BERUFSFÖRDERUNGSWERK
Berlin Brandenburg e. V.

UML - Klassendiagramme

Berufsförderungswerk Berlin-Brandenburg e. V.
2017

Dipl. Kfm. Patrick Lange (FH)

BERUFSFÖRDERUNGSWERK
Berlin Brandenburg e. V.

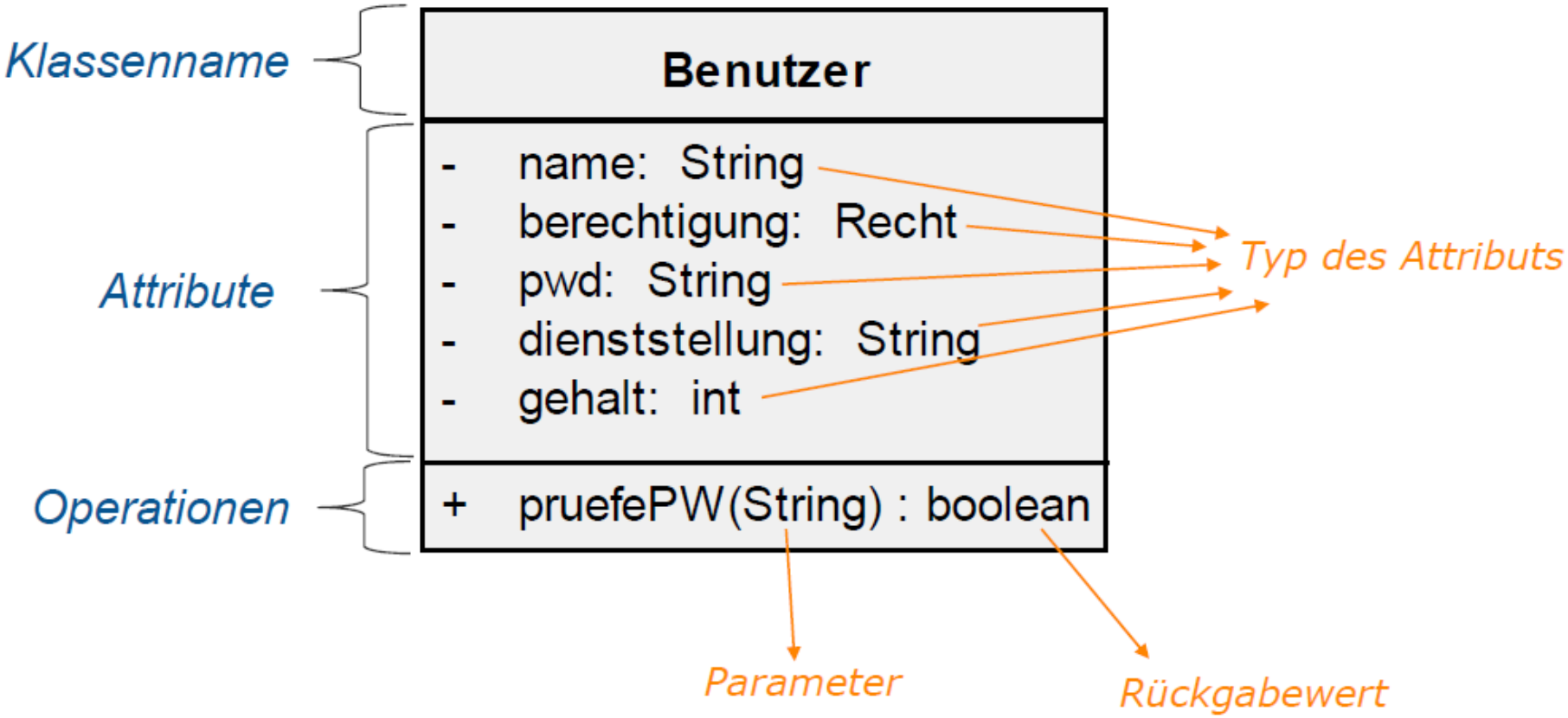
Berufsförderungswerk Berlin-Brandenburg e. V.

Klassen

Zweck

- Beschreibt den **strukturellen Aspekt** eines Systems auf **Typebene** in Form von Klassen, Interfaces und Beziehungen

Notation


Klassenname
Attribute
Operationen
Benutzer
- name: String
- berechtigung: Recht
- pwd: String
- dienststellung: String
- gehalt: int
+ pruefePW(String) : boolean
Typ des Attributs
Parameter
Rückgabewert

Attribute und Operationen

Sichtbarkeiten von Attributen und Operationen:

+ ... public

- ... private

... protected

~ ... package (vgl. Java)

Eigenschaften von Attributen:

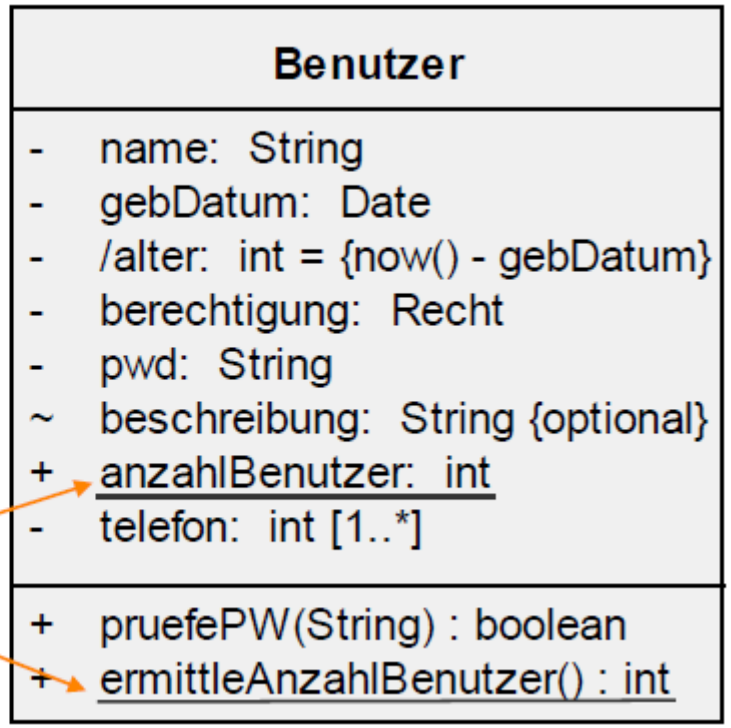
„ /“ attributname: abgeleitetes Attribut

Bsp.: /alter:int

{optional}: Nullwerte sind erlaubt

[n..m]: Multiplizität

Klassenattribute/-operationen



Identifikation von Klassen

- **Hauptwörter** herausfiltern
  - Eliminierung von irrelevanten Begriffen
  - Entfernen von Namen von Ausprägungen
  - Beseitigung vager Begriffe
  - Identifikation von Attributen
  - Identifikation von Operationen
  - Eliminierung von Begriffen, die zu Beziehungen aufgelöst werden können

Identifikation von Attributen

- **Adjektive** und **Mittelwörter** herausfiltern
  - Attribute beschreiben Objekte und sollten weder klassenwertig noch mehrwertig sein
  - abgeleitete Attribute sollten als solche gekennzeichnet werden
  - kontextabhängige Attribute sollten eher Assoziationen zugeordnet werden als Klassen
  - Attribute sind i.A. nur unvollständig in der Anforderungsbeschreibung definiert

Identifikation von Operationen

- **Verben** herausfiltern
  - Welche Operationen kann man mit einem Objekt ausführen?
  - Nicht nur momentane Anforderungen berücksichtigen, sondern Wiederverwendbarkeit im Auge behalten
  - Welche Ereignisse können eintreten?
  - Welche Objekte können auf diese Ereignisse reagieren?
  - Welche anderen Ereignisse werden dadurch ausgelöst?

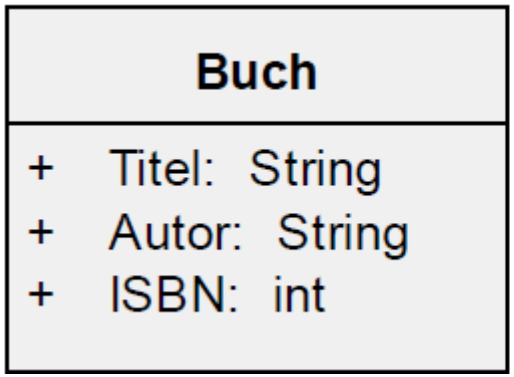
Beispielaufgabe

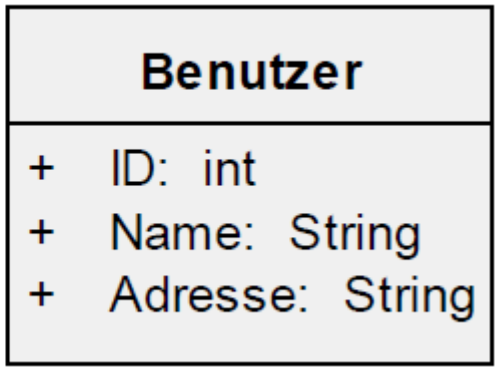
Franz Müller soll neben anderen Leuten die Bibliothek der Universität benutzen können. Im Verwaltungs-system werden die Benutzer erfasst, von denen eine eindeutige ID, Name und Adresse bekannt sind, und die Bücher, von denen Titel, Autor und ISBN-Nummer gespeichert sind.

Welche Klassen, Attribute oder Operationen sind erkennbar?

Beispielaufgabe

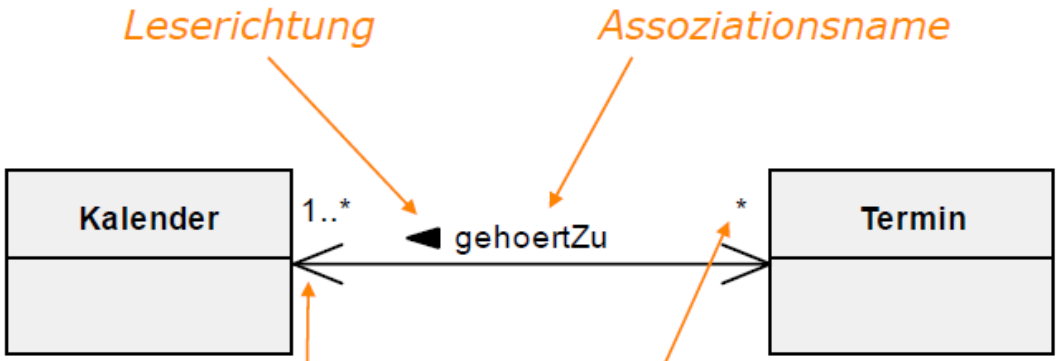
Franz Müller soll neben anderen Leuten die Bibliothek der Universität benutzen können. Im Verwaltungssystem werden die **Benutzer** erfasst, von denen eine **eindeutige ID**, **Name** und **Adresse** bekannt sind, und die **Bücher**, von denen **Titel**, **Autor** und **ISBN-Nummer** gespeichert sind.





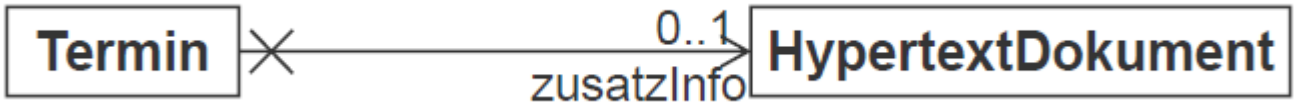
Assoziation

- Assoziationen zwischen Klassen modellieren mögliche **Objektbeziehungen** (*Links*) zwischen den **Instanzen der Klassen**



Assoziation: Navigationsrichtung

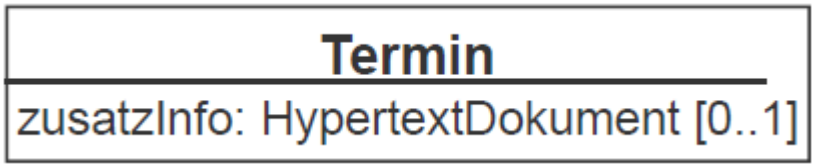
- Eine gerichtete Kante gibt an**, in welche Richtung die Navigation** von einem Objekt zu seinem Partnerobjekt **erfolgen kann**
- Ein **nicht-navigierbares Assoziationsende** wird durch ein "X" am Assoziationsende angezeigt



  - Navigation von einem bestimmten Termin zum entsprechenden Dokument
  - Umgekehrte Richtung - welche Termine beziehen sich auf ein bestimmtes Dokument? - wird nicht unterstützt
- **Ungerichtete Kanten** bedeuten "**keine Angabe** über Navigationsmöglichkeiten"
  - In Praxis wird oft bidirektionale Navigierbarkeit angenommen
- Die Angabe von Navigationsrichtungen stellt einen Hinweis für die spätere Entwicklung dar

Assoziation als Attribut

- Ein **navigierbares Assoziationsende** hat die gleiche Semantik wie ein Attribut der Klasse am gegenüberliegenden Assoziationsende
- Ein navigierbares Assoziationsende kann daher **anstatt** mit einer **gerichteten Kante** auch als **Attribut** modelliert werden
  - Die mit dem Assoziationsende verbundene Klasse muss dem **Typ** des Attributs entsprechen
  - Die **Multiplizitäten** müssen gleich sein
- Für ein navigierbares Assoziationsende sind somit alle Eigenschaften und Notationen von Attributen anwendbar

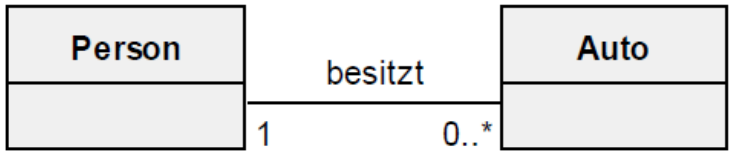


Assoziation: Multiplizität

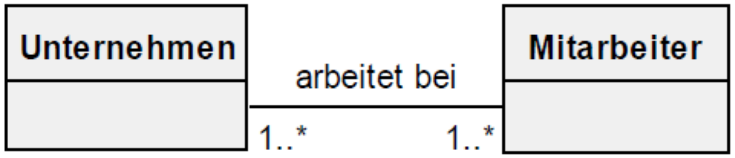
- **Bereich**: "min .. max"
- Beliebige **Anzahl**: "*" (= 0.. *)
- **Aufzählung** möglicher Kardinalitäten (durch Kommata getrennt)
- **Defaultwert**: 1

genau 1:	1
>= 0:	* *oder* 0..*
0 oder 1:	0..1 *oder* 0,1
fixe Anzahl (z.B. 3):	3
Bereich (z.B. >= 3):	3..*
Bereich (z.B. 3 - 6):	3..6
Aufzählung:	3,6,7,8,9 *oder* 3, 6..9

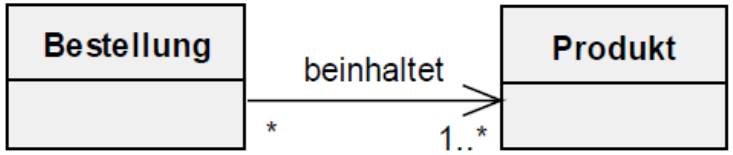
Assoziation



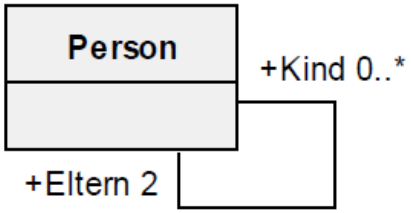
Ein Auto hat genau einen Besitzer, eine Person kann aber mehrere Autos besitzen (oder keines).



