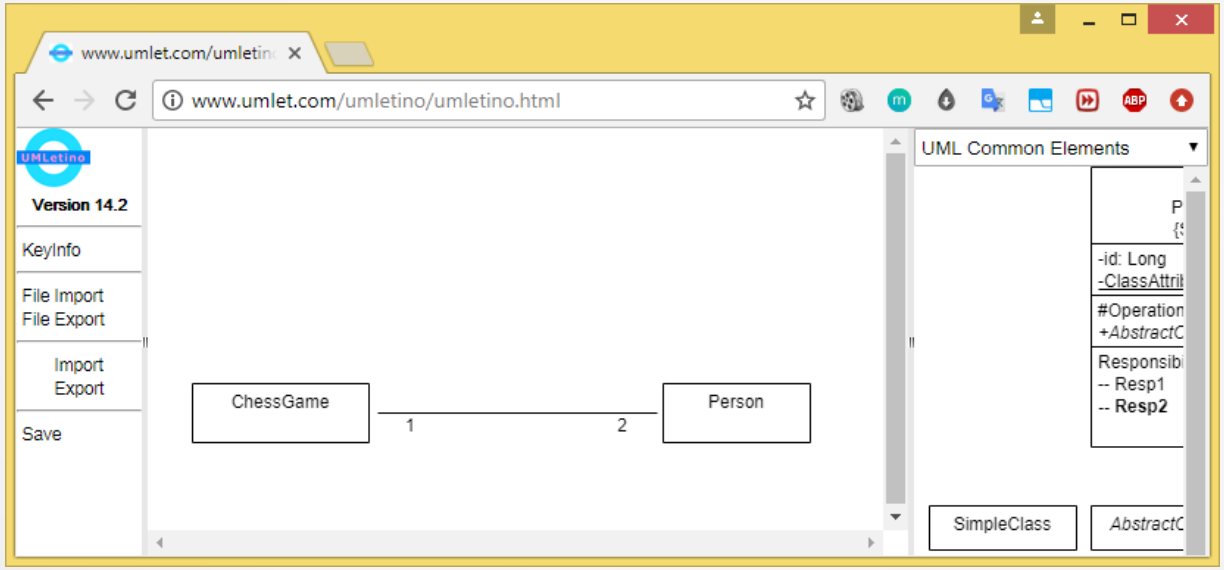
UML - offline: ArgoUML

http://argouml.stage.tigris.org

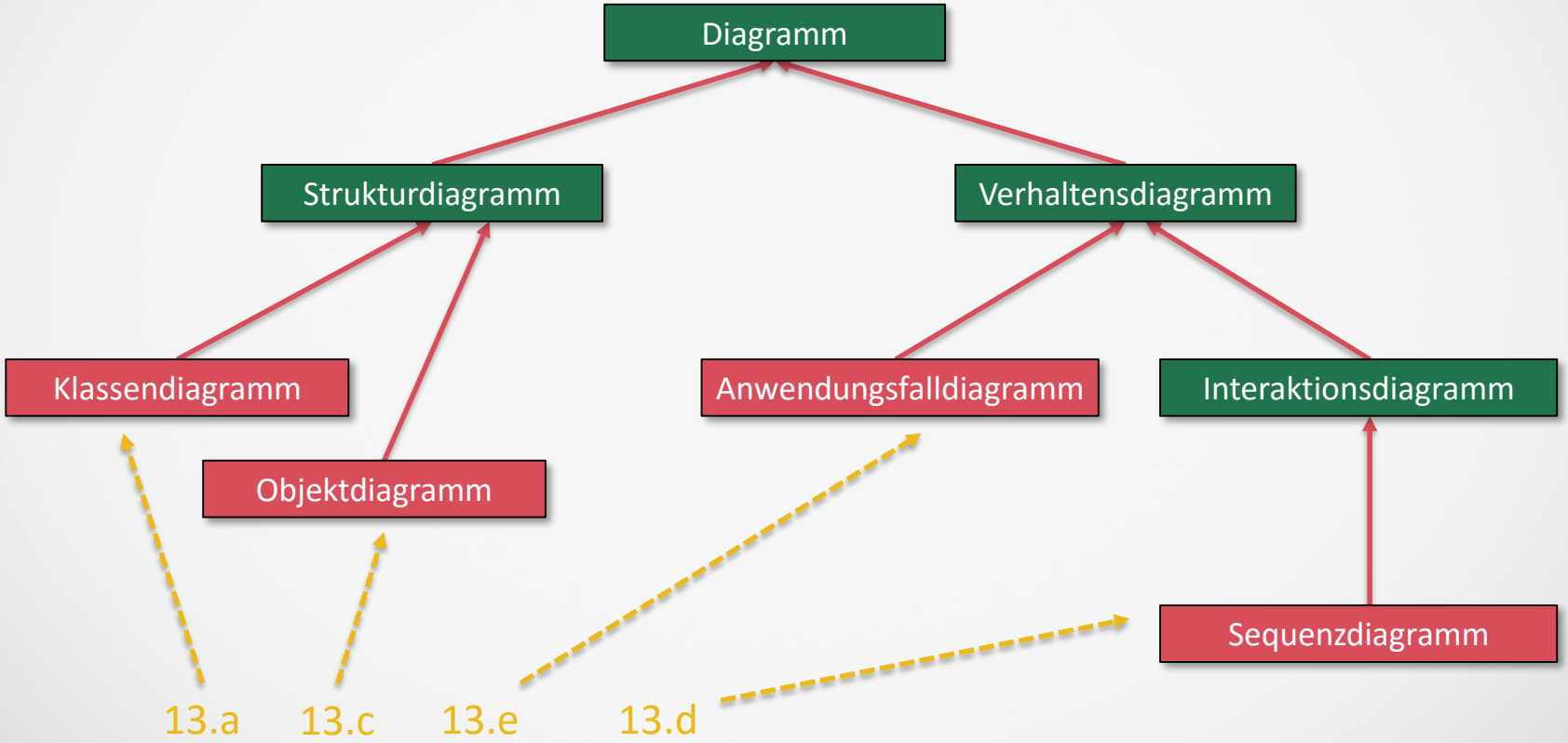


UML online: UMLet

www.umlet.com/umletino/umletino.html



Arten von UML-Diagrammen



Klassendiagramm I

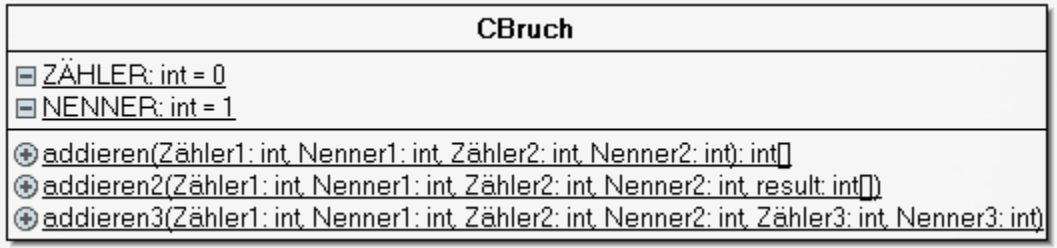
- ✓ Darstellung von Klassen **unabhängig** von der Programmiersprache
- ✓ Ist ein **Strukturdiagramm**
- ✓ Verwendung:
  - ✓ **Beschreibung eines im Code** umgesetzten Programmes
  - ✓ **Modellierung eines Sachverhalts** vor der konkreten Umsetzung in eine Programmiersprache
- ✓ Ohne Bezeichnung sind **Attribute private, Methoden public**
- ✓ Statische Methode/Attribute: Unterstreichung