In einem Unternehmen arbeitet mind. ein Mitarbeiter, ein Mitarbeiter arbeitet mind. in einem Unternehmen



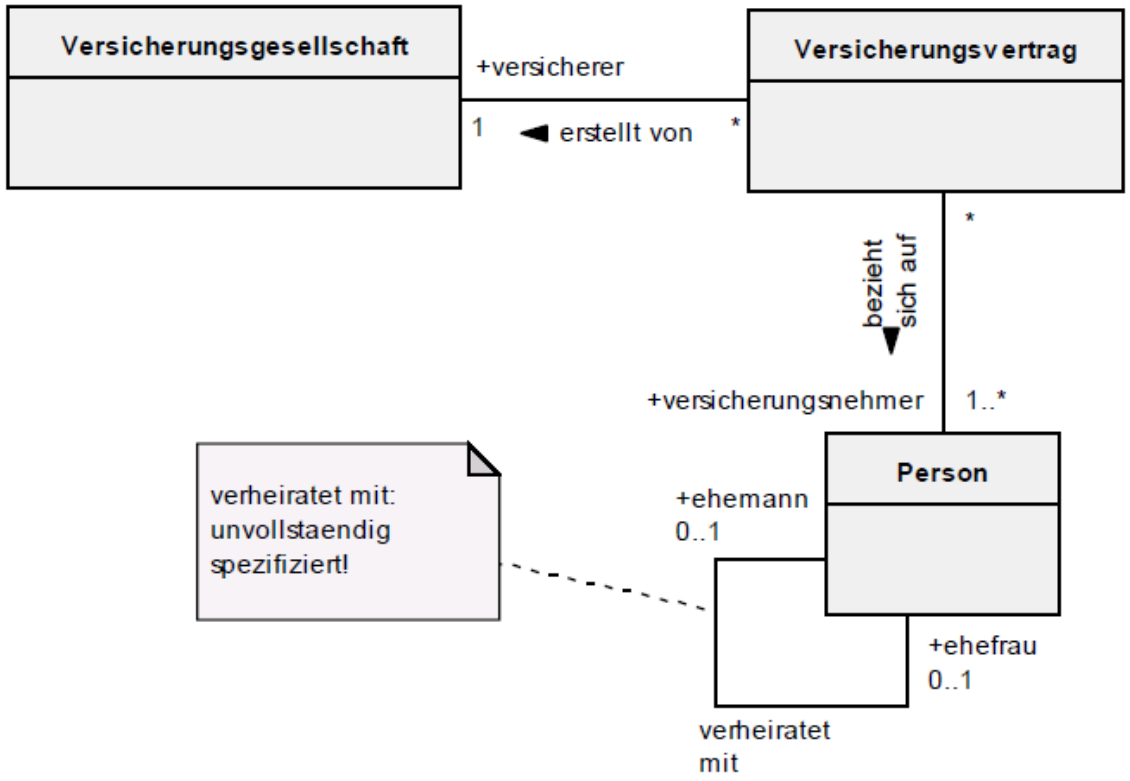
Eine Bestellung besteht aus 1-n Produkten, Produkte können beliebig oft bestellt werden. Von einer Bestellung kann festgestellt werden, welche Produkte sie beinhaltet.



Eine Person hat 2 Eltern, die Personen sind, und 0 bis beliebig viele Kinder.
Ist durch dieses Modell ausgeschlossen, dass eine Person Kind von sich selbst ist?

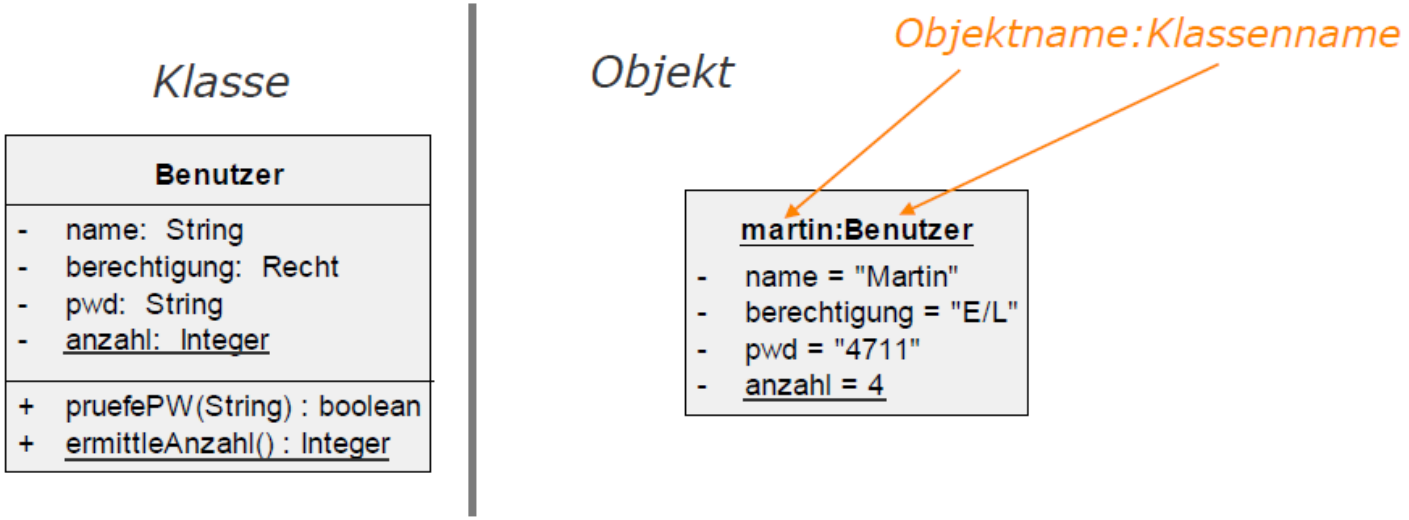
Assoziation bei Rollen

- Es können die **Rollen** festgelegt werden, die von den einzelnen Objekten in den Objektbeziehungen gespielt werden



Objektdiagramm

- Beschreibt den strukturellen Aspekt eines Systems auf Instanzebene in Form von Objekten und Links
- Momentaufnahme (snapshot) des Systems – konkretes Szenario
- Ausprägung zu einem Klassendiagramm
- Eigentlich eine »Instanzspezifikation«
- Prinzipiell kann jede Diagrammart auf Instanzebene modelliert werden

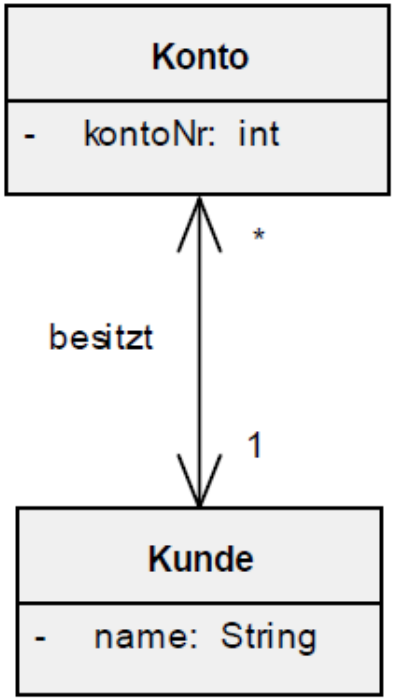


Konzept Objektdiagramm

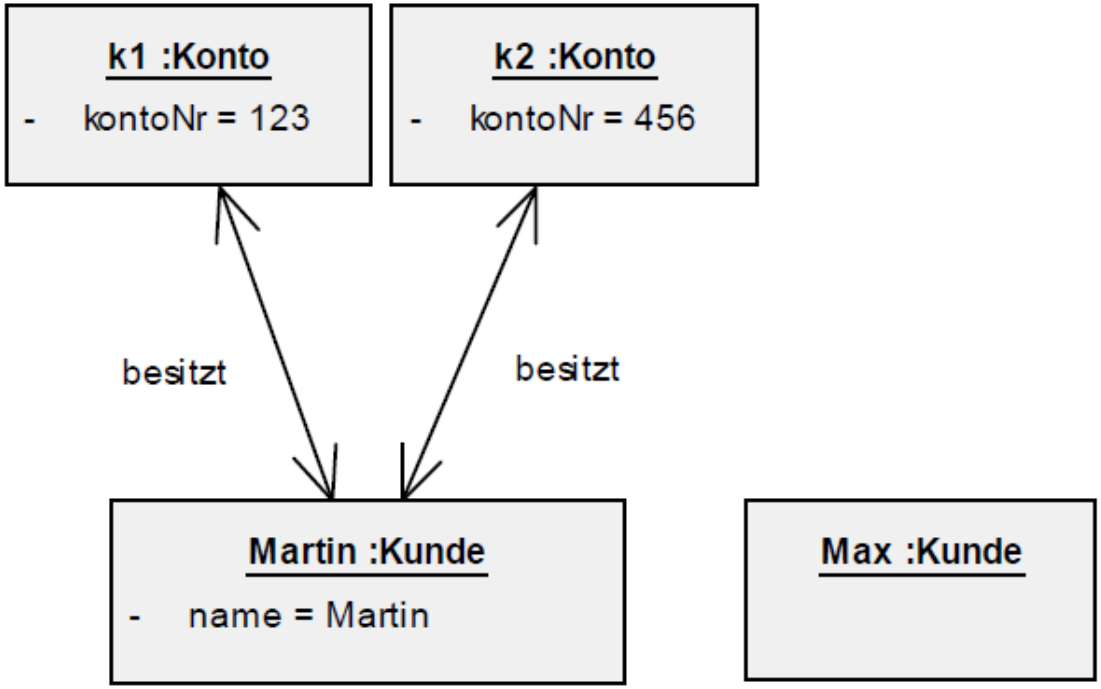
- **Basiskonzepte** des Objektdiagramms
  - Instanz einer Klasse: **Objekt**
  - Instanz einer Assoziation: **Link**
  - Instanz eines Datentyps: **Wert**
- **Einheitliche Notationskonventionen**
  - Gleiches Notationselement wie auf Typebene benutzen
  - Unterstreichen (bei Links optional)
- Objektdiagramm muss **nicht vollständig** sein
  - z.B. können Werte benötigter Attribute fehlen, aber auch Instanzspezifikation abstrakter Klassen modelliert werden

Beispiel

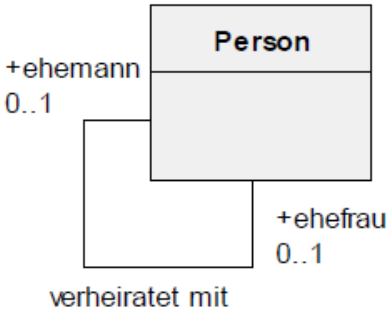
Klassendiagramm

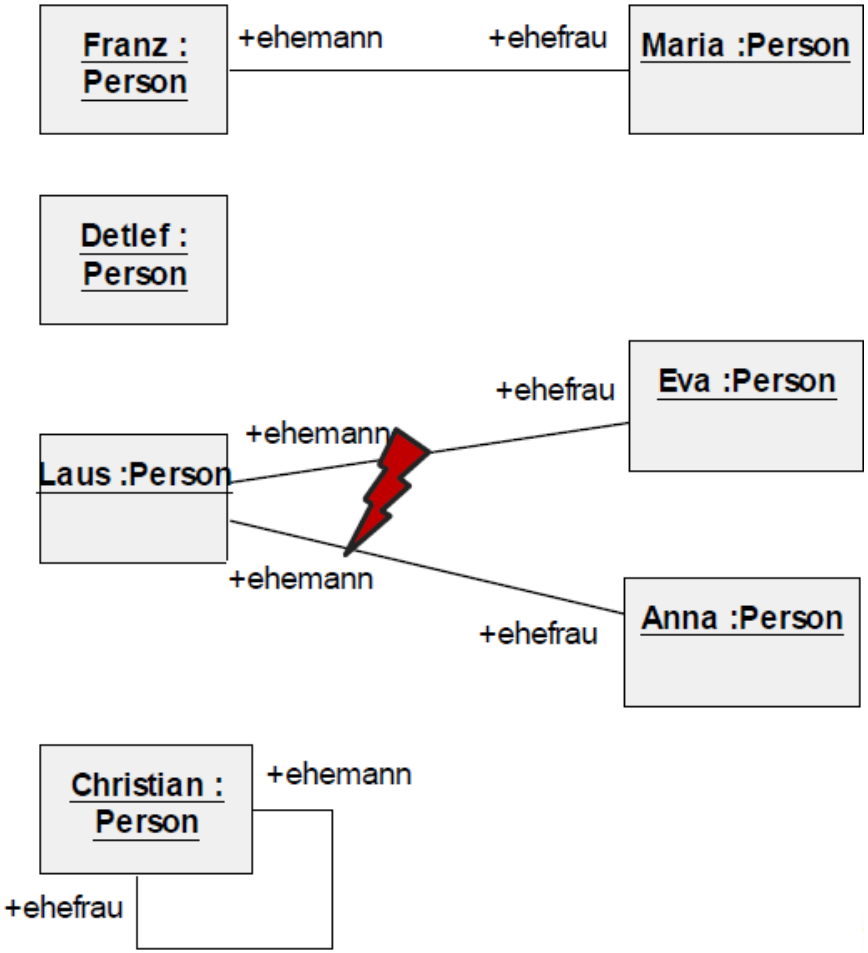


Objektdiagramm



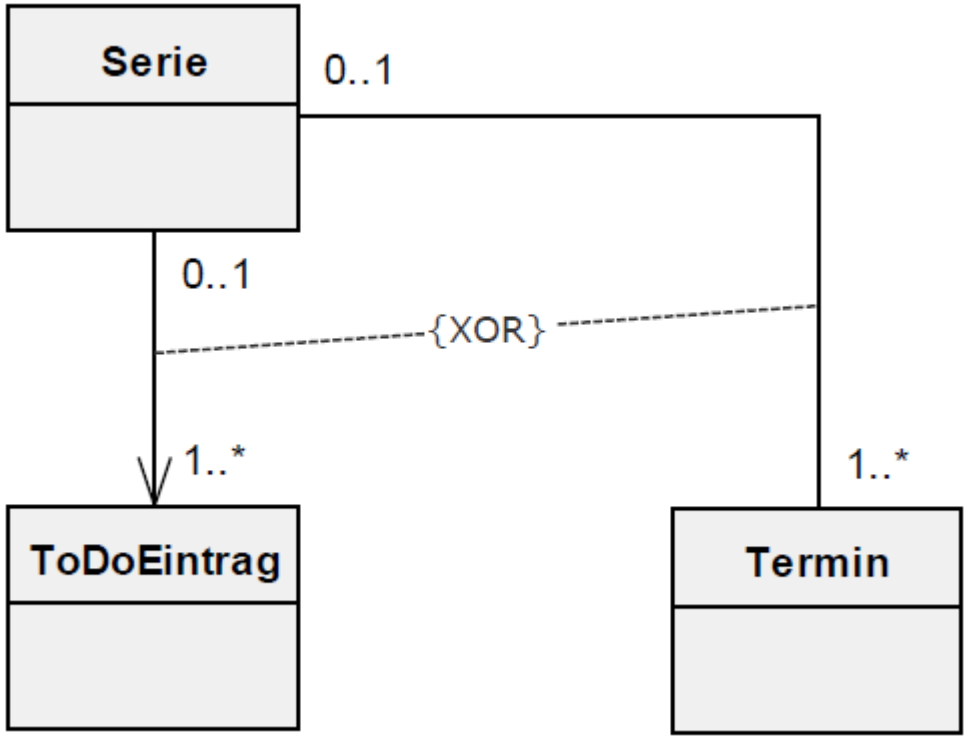
Objektdiagramm: Beispiel bei unärer Assoziation





Exklusive Assoziation

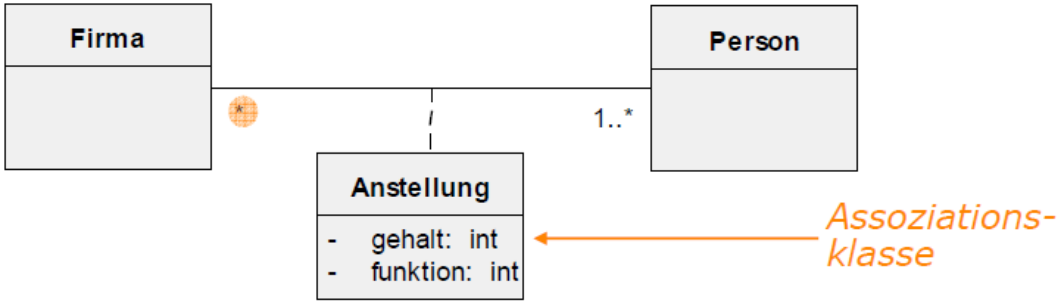
- Für ein bestimmtes Objekt kann zu einem bestimmten Zeitpunkt **nur eine von verschiedenen möglichen Assoziationen** instanziert werden: **{xor}**



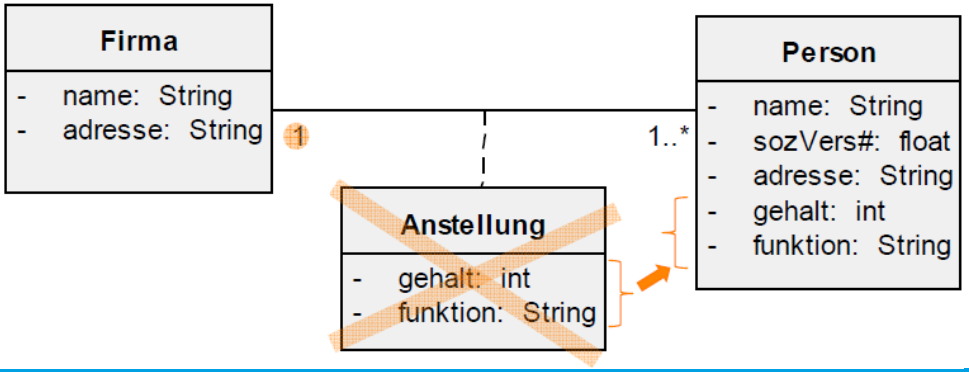
Assoziationsklasse

Kann **Attribute der Assoziation** enthalten

- Bei m:n-Assoziationen mit Attributen notwendig

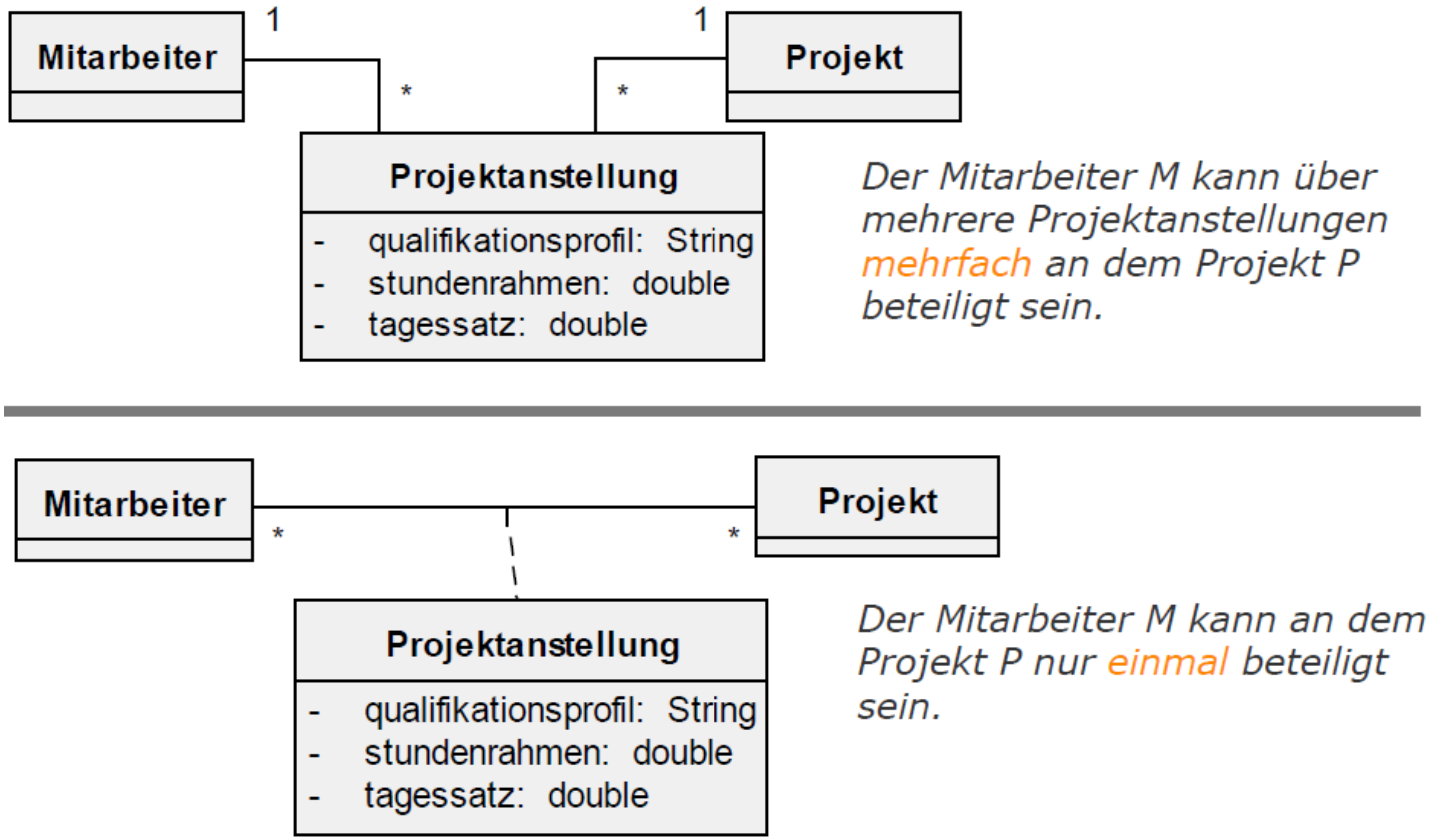


- Bei 1:1 und 1:n-Assoziationen sinnvoll aus Flexibilitätsgründen (Änderung der Multiplizität möglich)



Assoziationsklasse

- Normale Klasse ungleich Assoziationsklasse



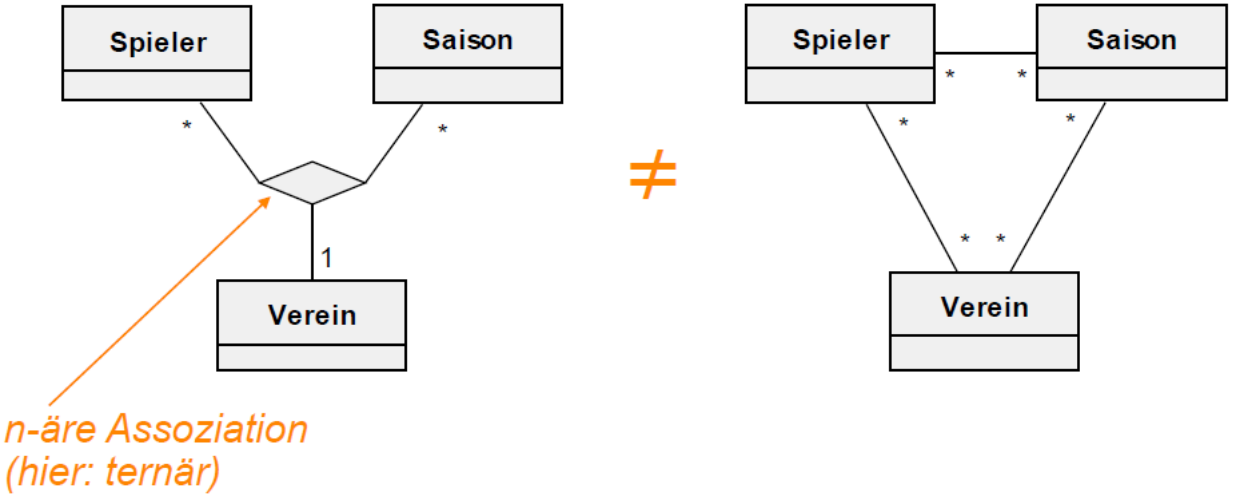
n-äre Assoziation

- **Beziehung zwischen mehr als zwei Klassen**
  - Navigationsrichtung kann nicht angegeben werden
  - Multiplizitäten geben an, wie viele Objekte einer Rolle/Klasse einem festen (n-1)-Tupel von Objekten der anderen Rollen/Klassen zugeordnet sein können
- **Multiplizitäten implizieren Einschränkungen**, in einem bestimmten Fall funktionale Abhängigkeiten
- Liegen (n-1) Klassen (z.B. A und B) einer Klasse (z.B. C) mit Multiplizität von max. 1 gegenüber, so existiert eine funktionale Abhängigkeit (A, B) → (C)

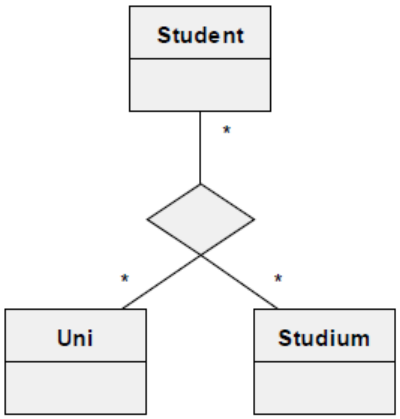
n-äre Assoziation

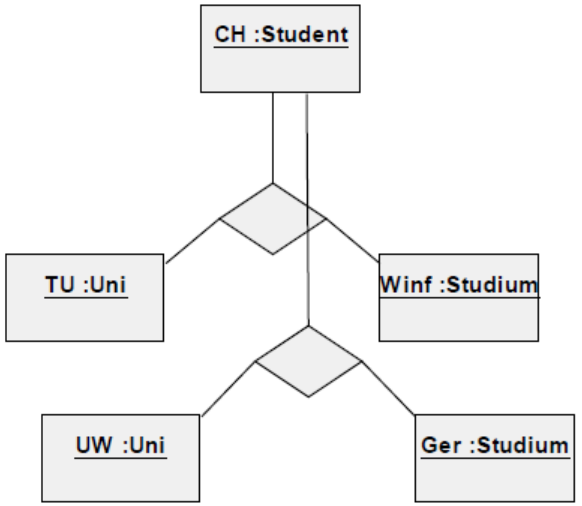
Beispiel

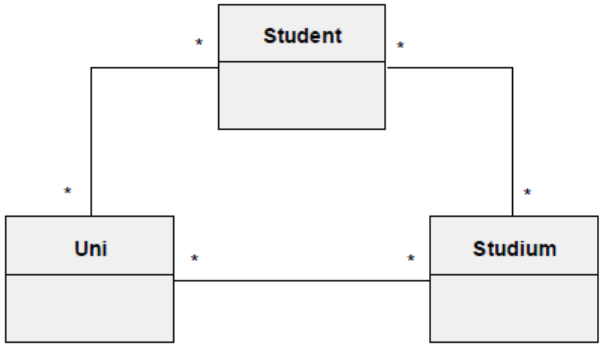
- (Spieler, Saison) → (Verein)
  - Ein Spieler spielt in einer Saison bei genau **einem** Verein
- (Saison, Verein) → (Spieler)
  - In einer Saison spielen bei einem Verein **mehrere** Spieler
- (Verein, Spieler) → (Saison)
  - Ein Spieler spielt in einem Verein in **mehreren** Saisonen