Klassendiagramm II

-: private
+: public
#: protected

Klassenname(unverzichtbar)

Attribute



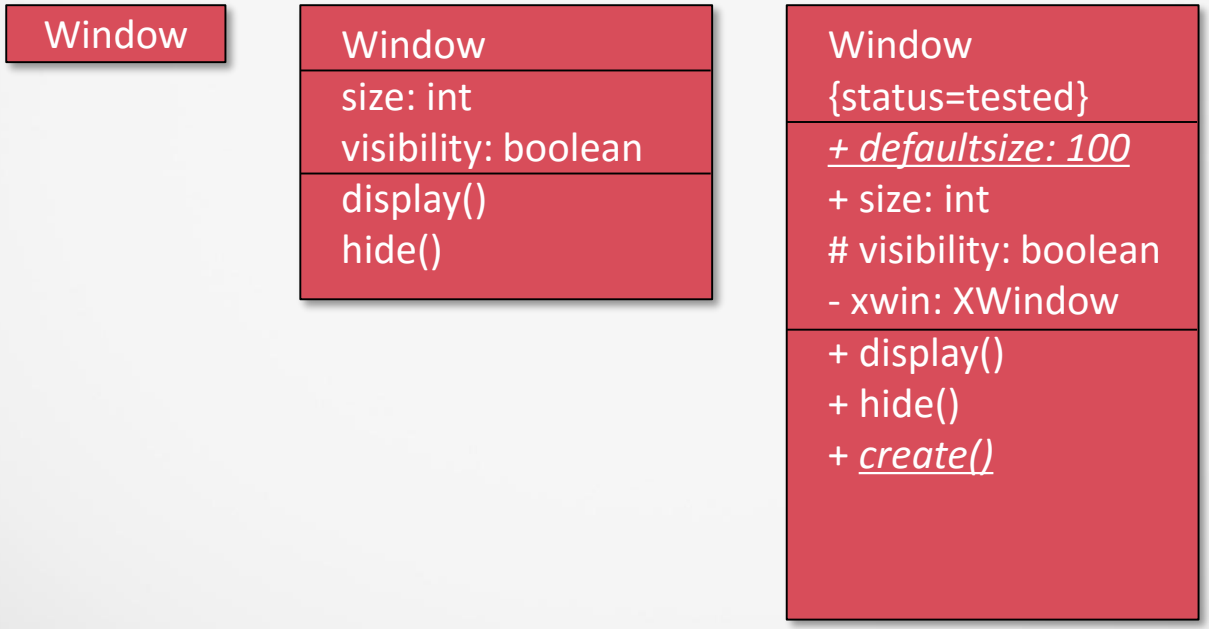
unterstreichen: static

Operationen

Klassendiagramm III

Variabler Detaillierungsgrad:

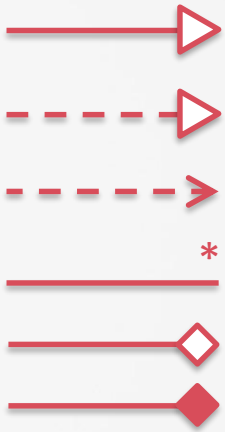
Fließender Übergang



Beziehungen in Klassendiagrammen I

Arten von Beziehungen:

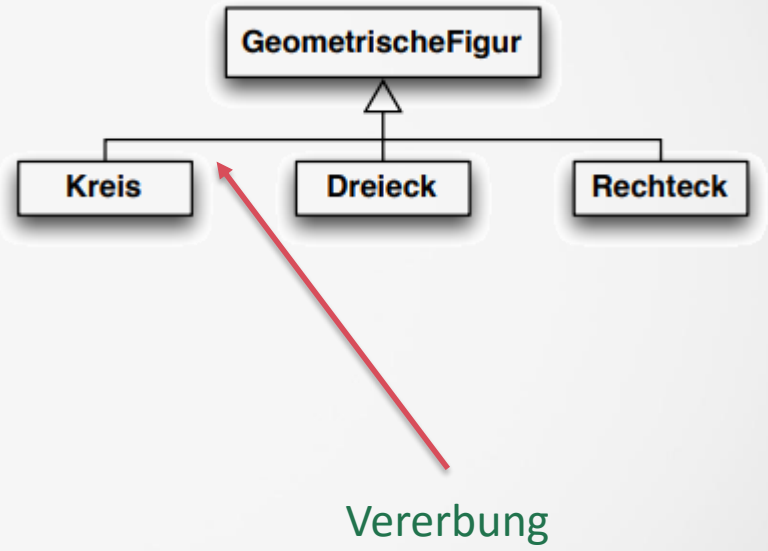
- Generalisierung(Vererbung)
- Realisation(Implementierung)
- Abhängigkeit(Objekt von Klasse)
- Assoziation
  - Spezialfall: Aggregation
  - Spezialfall: Komposition



Aggregation, wo Objekte Teile eines anderen Objekts sind!

Realisierung(Vererbung)

```
public class GeometrischeFigur {...}
public class Kreis extends GeometrischeFigur
{...}
public class Dreieck extends
GeometrischeFigur {...}
public class Rechteck extends
GeometrischeFigur {...}
```

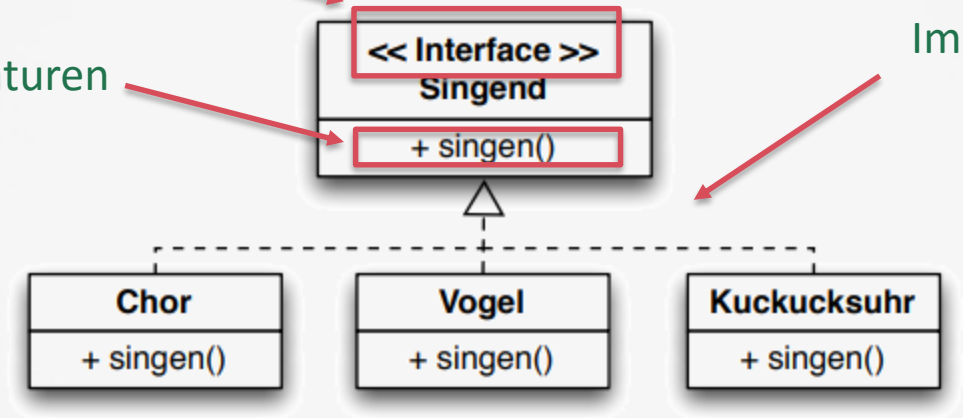


Generalisierung(Implementierung)