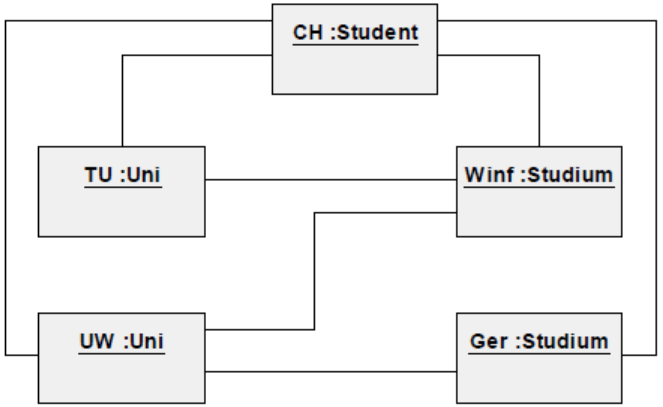


n-äre Assoziation









Aggregation

- Aggregation ist eine spezielle Form der Assoziation mit folgenden Eigenschaften:
  - Transitivität:

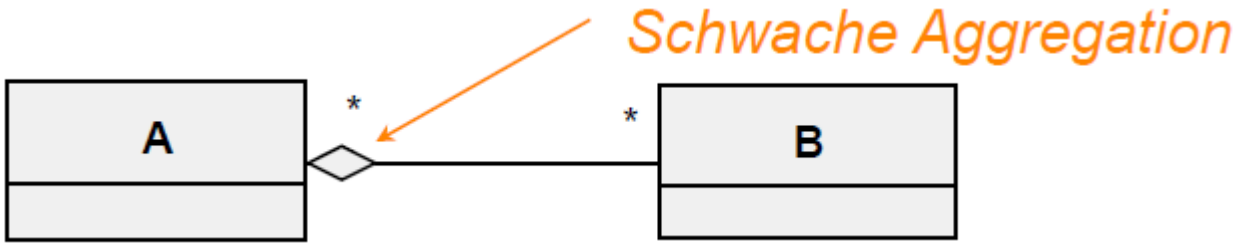
    C ist Teil von B u. B ist Teil von A C ist Teil von A
    - Bsp.: Kühlung = Teil von Motor & Motor = Teil von Auto

      → Kühlung ist (indirekter) Teil von Auto
  - Anti-Symmetrie:

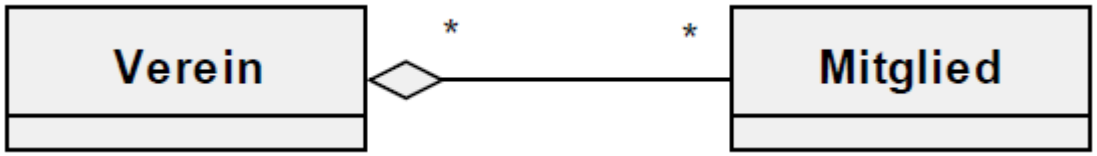
    B ist Teil von A → A ist nicht Teil von B
    - Bsp.: Motor ist Teil von Auto, Auto ist nicht Teil von Motor

- UML unterscheidet zwei Arten von Aggregationen:
  - Schwache Aggregation (shared aggregation)
  - Starke Aggregation – Komposition (composite aggregation)

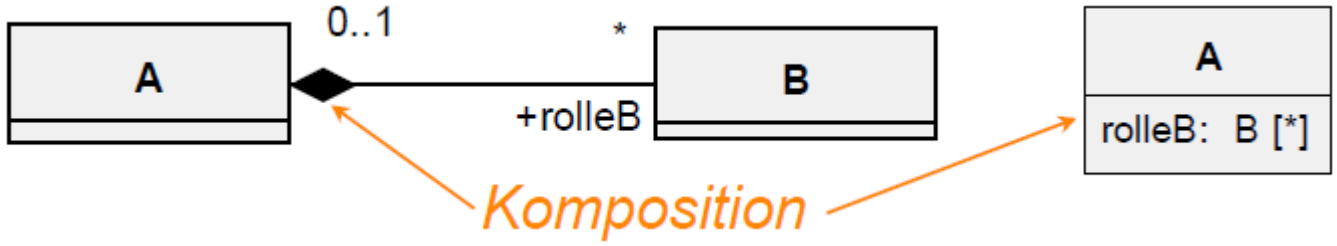
Schwache Aggregation



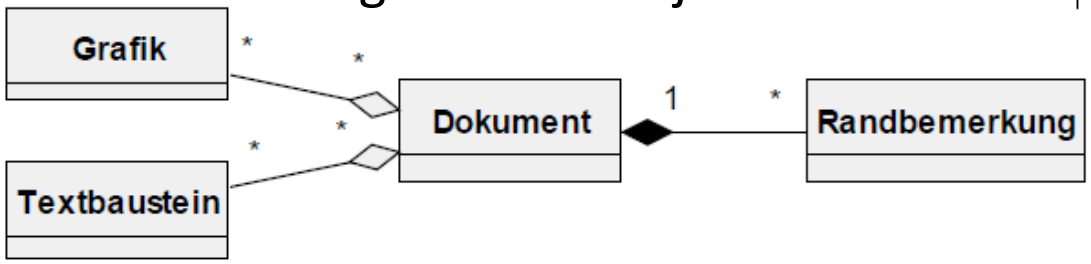
- Schwache Zugehörigkeit der Teile, d.h. Teile sind **unabhängig** von ihrem Ganzen
- Die Multiplizität des aggregierenden Endes der Beziehung (Raute) kann > 1 sein
- Es gilt nur eingeschränkte **Propagierungssemantik**
- Die zusammengesetzten Objekte bilden einen **gerichteten, azyklischen Graphen**



Starke Aggregation (= Komposition)

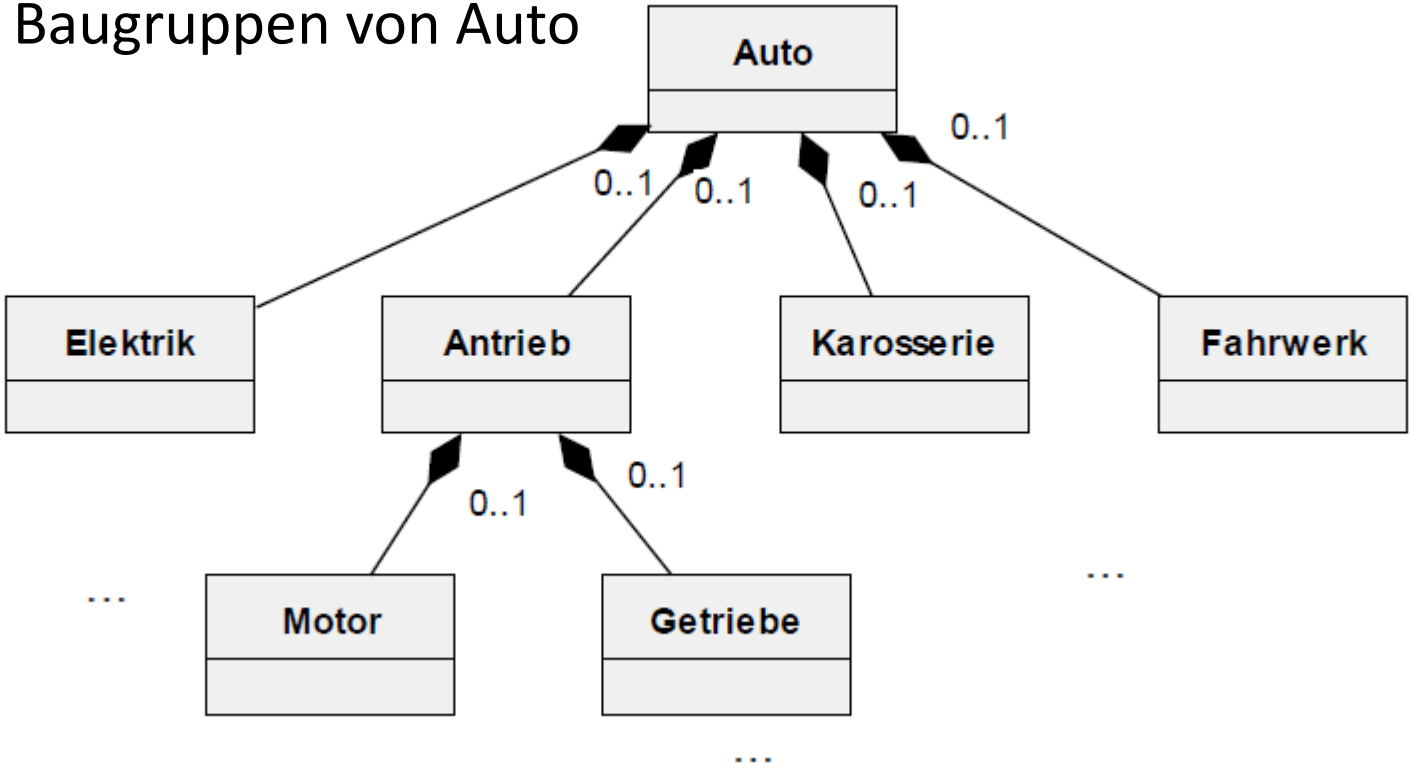


- Ein bestimmter Teil darf zu einem bestimmten Zeitpunkt in **maximal einem zusammengesetzten Objekt enthalten** sein
- Die **Multiplizität** des aggregierenden Endes der Beziehung kann (maximal) 1 sein
- **Abhängigkeit** der Teile vom zusammengesetzten Objekt
- **Propagierungssemantik**
- Die zusammengesetzten Objekte bilden einen **Baum**



Starke Aggregation

- Mittels starker Aggregation kann eine Hierarchie von „Teil-von“- Beziehungen dargestellt werden (Transitivität!)
- Beispiel: Baugruppen von Auto