Stereotyp im Namensfeld

Methodensignaturen

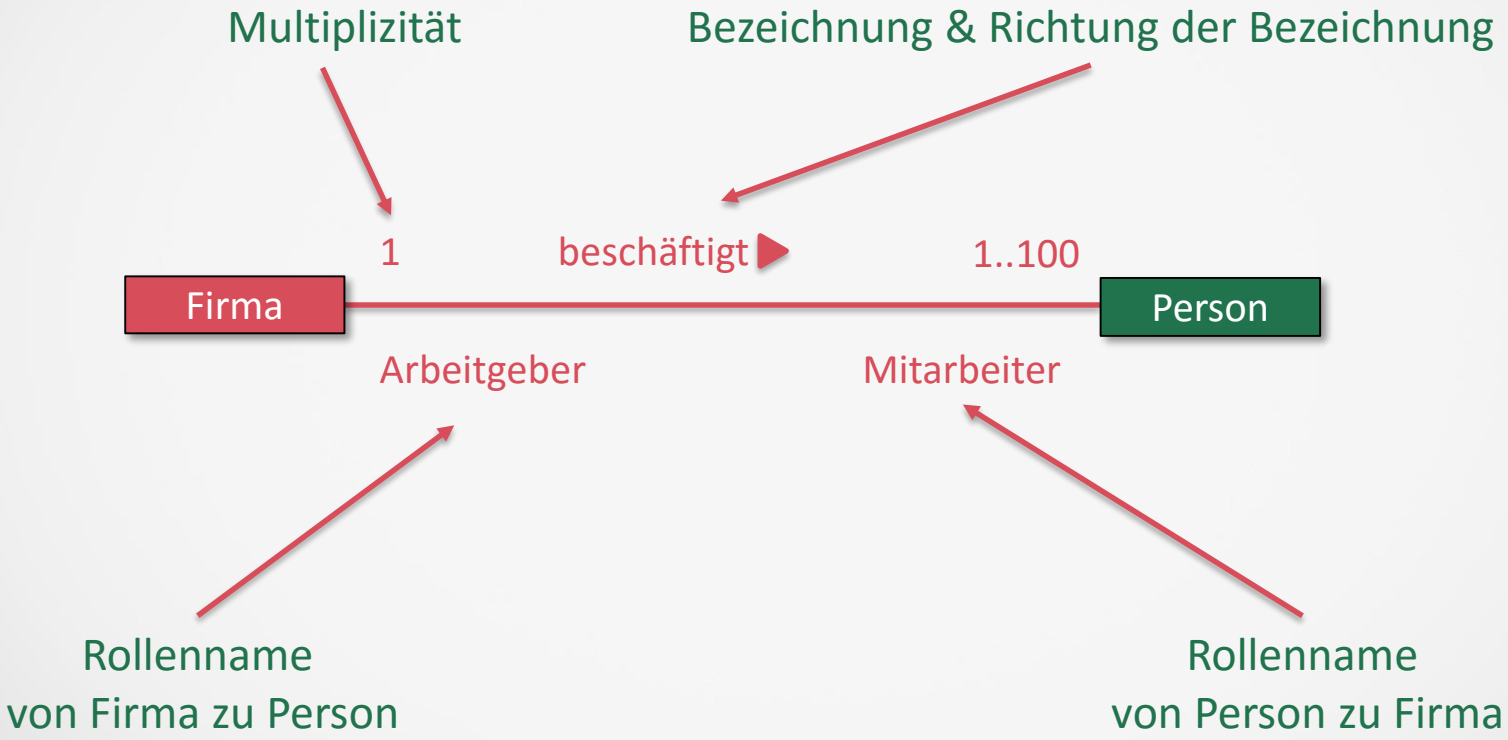
Implementierung



Assoziationen Allgemeine Informationen

- ✓ Assoziation bilden die Beziehungen zwischen Klassen ab
- ✓ Sie **können** mit Multiplizitäten versehen werden
- ✓ Eine Multiplizität gibt den Bereich der erlaubten Kardinalitäten an
- ✓ Eine Kardinalität bezeichnet die Anzahl der zulässigen Objekte
- ✓ Keine Angabe: 1
- ✓ Können durch einen Namen bezeichnet werden

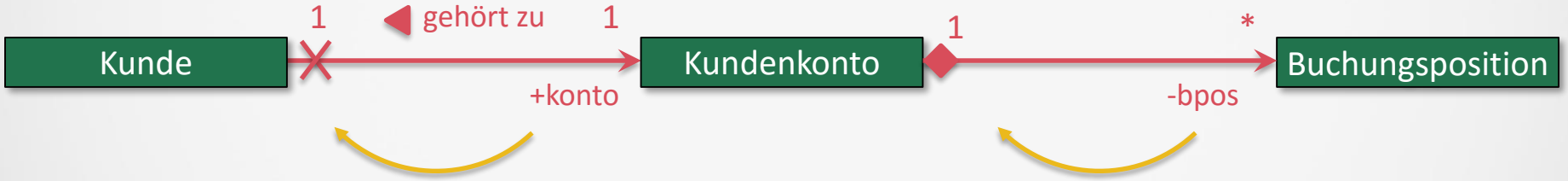
Assoziationen - Übersicht



Assoziationen - Rollennamen

- ✓ Die Rollen sind die Namen der Eigenschaften (Attribute), die der Assoziation oder einer der beteiligten Klassen gehören.
- ✓ Assoziationen werden in Programmiersprachen in der Regel dadurch realisiert, dass die beteiligten Klassen entsprechende Attribute erhalten.
- ✓ Außer Rollennamen können auch Sichtbarkeitsangaben auf jeder Seite der Assoziation angebracht werden. Ist beispielsweise ein Assoziationsende als privat (-) deklariert, so kann das Objekt selbst, d. h. die Operationen des Objekts, die Assoziation benutzen, benachbarte Klassen erhalten jedoch keinen Zugriff.

Assoziationen - Rollennamen



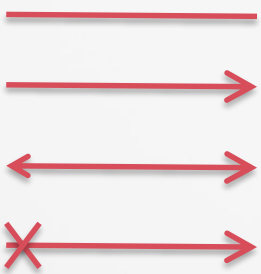
Die Klasse „Kunde“ erhält ein Attribut "konto" als Referenz auf ein Objekt der Klasse Kundenkonto

Die Klasse Kundenkonto erhält ein privates Attribut "bpos" mit einem Sammlungsobjekt (Collection bzw. Unterklasse davon), welches die Buchungsposition-Objekte referenziert