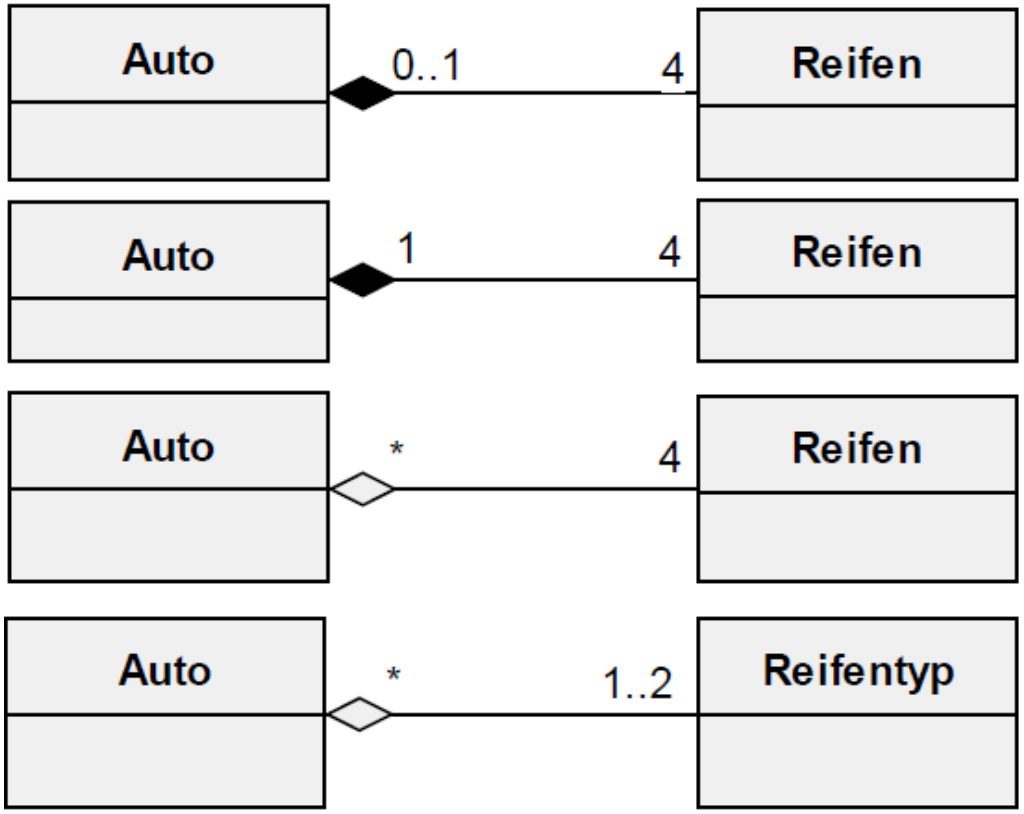


Starke Aggregation vs. Assoziation

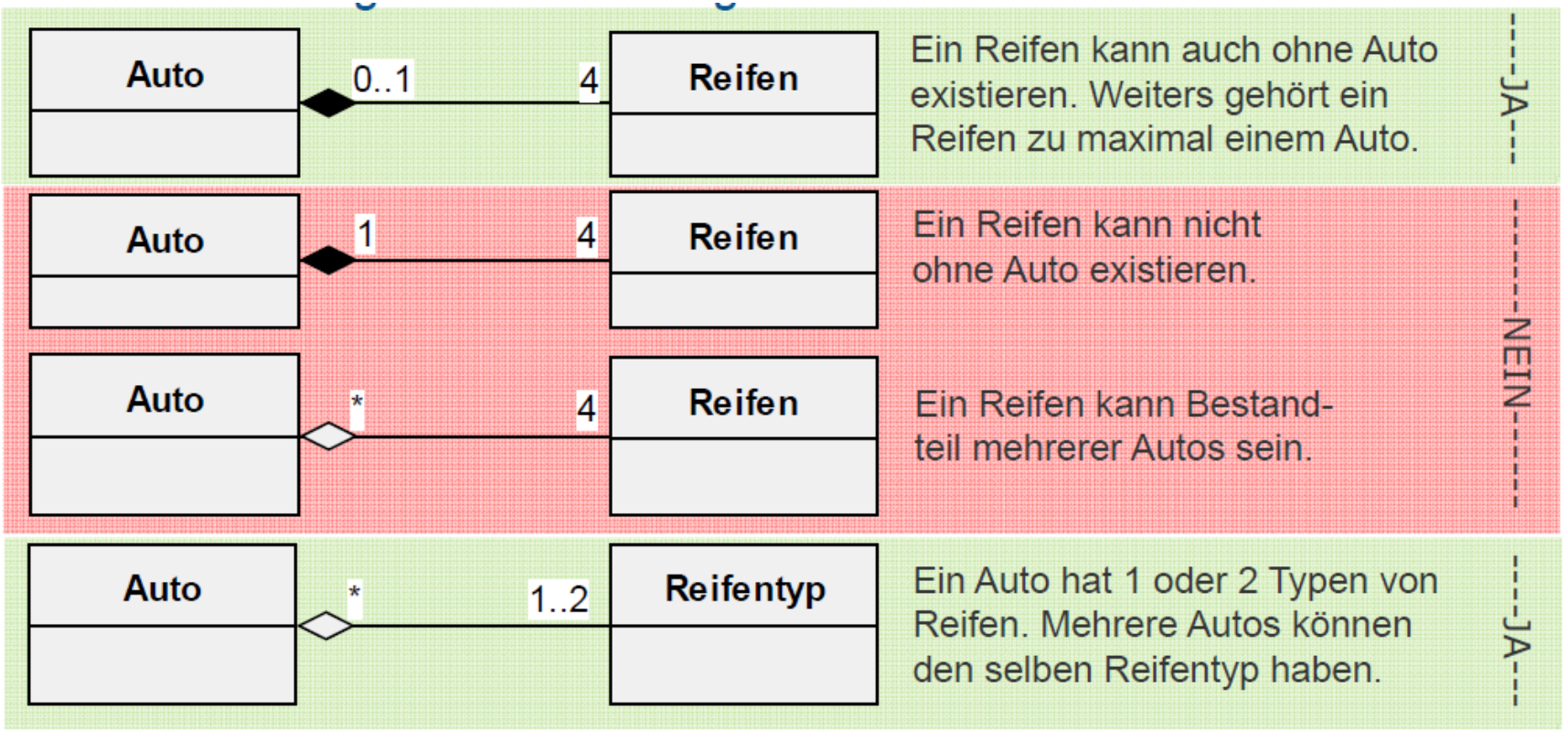
- **Einbettung**
  - Die Teile sind i.A. physisch im Kompositum enthalten
  - Über Assoziation verbundene Objekte werden über Referenzen realisiert
- **Sichtbarkeit**
  - Ein Teil ist nur für das Kompositum sichtbar
  - Das über eine Assoziation verbundene Objekt ist i.A. öffentlich sichtbar
- **Lebensdauer**
  - Das Kompositum erzeugt und löscht seine Teile
  - Keine Existenzabhängigkeit zwischen assoziierten Objekten
- **Kopien**
  - Kompositum und Teile werden kopiert
  - Nur die Referenzen auf assoziierte Objekte werden kopiert

Aufgabe: Komposition und Aggregation

- Welche der folgenden Beziehungen trifft zu?

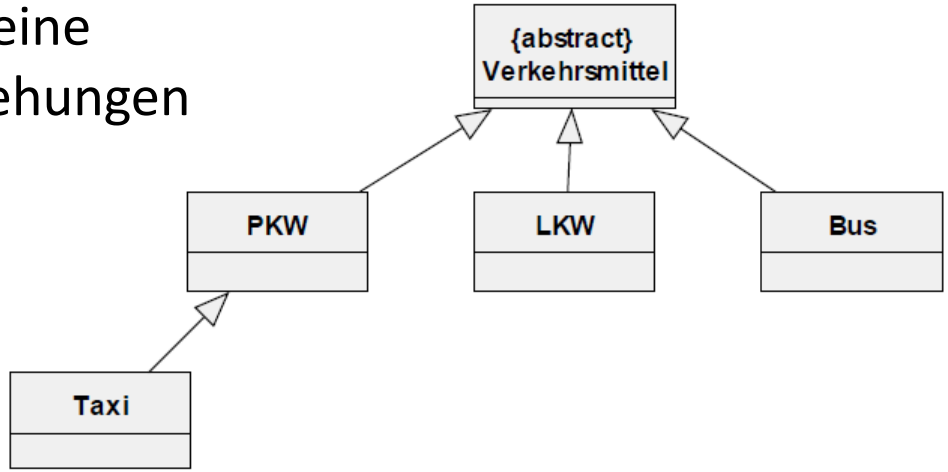


Lösung: Komposition und Aggregation



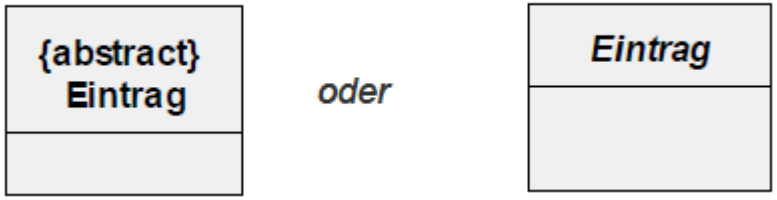
Generalisierung

- **Taxonomische Beziehung** zwischen einer spezialisierten Klasse und einer allgemeineren Klasse
  - Die spezialisierte Klasse **erbt** die Eigenschaften der allgemeineren Klasse
  - Kann **weitere Eigenschaften** hinzufügen
  - Eine **Instanz der Unterklasse** kann überall dort verwendet werden, wo eine **Instanz der Oberklasse** erlaubt ist (zumindest syntaktisch)
- Mittels Generalisierung wird eine Hierarchie von „ist-ein“- Beziehungen dargestellt (Transitivität!)

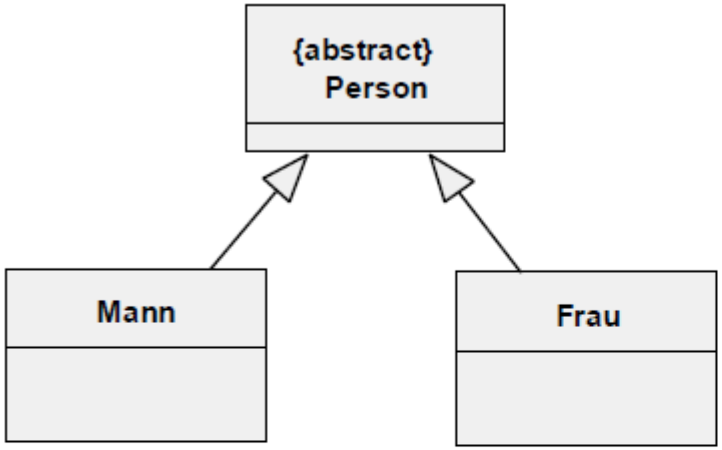


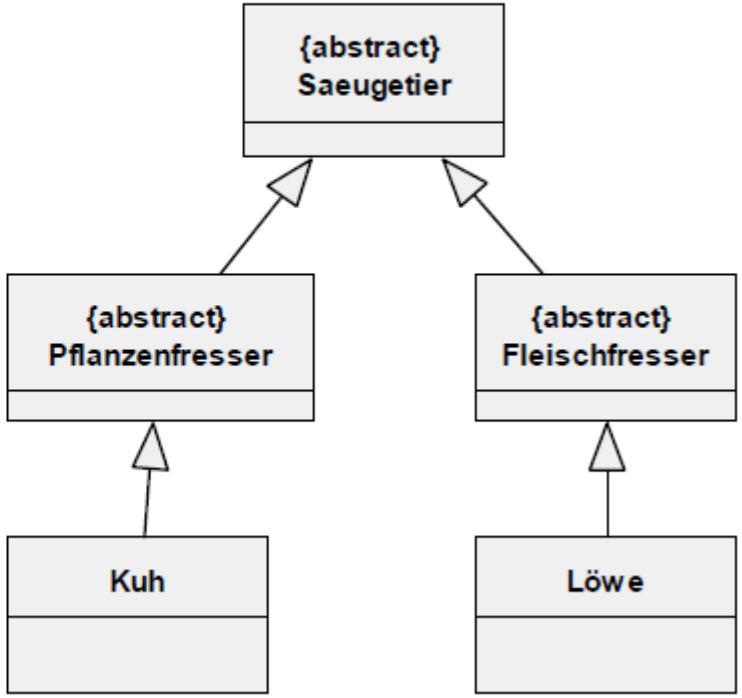
Abstrakte Klasse

- Klasse, die **nicht instanziert** werden kann
- **Nur in Generalisierungshierarchien** sinnvoll
- Dient zum **"Herausheben" gemeinsamer Merkmale** einer Reihe von Unterklassen
- Notation: Schlüsselwort {abstract} oder Klassenname in kursiver Schrift
- Mit analoger Notation wird zwischen konkreten (= implementierten) und abstrakten (= nur spezifizierten) **Operationen** einer Klasse unterschieden



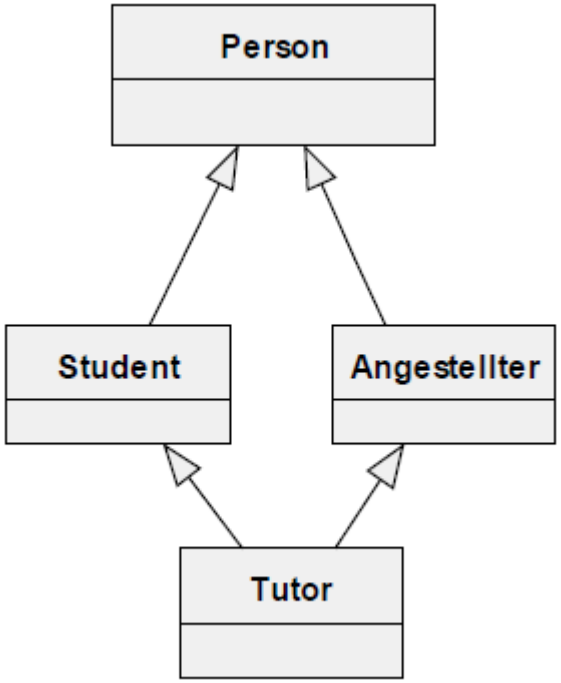
Abstrakte Klasse





Mehrfachvererbung

- Klassen müssen nicht nur eine Oberklasse haben, sondern können auch von mehreren Klassen erben

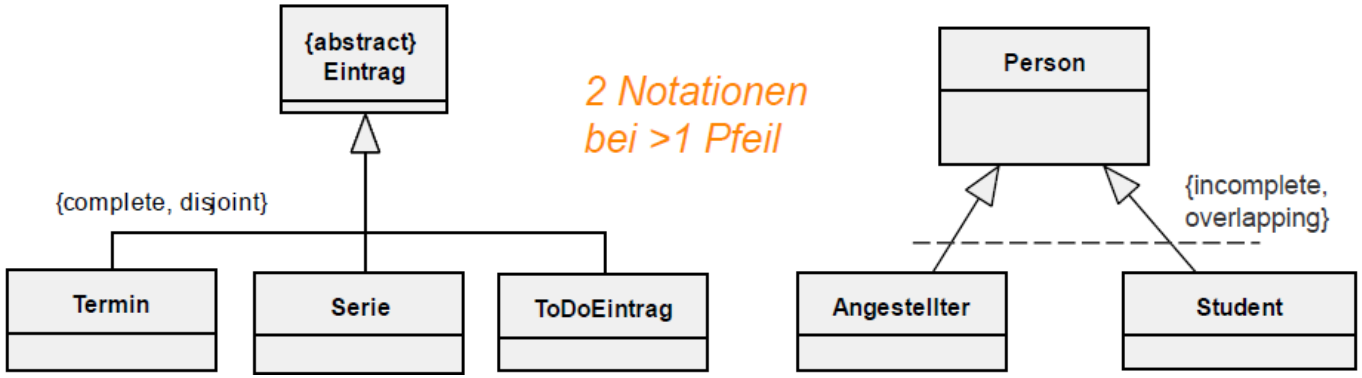


Generalisierung: Eigenschaften

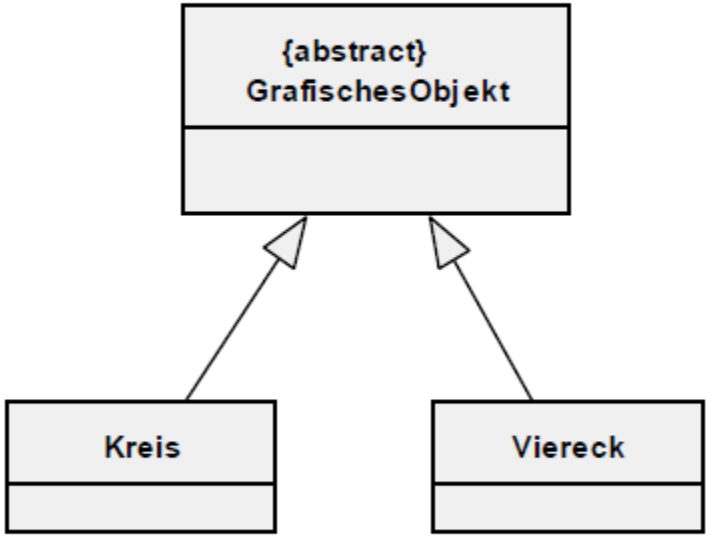
- Unterscheidung kann vorgenommen werden in
  - **Unvollständig / vollständig**:

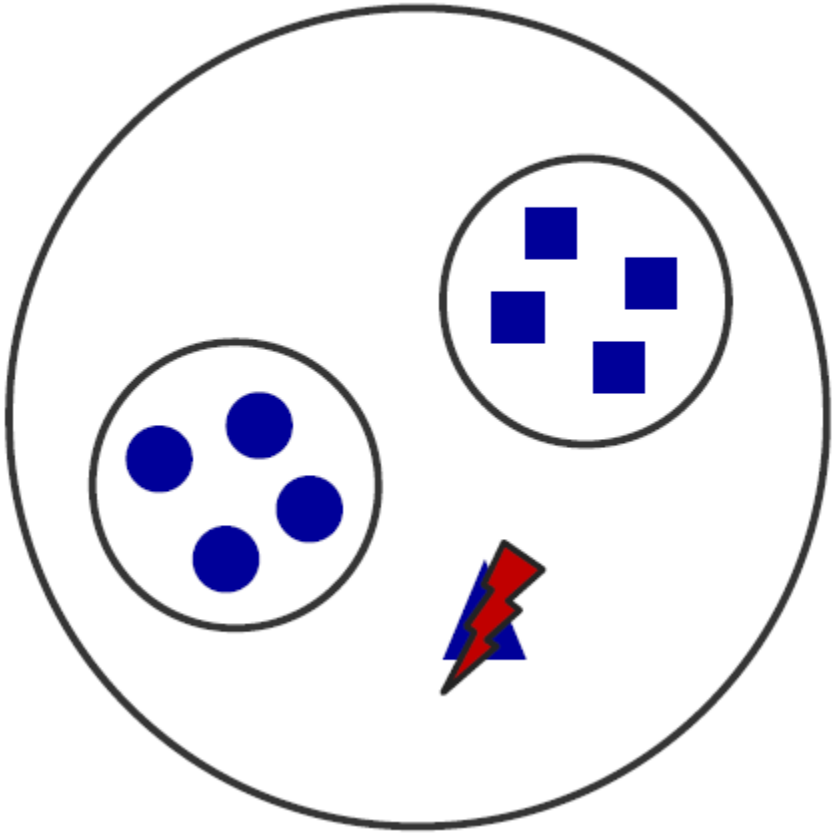
    In einer vollständigen Generalisierungshierarchie muss jede Instanz der Superklasse auch Instanz mindestens einer Subklasse sein
  - **Überlappend / disjunkt**:

    In einer überlappenden Generalisierungshierarchie kann ein Objekt Instanz von mehr als einer Subklasse sein
  - **Default**: unvollständig, disjunkt



Generalisierung: Eigenschaften

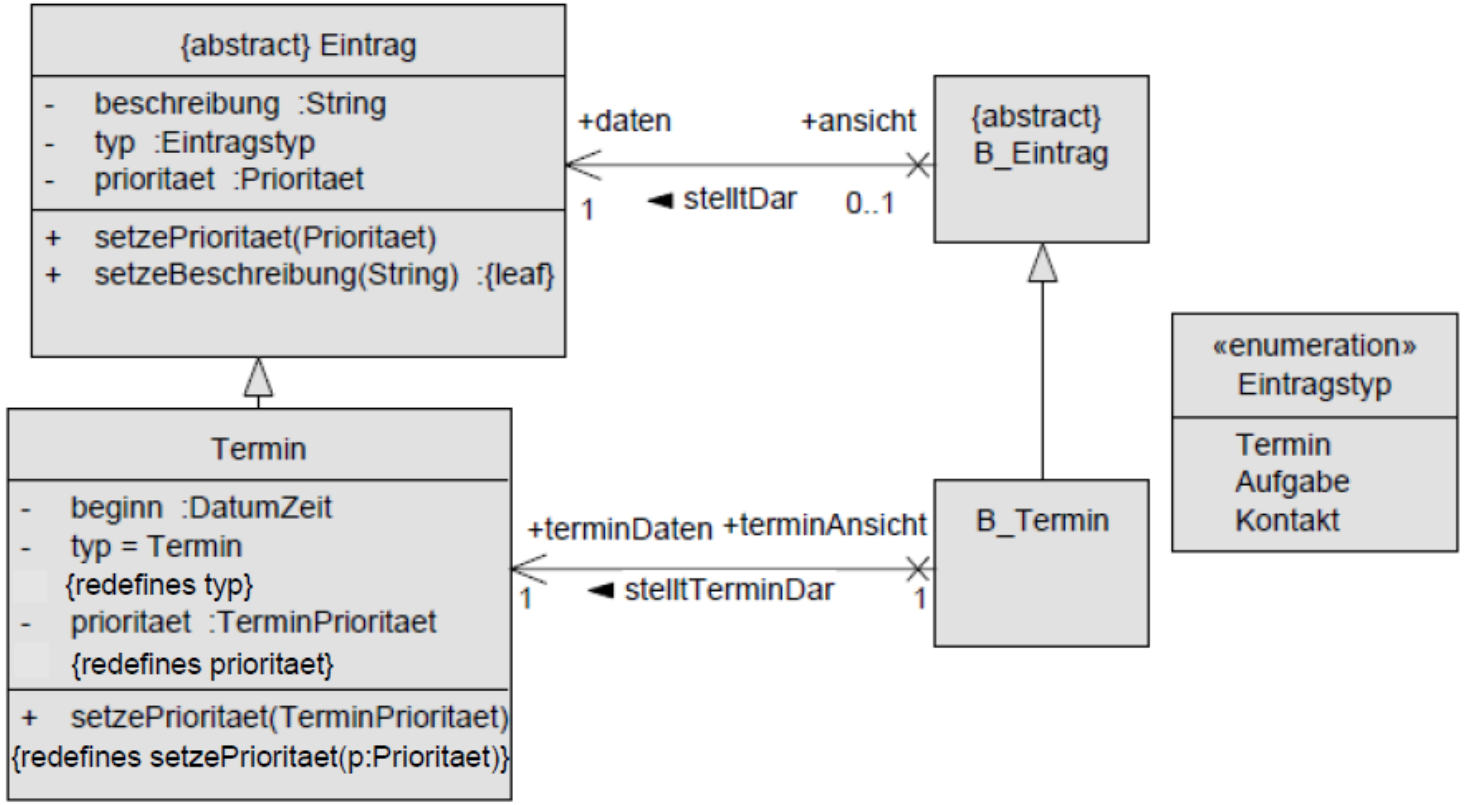




Generalisierung: Redefinition von geerbten Merkmalen

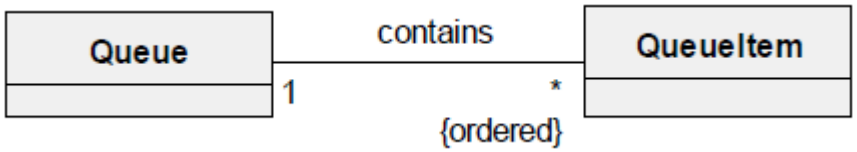
- Geerbte Merkmale können **in Subklasse redefiniert** werden
  - {redefines <feature>}
- **Redefinierbare Merkmale**
  - Attribute
  - Navigierbare Assoziationsenden
  - Operationen
- Redefinition von Operationen in (in)direkten Subklassen kann auch verhindert werden, indem die Operation mit der Eigenschaft {leaf} gekennzeichnet wird
- Das redefinierte Merkmal muss konsistent zum ursprünglichen Merkmal sein – verschiedenste **Konsistenzregeln** – z.B.
  - Ein redefiniertes Attribut ist konsistent zum ursprünglichen Attribut, wenn sein **Typ gleich oder ein Subtyp** des ursprünglichen Typs ist
  - Das **Intervall der Multiplizität** muss in jenem des ursprünglichen Attributs **enthalten** sein
  - Die Signatur einer Operation muss die **gleiche Anzahl an Parametern** aufweisen, etc.

Generalisierung: Redefinition von geerbten Merkmalen



Ordnung und Eindeutigkeit von Assoziationen

- **Ordnung** {ordered} ist **unabhängig von Attributen**

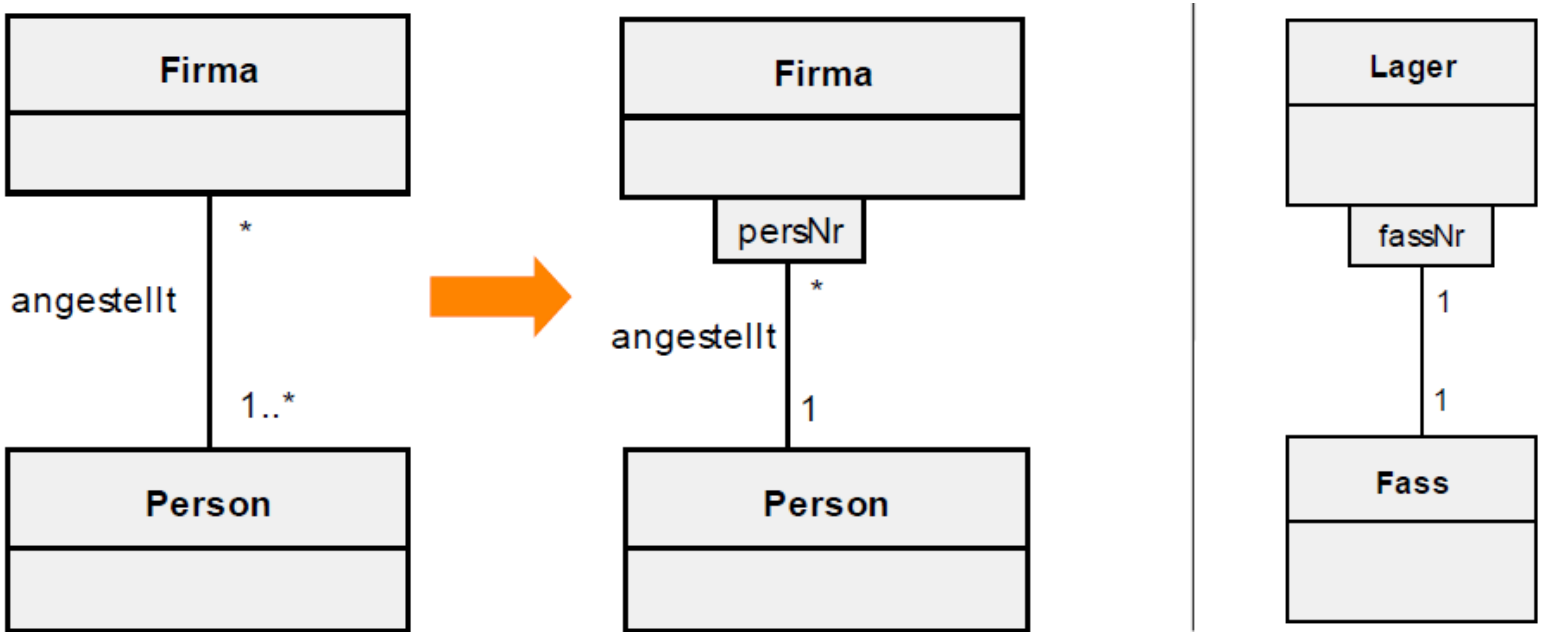


- **Eindeutigkeit**
  - Wie bei Attributen durch {unique} und {nonunique}
  - Kombination mit Ordnung {set}, {bag}, {sequence} bzw. {seq}

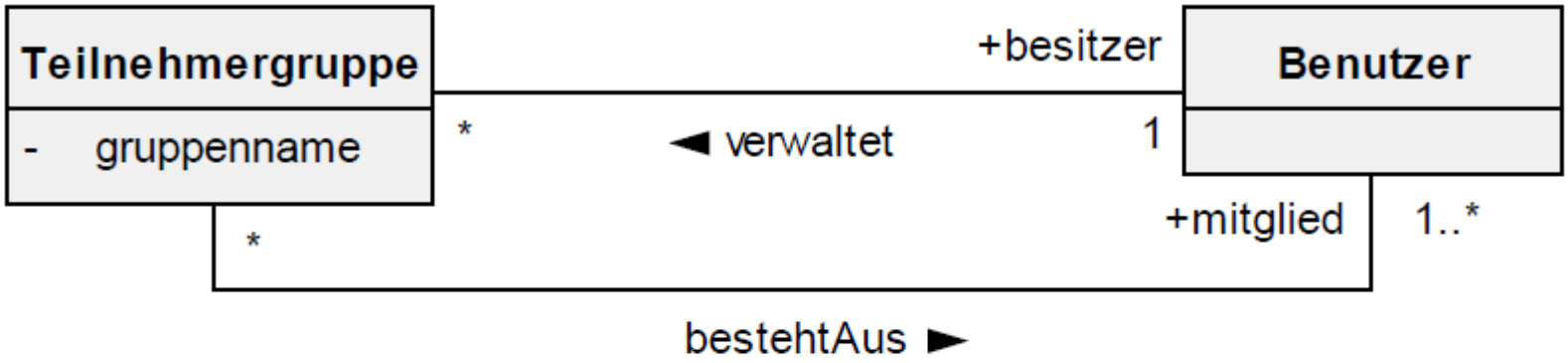
Eindeutigkeit	Ordnung	Kombination	Beschreibung
unique	*unordered*	set	Menge (Standardwert)
unique	*ordered*	*orderedSet*	Geordnete Menge
nonunique	*unordered*	*bag*	Multimenge, d.h. Menge mit Duplikaten
nonunique	*ordered*	*sequence*	Geordnete Menge mit Duplikaten (Liste)

Qualifizierte Assoziation

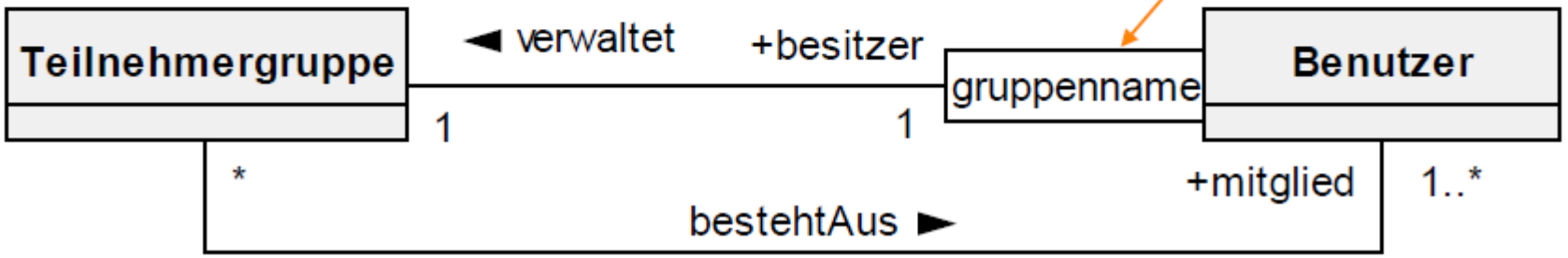
- Besteht aus einem **Attribut** oder einer **Liste von Attributen**, deren Werte die Objekte der assoziierten Klasse partitionieren
- **Reduziert** meist die **Multiplizität**
- Stellt eine **Eigenschaft der Assoziation** dar



Qualifizierte Assoziation



muss je Benutzer eindeutig sein!