Assoziationen: Navigation

Eine Richtung

Zwei Richtungen



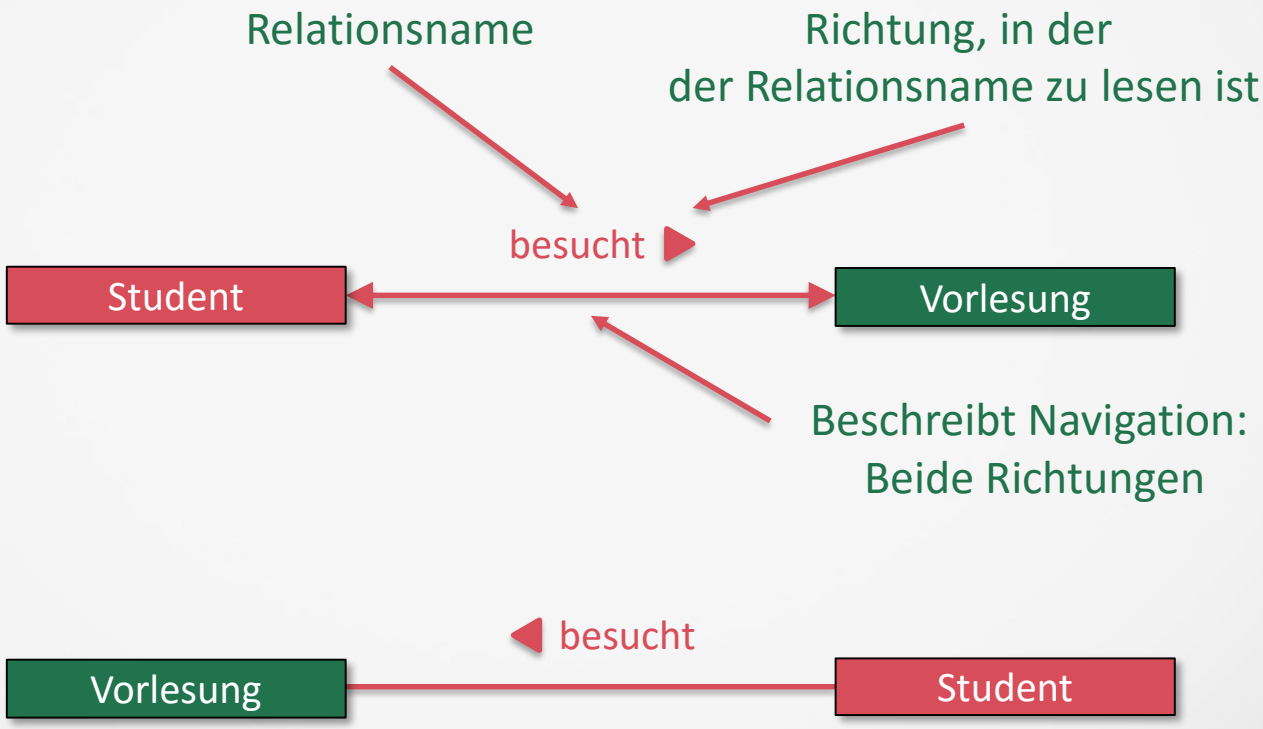
allgemein(=es gibt keine genaueren Angaben!)

nur eine Richtung

beide Richtungen

keine Navigierbarkeit von rechts nach links

Assoziationen: Bezeichungen und Bezeichnungsrichtung



Assoziationen: Multiplizitäten I

Pro Vorlesung können
0 bis 100 Studenten teilnehmen

Ein Student kann 0 oder
mehr Vorlesungen besuchen

Varianten:

*: 0 oder mehr
3: genau 3
1..*: 1 oder mehr
10..30: 10 to 30

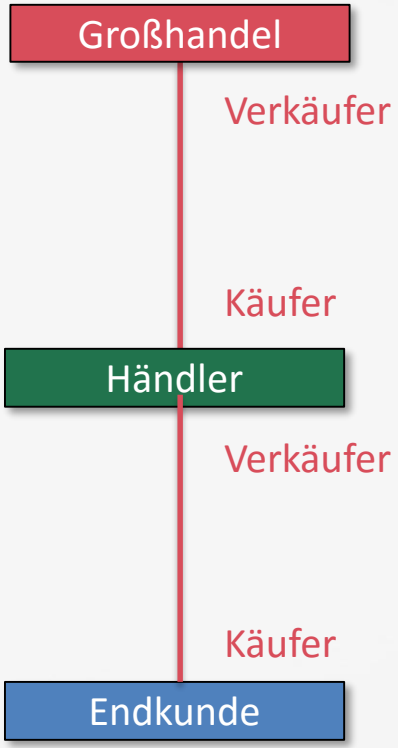
Assoziationen: Multiplizitäten II

Jede Instanz von B kann mit
x Instanzen von A assoziiert sein

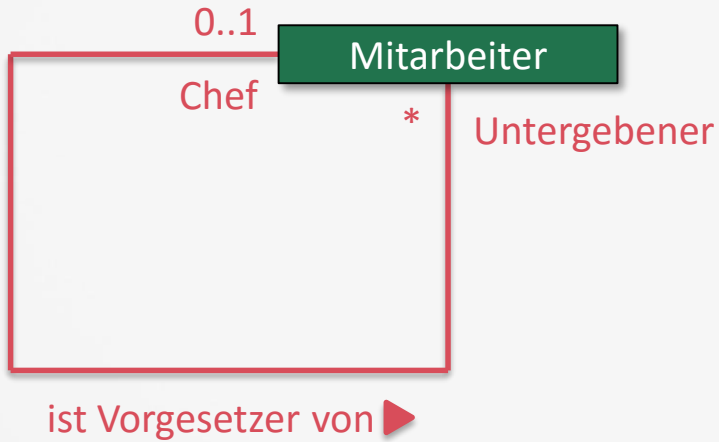
Jede Instanz von A kann mit
y Instanzen von B assoziiert sein

Die Multiplizität am jeweiligen Ende einer Assoziation gibt an, mit wie vielen Zielobjekten ein Quellobjekt in Beziehung stehen kann

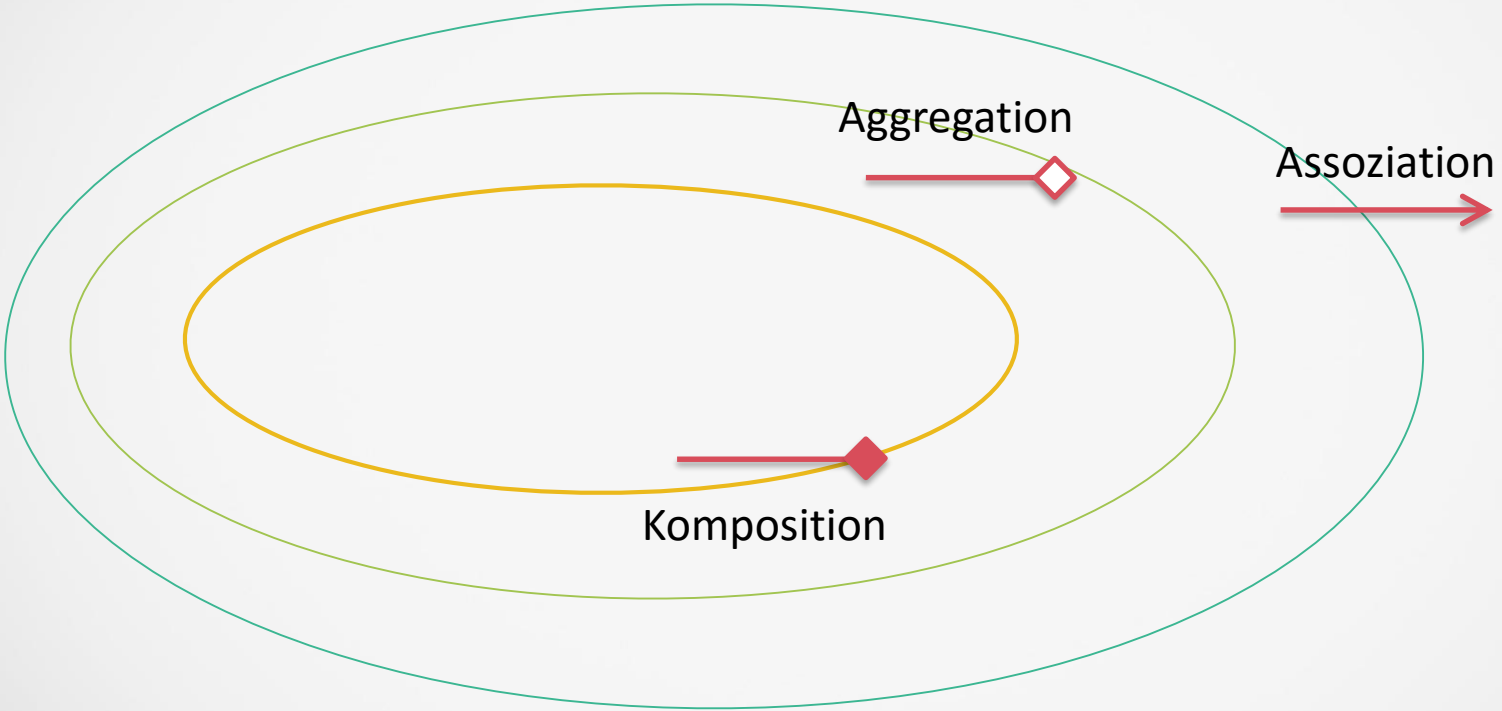
Assoziationen: Rollen



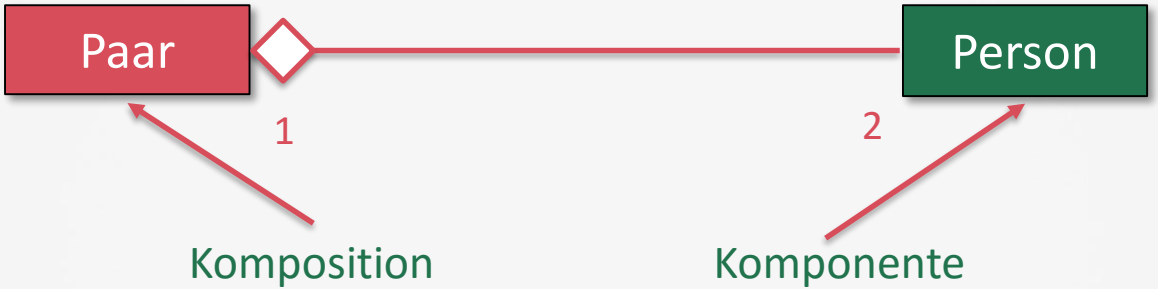
Assoziationen: ”Reflexive” Assoziationen



Beziehungen in Klassendiagrammen - Übersicht



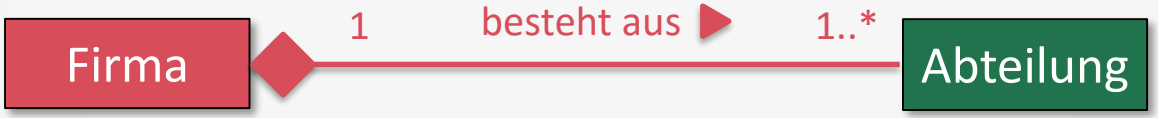
Assoziation: Spezialfall Aggregation



Objekte der Klasse Person können zu Objekten der Klasse Paar gehören,
Person-Objekte können aber auch allein existieren

Ein „Ganzes“ enthält mehrere „Teile“

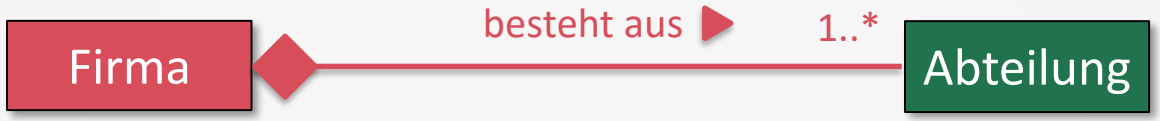
Assoziation: Spezialfall Komposition



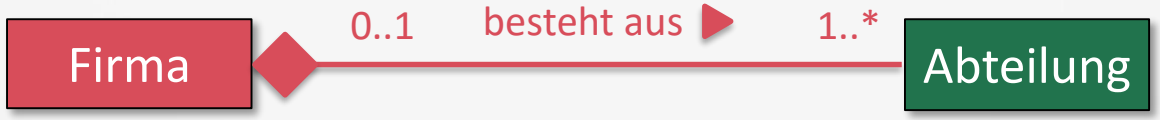
Die gefüllte Raute wird als Komposition bezeichnet und drückt aus, dass Objekte der Klasse Abteilung nur dann existieren können, wenn das/die zugehörigen Objekte der Klasse Firma existieren.

Stirbt das Objekt der Klasse Firma, müssen auch die zugehörigen Objekte der Klasse Abteilung sterben. Umgekehrt, wird ein Abteilung-Objekt gelöscht, bleibt das zugehörige Firma-Objekt davon unberührt

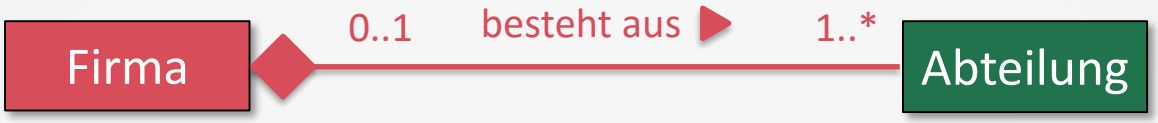
Assoziation: Spezialfall Komposition II



Keine Angabe auf der (linken) Kompositionsseite: 0..1!
Gleichwertig zu oben:



Assoziation: Spezialfall Aggregation/Komposition



Kardinalität 0 bedeutet hier:

- Teile können unabhängig von Ganzem erzeugt werden
- Sehr praxisrelevanter Fall: Vieles wir „bottom-up“ produziert