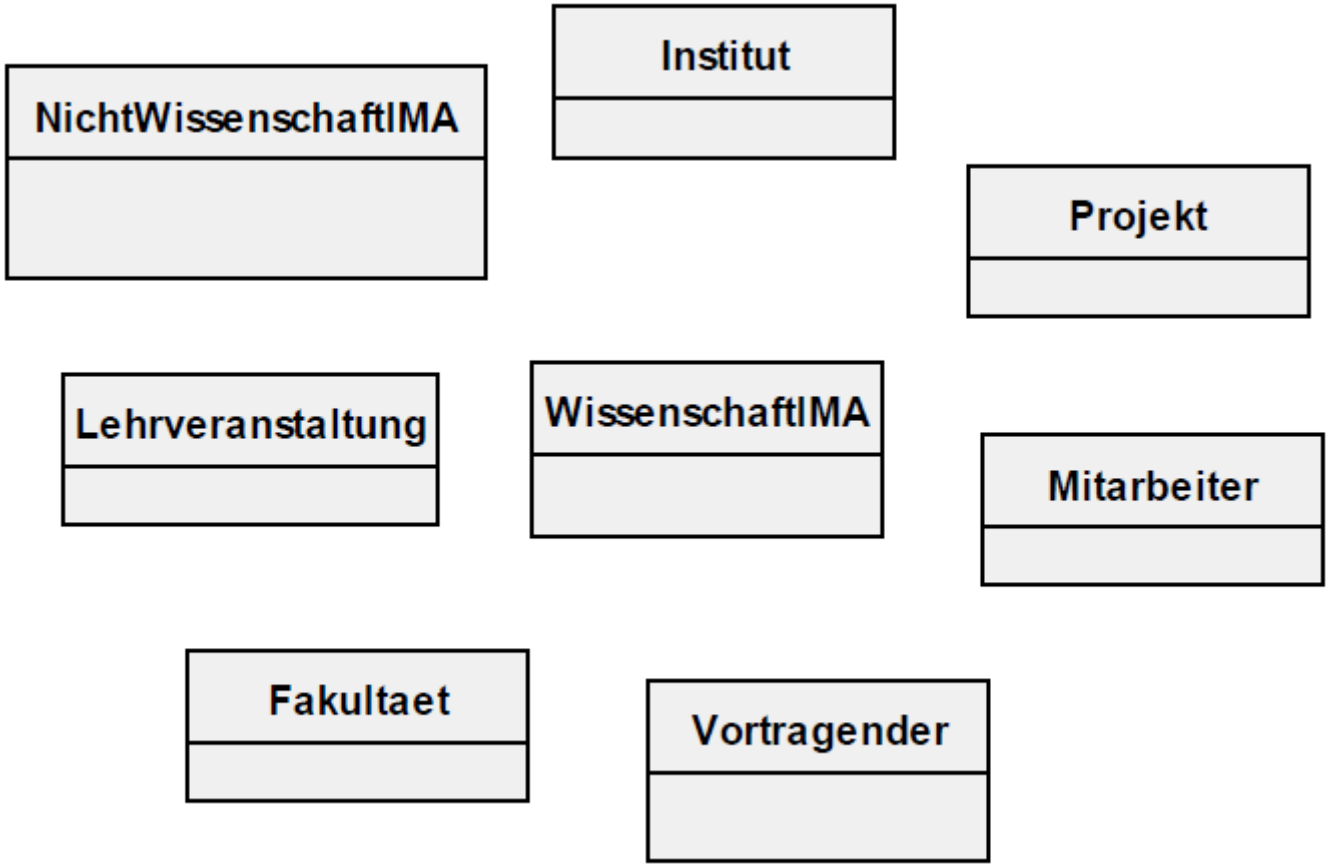
Aufgabe

- Gesucht ist ein vereinfacht dargestelltes Modell der TU Wien entsprechend der folgenden Spezifikation.
- Die Uni besteht aus mehreren Fakultäten, die sich wiederum aus verschiedenen Instituten zusammensetzen. Jede Fakultät und jedes Institut besitzt eine Bezeichnung. Für jedes Institut ist eine Adresse bekannt. Jede Fakultät wird von ihrem Dekan, einem Mitarbeiter, geleitet. Die Gesamtanzahl der Mitarbeiter ist bekannt. Mitarbeiter haben eine Sozialversicherungsnummer, einen Namen und eine E-Mail-Adresse. Es wird zwischen wissenschaftlichem und nicht-wissenschaftlichem Personal unterschieden. Wissenschaftliche Mitarbeiter sind zumindest einem Institut zugeordnet. Für jeden wissenschaftlichen Mitarbeiter ist seine Fachrichtung bekannt. Weiterhin können wissenschaftliche Mitarbeiter für eine gewisse Anzahl an Stunden an Projekten beteiligt sein, von welchen ein Name und Anfangs- und Enddatum bekannt sind. Manche wissenschaftliche Mitarbeiter führen Lehrveranstaltungen durch – diese werden als Vortragende bezeichnet. LVAs haben eine ID, einen Namen und eine Stundenanzahl.

Lösung: Klassen

- Die Uni besteht aus mehreren **Fakultäten**, die sich wiederum aus verschiedenen **Instituten** zusammensetzen. Jede Fakultät und jedes Institut besitzt eine Bezeichnung. Für jedes Institut ist eine Adresse bekannt. Jede Fakultät wird von ihrem Dekan, einem Mitarbeiter, geleitet. Die Gesamtanzahl der **Mitarbeiter** ist bekannt. Mitarbeiter haben eine Sozialversicherungsnummer, einen Namen und eine E-Mail-Adresse. Es wird zwischen **wissenschaftlichem** und **nicht-wissenschaftlichem Personal** unterschieden. Wissenschaftliche Mitarbeiter sind zumindest einem Institut zugeordnet. Für jeden wissenschaftlichen Mitarbeiter ist seine Fachrichtung bekannt. Weiterhin können wissenschaftliche Mitarbeiter für eine gewisse Anzahl an Stunden an **Projekten** beteiligt sein, von welchen ein Name und Anfangs- und Enddatum bekannt sind.
- Manche wissenschaftliche Mitarbeiter führen **Lehrveranstaltungen** durch - diese werden als **Vortragende** bezeichnet. LVAs haben eine ID, einen Namen und eine Stundenanzahl.

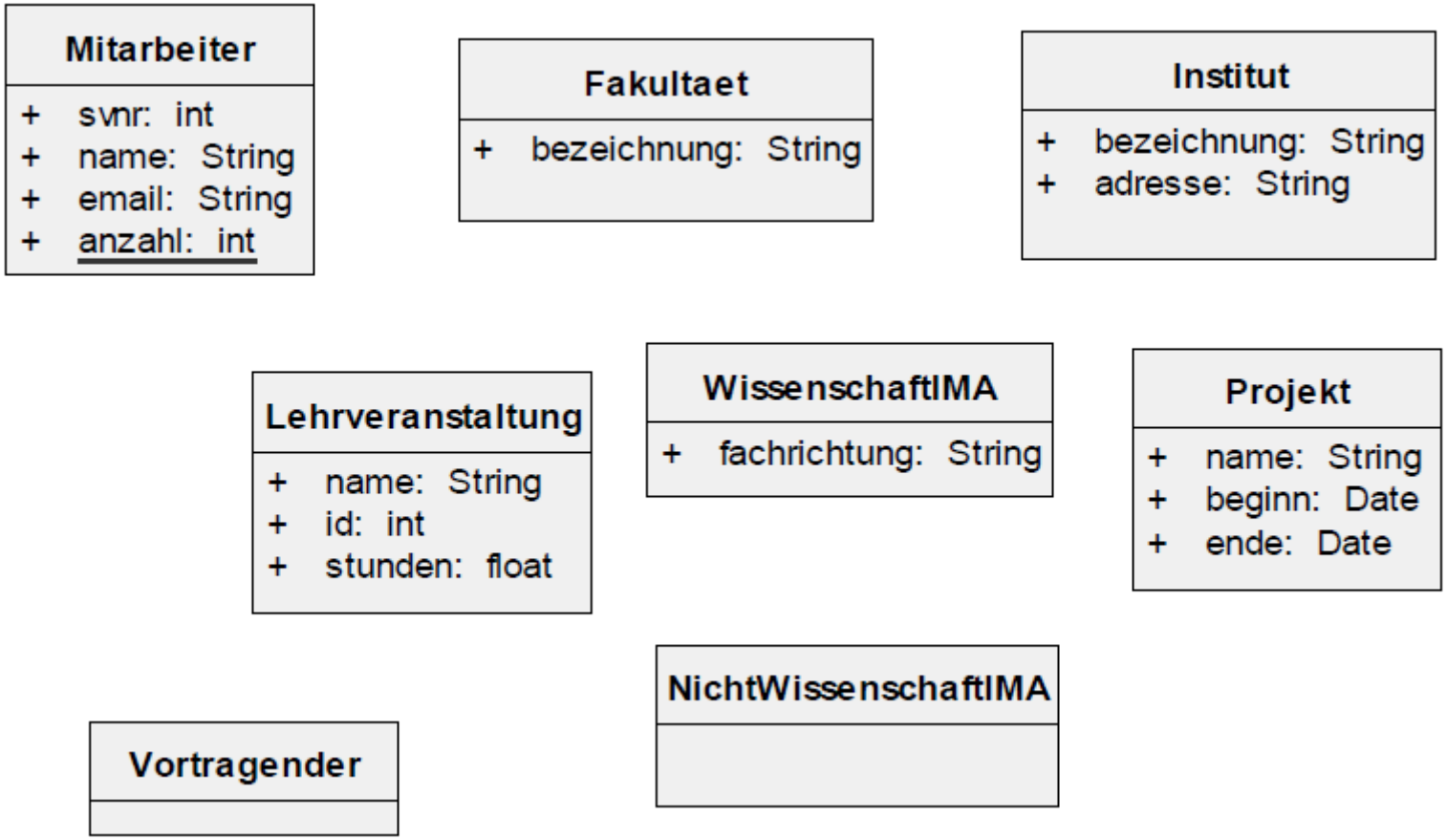
Lösung: Klassen



Lösung: Attribute

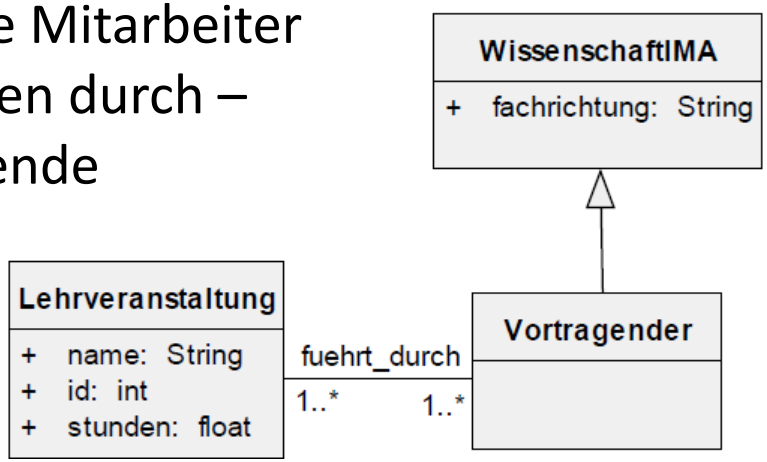
- Die Uni besteht aus mehreren Fakultäten, die sich wiederum aus verschiedenen Instituten zusammensetzen. Jede Fakultät und jedes Institut besitzt eine **Bezeichnung**. Für jedes Institut ist eine **Adresse** bekannt. Jede Fakultät wird von ihrem Dekan, einem Mitarbeiter, geleitet. Die **Gesamtanzahl** der Mitarbeiter ist bekannt. Mitarbeiter haben eine **Sozialversicherungsnummer**, einen **Namen** und eine **E-Mail-Adresse**. Es wird zwischen wissenschaftlichem und nicht-wissenschaftlichem Personal unterschieden. Wissenschaftliche Mitarbeiter sind zumindest einem Institut zugeordnet. Für jeden wissenschaftlichen Mitarbeiter ist seine **Fachrichtung** bekannt. Weiterhin können wissenschaftliche Mitarbeiter für eine **gewisse Anzahl an Stunden** an Projekten beteiligt sein, von welchen ein **Name** und **Anfangs-** und **Enddatum** bekannt sind. Manche wissenschaftliche Mitarbeiter führen Lehrveranstaltungen durch - diese werden als Vortragende bezeichnet. LVAs haben eine **ID**, einen **Namen** und eine **Stundenanzahl**.