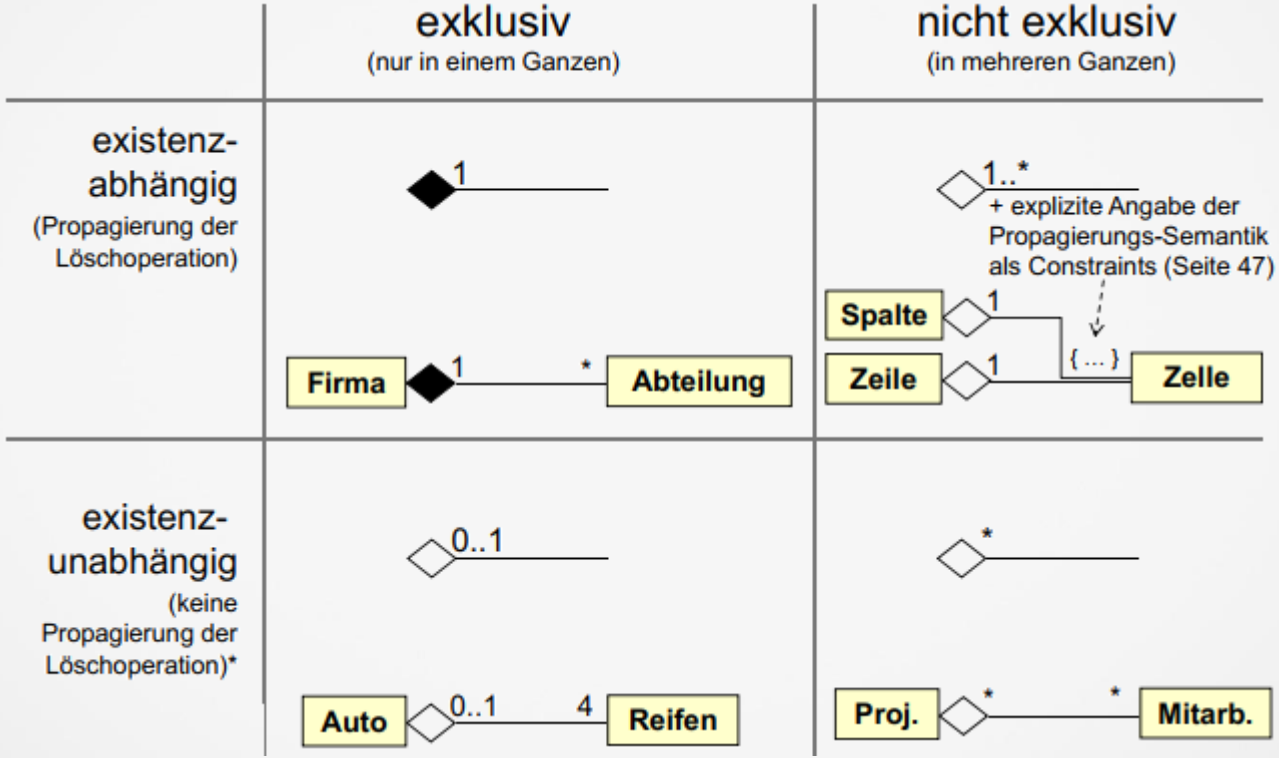
Kardinalität 1 bedeutet hier:

- Sobald die Teile einem Ganzen zugeordnet werden, sind sie exklusiv darin enthalten

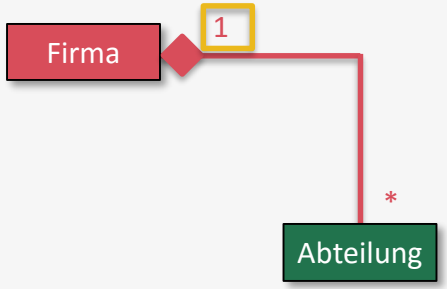
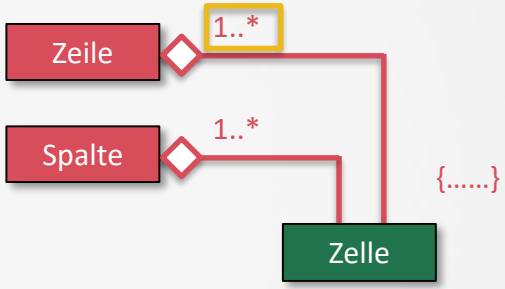
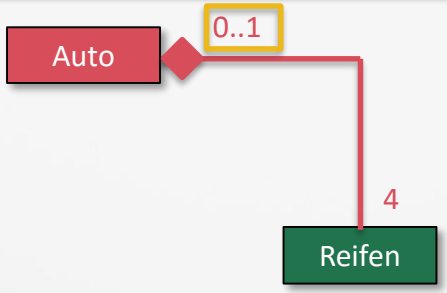
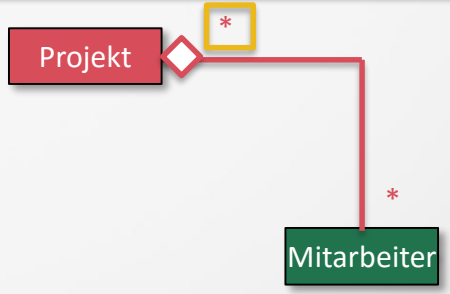
Assoziationen ” Ganzes” mit seinen “Teilen”

- ✓ Oft kaskadierende Operationen, die auf allen Teilen durchgeführt werden
  - ✓ Beispiel 1: Verschiebung eines Rechtecks verschiebt alle seine Kanten
  - ✓ Beispiel 2: Laden des Ganzen von Platte lädt alle seine Teile
- ✓ Oft Existenz-Abhängigkeit
  - ✓ Teil existiert nicht ohne Ganzem
  - ✓ Entspricht dem Kaskadierungsverhalten beim Löschen: Teile werden mit gelöscht oder an anderes Ganzes weitergegeben
- ✓ Impliziert oft Exklusivität
  - ✓ Teil ist in genau einem Ganzen enthalten
- ✓ Mögliche semantische Kategorien
  - ✓ (existenz-)abhängig, exklusiv
  - ✓ (existenz-)abhängig, nicht exklusiv
  - ✓ (existenz-)unabhängig, exklusiv
  - ✓ (existenz-)unabhängig, nicht exklusiv

Assoziationen: Die vier Kategorien

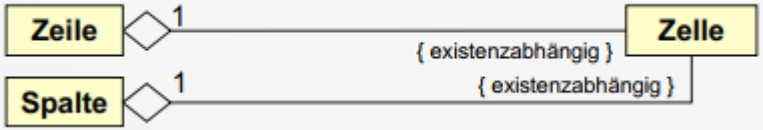


Assoziationen: Die vier Kategorien

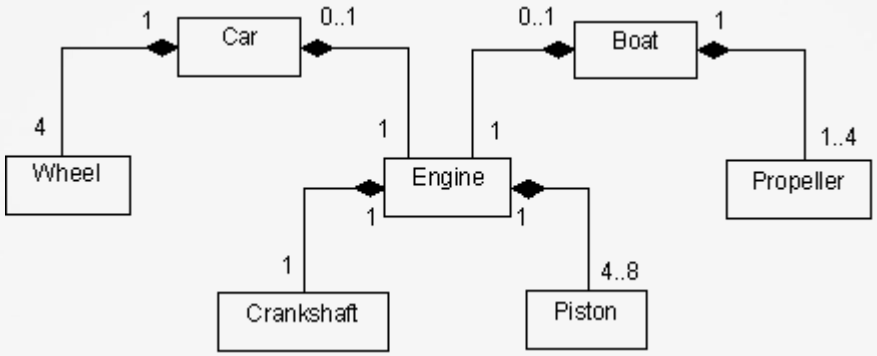
	exklusiv (nur in einem Ganzen)	nicht exklusiv (in mehreren Ganzen)
existenzabhängig Wird das Ganze gelöscht, werden *auch* die Teile gelöscht		
existenzunabhängig: Wird das Ganze gelöscht, werden *nicht* die Teile gelöscht		

Assoziationen Allgemeine Informationen

- ✓ Bisher:
  - ✓ Spezialnotationen (z.B. „Komposition“)
  - ✓ Kardinalitäten (z.B. 1..4)
- ✓ Nötige Erweiterung: Notation für beliebige Bedingungen
  - ✓ Angabe von Bedingung in geschweiften Klammern: { Bedingung }
  - ✓ vordefinierte Eigenschaften: abstract, readOnly, query, leaf, ordered, …
  - ✓ beliebige umgangssprachliche Formulierung

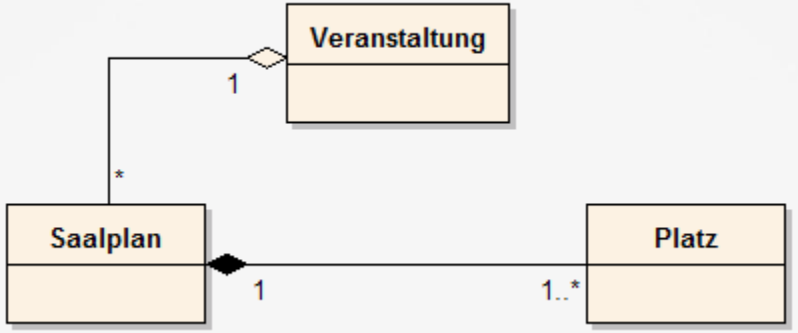


Assoziation: Spezialfall Komposition III



http://www.jot.fm/issues/issue_2004_11/column5/

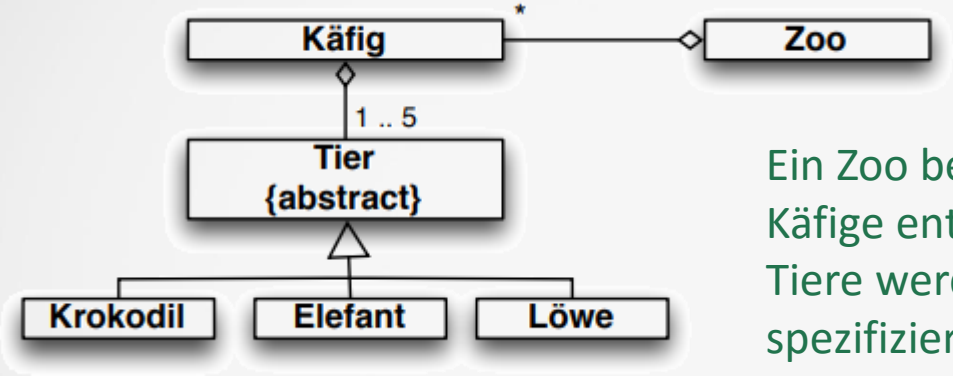
Assoziation: Beispiel Komposition/Aggregation



?

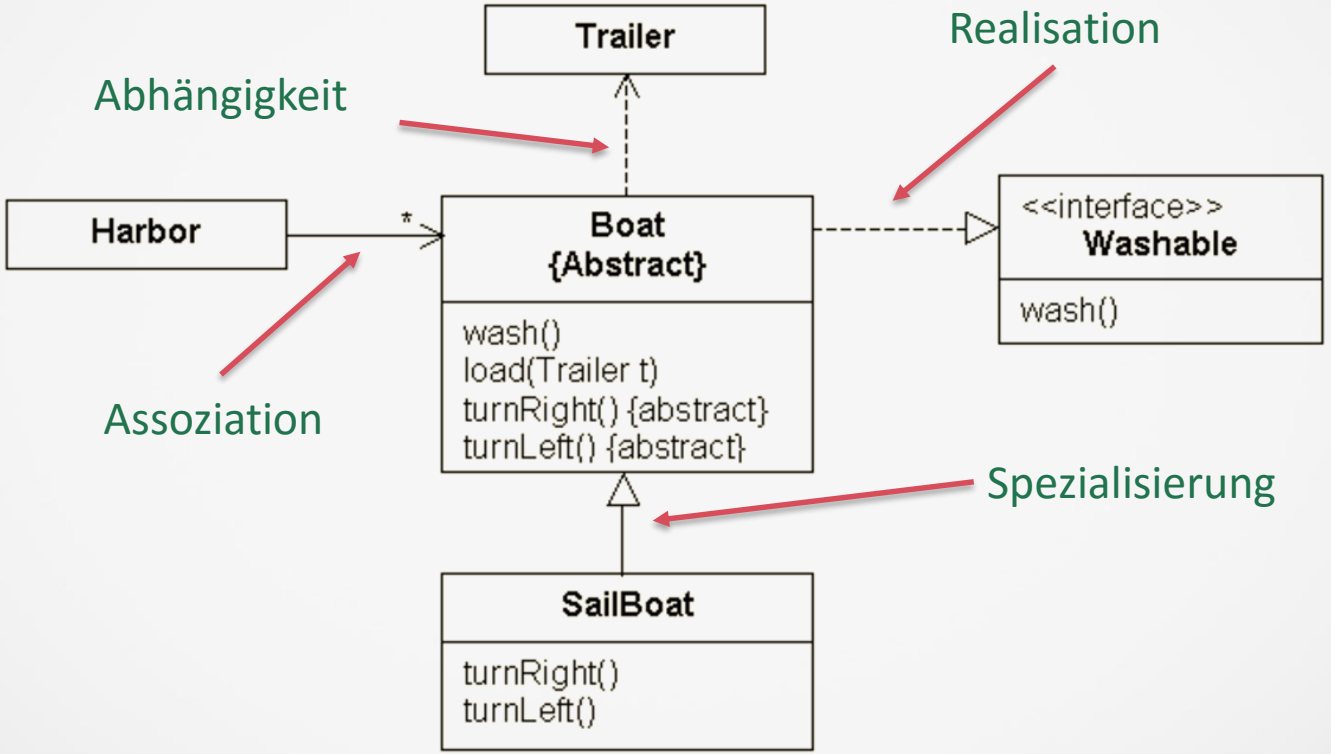
- Ein Saalplan hat beliebig viele Plätze, aber mindestens einen.
- Ein Platz gehört genau zu einem Saalplan.
- Jeder Saalplan gehört zu einer Veranstaltung.
- Eine Veranstaltung kann beliebig viele Saalpläne enthalten.
- Der Saalplan könnte auch zu einer anderen Veranstaltung zugeordnet werden (Aggregation), muss allerdings immer eine Veranstaltung haben.
- Der Platz gehört zu einem Saalplan, diese Beziehung kann nicht geändert werden (Komposition).

Beispiele I - Zoo

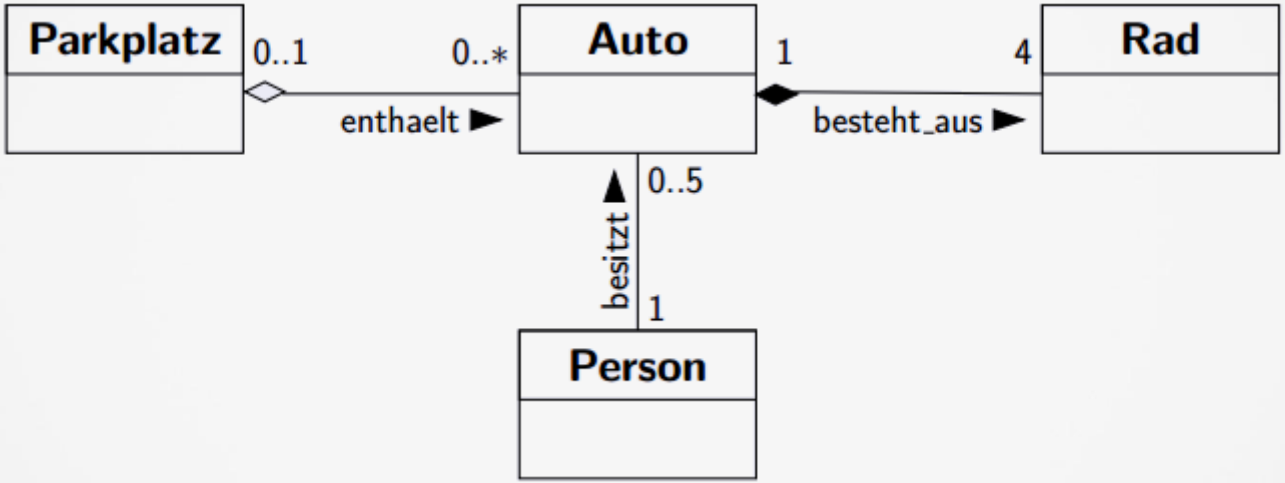


Ein Zoo besteht aus mehreren Käfigen.
Käfige enthalten Tiere.
Tiere werden in Unterklassen genauer spezifiziert.
Die Klasse Tier ist abstrakt, von ihr dürfen keine Objekte erzeugt werden.