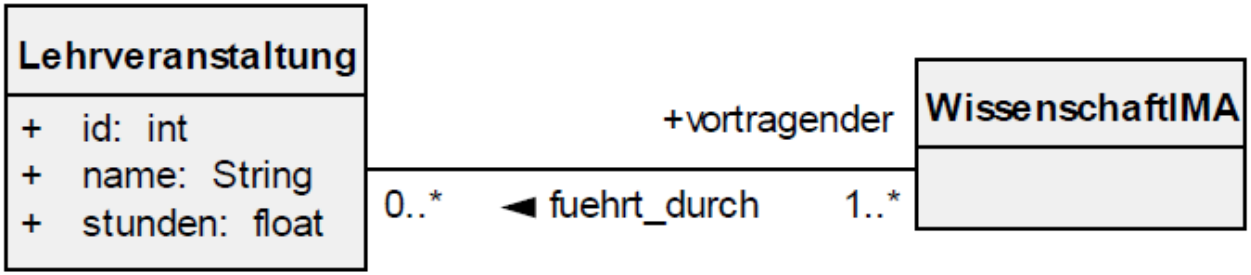
Lösung: Attribute



Assoziation

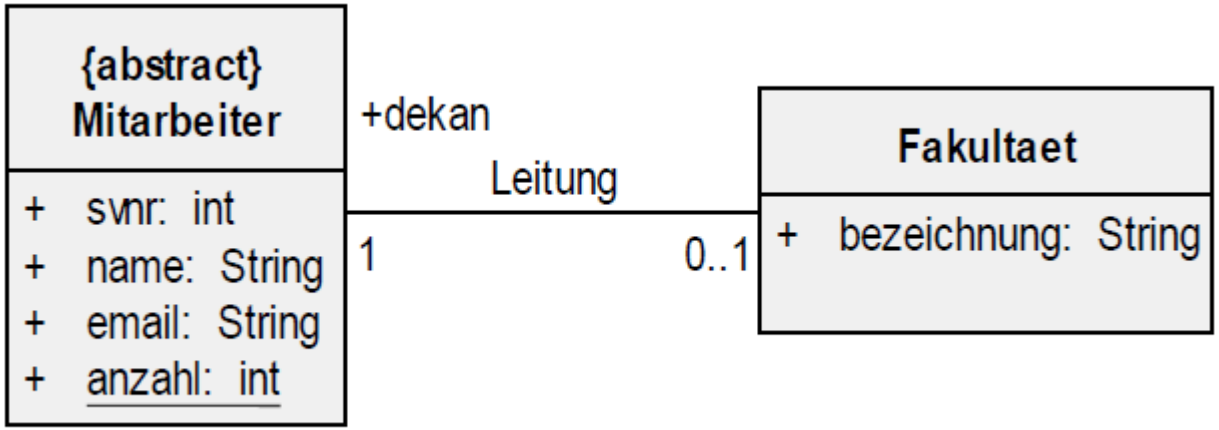
- Manche wissenschaftliche Mitarbeiter führen Lehrveranstaltungen durch – diese werden als Vortragende bezeichnet.



- 

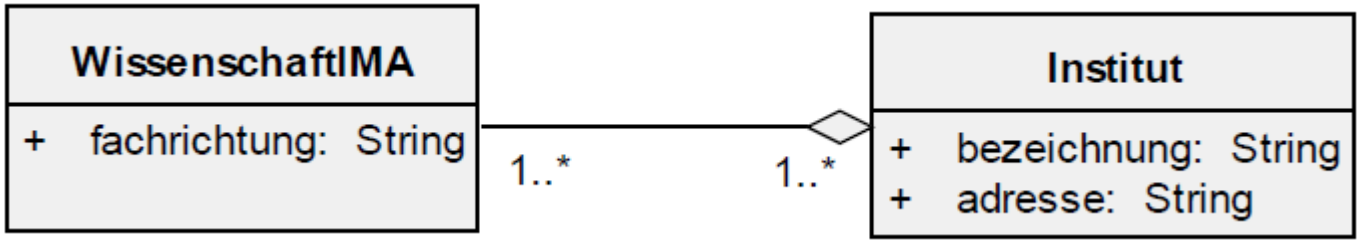
Assoziation

- Jede Fakultät wird von ihrem Dekan, einem Mitarbeiter, geleitet.



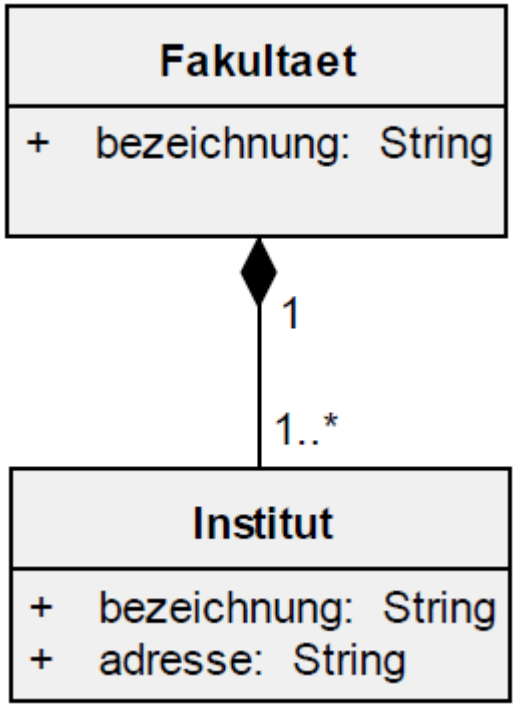
Schwache Aggregation

- Wissenschaftliche Mitarbeiter sind zumindest einem Institut zugeordnet.



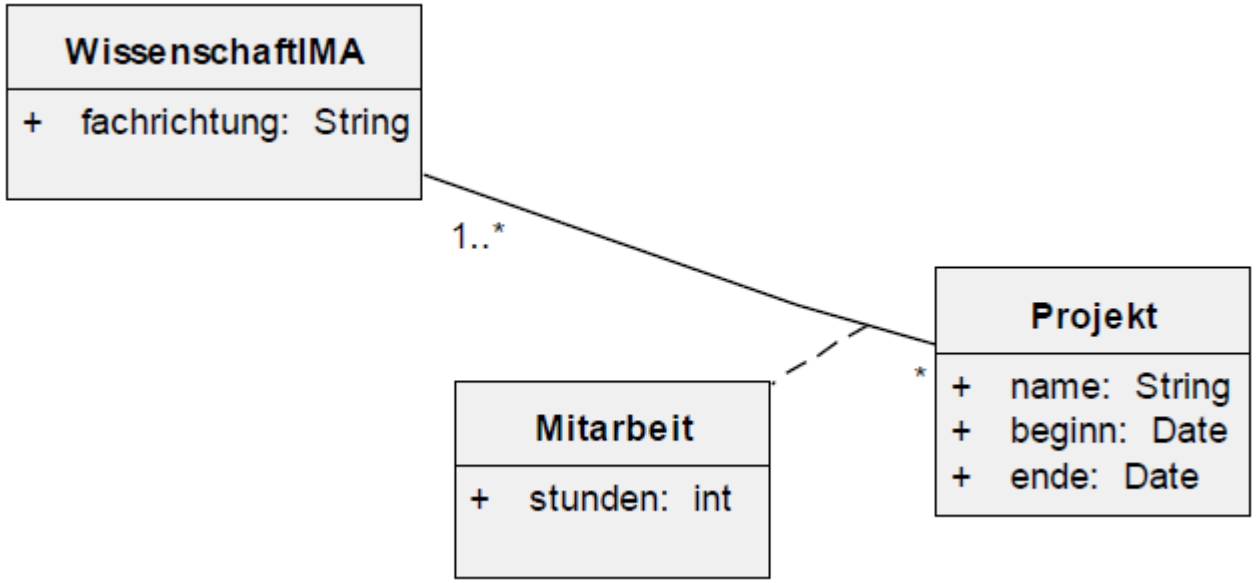
Starke Aggregation

- Die Uni besteht aus mehreren Fakultäten, die sich wiederum aus verschiedenen Instituten zusammensetzen.

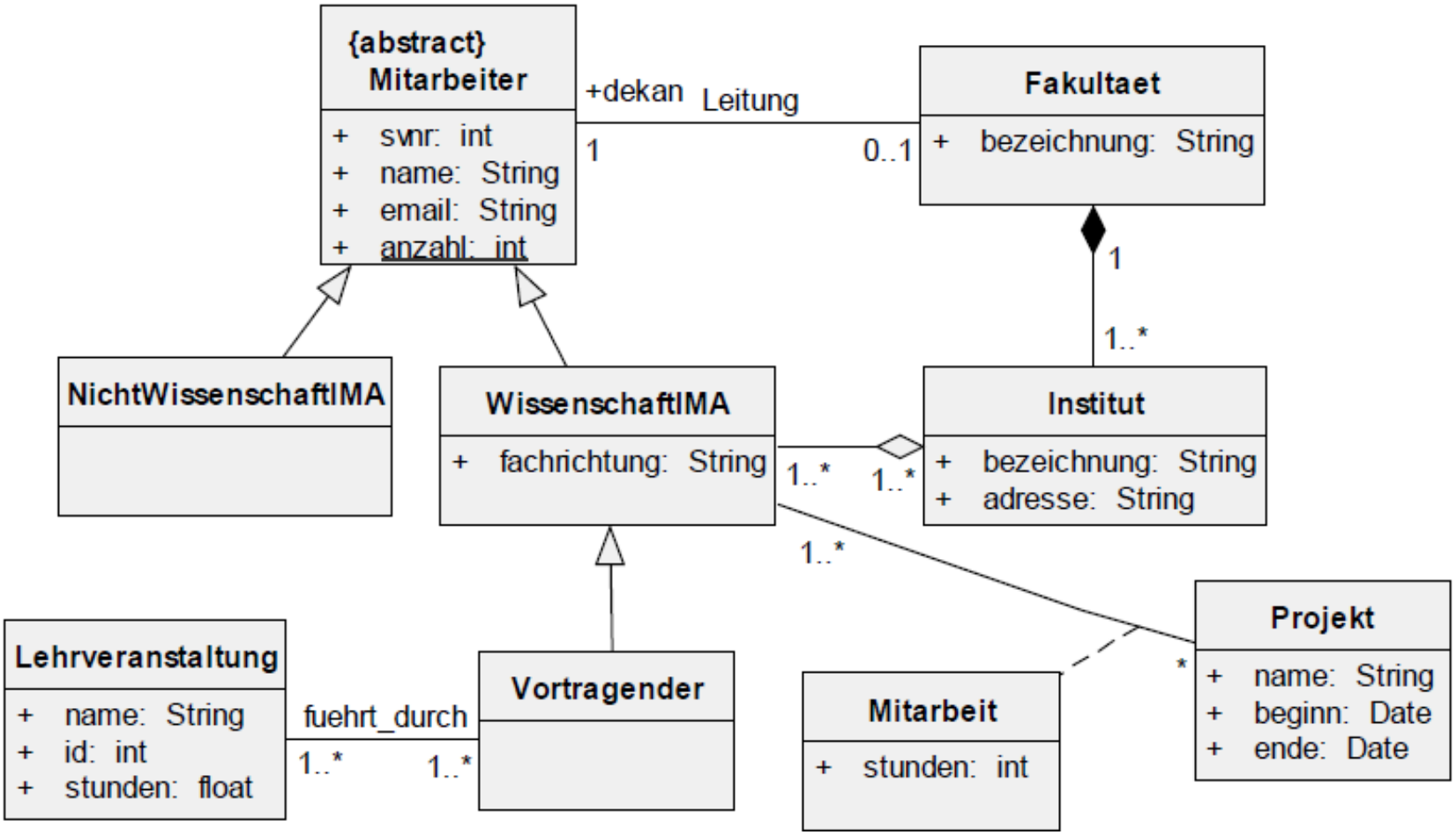


Assoziationsklasse

- Weiterhin können wissenschaftliche Mitarbeiter an Projekten beteiligt sein.



Gesamtlösung



Aufgabe

- Jede Person hat einen Namen, eine Telefonnummer und E-Mail. Jede Wohnadresse wird von nur einer Person bewohnt. Es kann aber sein, dass einige Wohnadressen nicht bewohnt sind. Den Wohnadressen sind je eine Strasse, eine Stadt, eine PLZ und ein Land zugeteilt. Alle Wohnadressen können bestätigt werden und als Beschriftung (für Postversand) gedruckt werden. Es gibt zwei Sorten von Personen: Student, welcher sich für ein Modul einschreiben kann und Professor, welcher einen Lohn hat. Der Student besitzt eine Matrikelnummer und eine Durchschnittsnote. Modellieren Sie diesen Sachverhalt mit einem UML Klassendiagramm.

Aufgabe

Entwerfen Sie ein vollständiges Klassendiagramm.

Allgemeine Infos zum Aufgabenumfeld:

- Sie entwickeln Software für ein Computer-Schachspiel. Ihre Aufgabe ist der Entwurf des Spielbretts mit den Figuren. Zeigen Sie bei Assoziationen die Richtung des Zugriffs durch Pfeile an

Die folgenden Informationen sollen darstellt werden:

- Ein Schach-Spiel besteht aus einem Schachbrett und 2 Mannschaften. Das Schachbrett besteht aus 64 Feldern, die jeweils eine x- und eine y-Koordinate haben. Die Mannschaften unterscheiden sich durch die Farbe. Jede Mannschaft besteht aus insgesamt 16 Figuren. Das sind 8 Bauern, 2 Türme, 2 Läufer, 2 Springer, 1 Dame und 1 König. Sorgen Sie dafür, dass die folgenden Informationen im Modell enthalten sind:
  - Jede Figur steht entweder auf einem Feld oder wurde bereits geschlagen
  - Jede Figur weiß, zu welcher Mannschaft sie gehört
  - Umgekehrt kennt auch jede Mannschaft ihre Figuren
  - Stellen Sie im Klassendiagramm dar, dass eine Figur nicht gleichzeitig z. B. Bauer und Läufer sein darf. Eine Figur ist entweder ein Bauer, eine Dame, ein Turm, oder...
  - Jedes Feld weiß, ob es durch eine Figur besetzt ist – und wenn ja mit welcher
  - Jede Figur soll eine Methode *moveTo* bieten, die es erlaubt die Figur auf ein anderes Zielfeld zu bewegen.