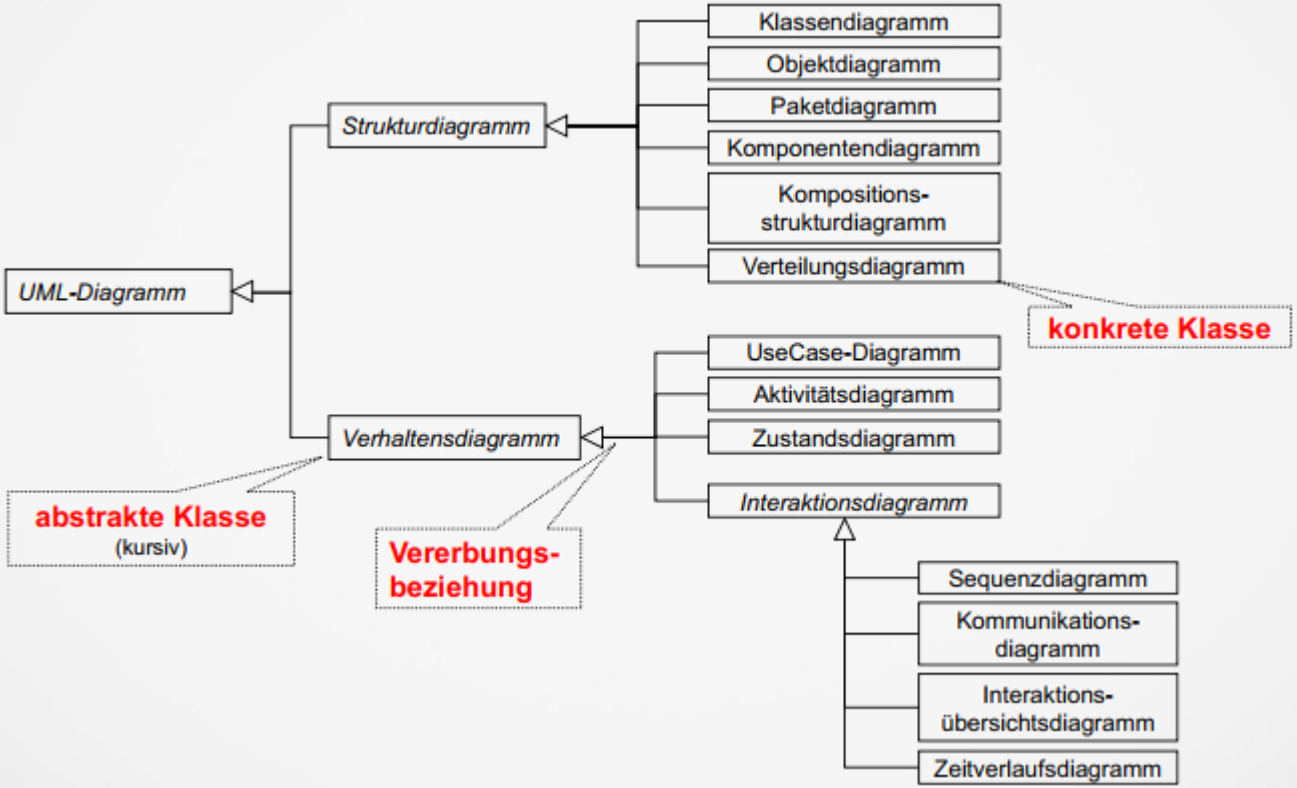
Beispiele II - Boat



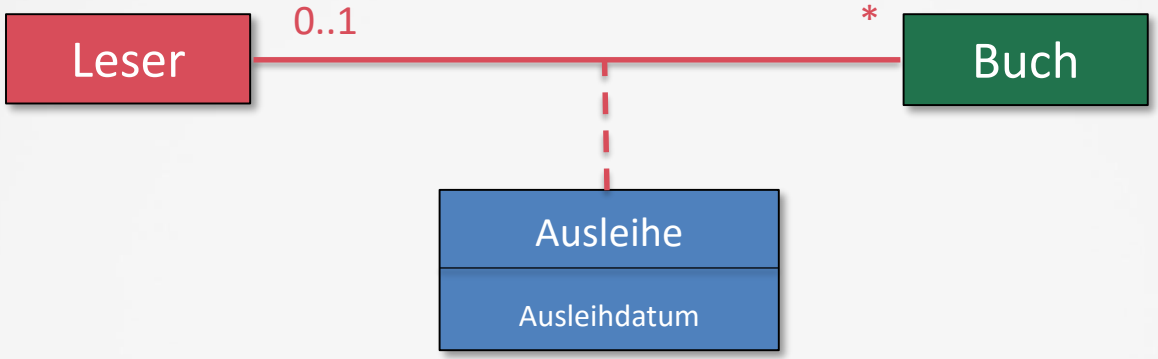
Beispiele III - Auto



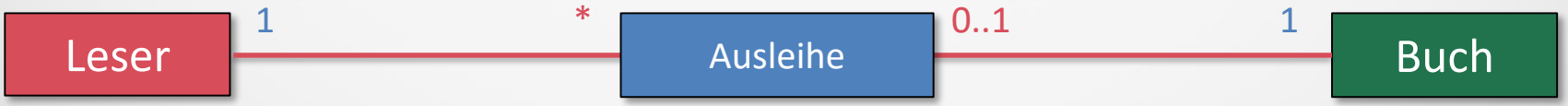
UML Diagrammtypen



Assoziationsklassen I

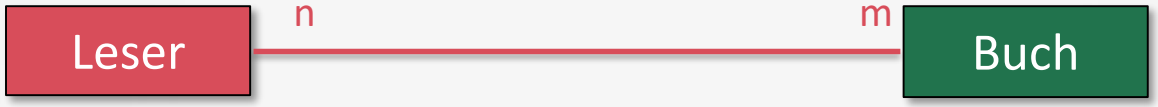


Wenn es zu einer Ausleihe eines Buches kommt, wird der Ausleihe ein Ausleihdatum zugeordnet! Für eine Kodierung muss die Assoziationsklasse aufgelöst werden:



Assoziationsklassen II

Begründungen der „1-en“: Bei Datenbanken werden n:m immer zu 1:n – Relationen aufgelöst(1:m ist das gleiche wie 1:n!)



In der Tabelle Ausleihe werden alle Daten vom Ausleihvorgang gespeichert!