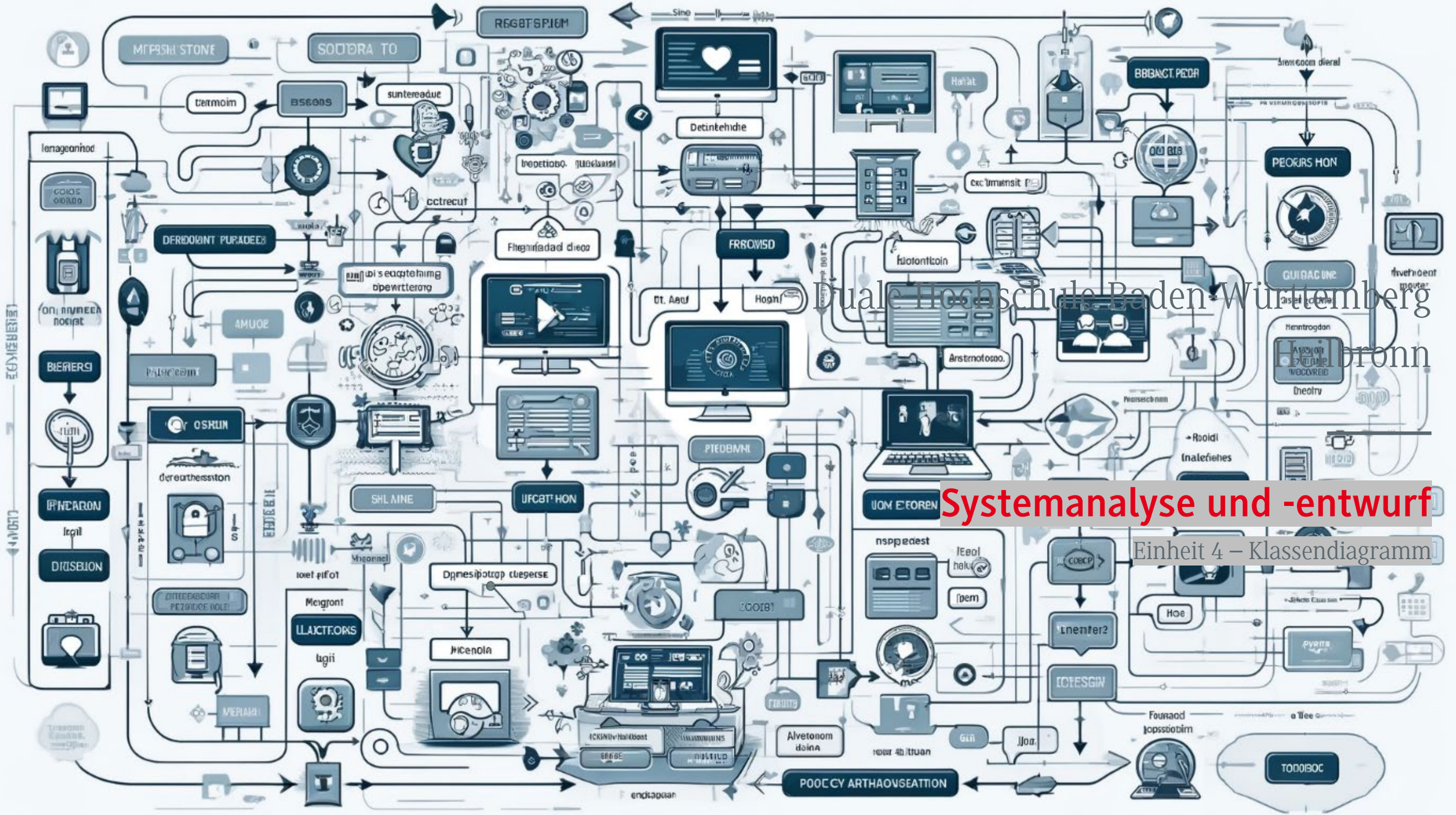


Duale Hochschule Baden-Württemberg

Heilbronn

Systemanalyse und -entwurf

Einheit 4 – Klassendiagramm

Agenda Systemanalyse und -entwurf

	Thema	Inhalte	(Geplanter) Termin
1	Einführung	Einführung in Veranstaltung. Modulhandbuch, Lernziele und Überblick, empfohlene Literatur. Requirements Game. Überblick Anforderungsmanagement und UML. Aufteilung in Teams und Themen.	31.03 NM
2	Systemanalyse und Entwurf	Das **UML-Anwendungsfalldiagramm** (Use Cases). Zweck und Bestandteile, erläutert mit Beispielen. Umsetzung eines Beispiels in Kleingruppen.	02.04. NM
3	Systemanalyse und Entwurf	Das **UML-Aktivitätsdiagramm**. Zweck und Bestandteile, erläutert mit Beispielen. Umsetzung eines Beispiels in Kleingruppen Vorbereitung zur Anforderungssammlung.	03.04. VM
4	Action! Anforderungssammlung	**Anforderungssammlung** für Projekte (13:30-15:00) mit Kunden oder Brainstorming. Danach (zusammen): Reflexion, Arbeit in Teams (Tooling und Modellierung), Puffer	15.04. NM
5	Guest Lecture	Guest Lecture „Moderne Software-Entwicklung in großen Unternehmen: von der Idee zum Produkt am Beispiel der SAP Internen Development Plattform (IDP) Hyperspace“. Das **UML-Klassendiagramm**. Zweck und Bestandteile, erläutert mit Beispielen. Umsetzung eines Beispiels in Kleingruppen	17.04. VM
6	Präsentationen	**Präsentationen** und Integration der Projekte, Reflexion	14.05 NM
7	Integration	**Erstellen von Beispielsaufgaben** inkl. Musterlösungen in Teams.	21.05 NM
8	Integration	**Messe der Diagramme, Reflexion** **Verankern** des erlernten Vorgehens mit UML an weiteren Beispielen. Entweder einzeln oder in Kleingruppen.	03.06 VM
9	Abschluss	Fragerunde, Wiederholungen, Vorbesprechung **Klausur**	03.06. NM

Willkommen im Sprint 2

Sprint 2 - Wichtigsten Grundlagen und Konzepte

Für Teilnehmer/innen verborgen

Für Teilnehmer/innen verborgen

Backlog

- ✓ Als Professorin möchte ich Studierenden vermitteln, was Use Case Diagramme sind, damit sie sie einsetzen können.
- ✓ Als Professorin möchte ich Studierenden vermitteln, wie Use Case Diagramme zu modellieren sind, damit sie sie Anforderungen in Use Cases übersetzen können.
- Als Professorin möchte ich Studierenden vermitteln, was Klassendiagramme sind, damit sie sie einsetzen können.
- Als Professorin möchte ich Studierenden vermitteln, wie Klassendiagramme zu modellieren sind, damit sie sie Anforderungen in Use Cases übersetzen können.
- Als Professorin möchte ich Studierenden vermitteln, was Aktivitätsdiaramme sind, damit sie sie einsetzen können.
- Als Professorin möchte ich Studierenden vermitteln, wie Aktivitätsdiaramme zu modellieren sind, damit sie sie Anforderungen in Use Cases übersetzen können.
- ✓ Als Student:in möchte ich verstehen, wie Anforderungen erhoben werden, damit ich Kunden aufmerksam zuhören und richtige Fragen stellen kann
- Als Student:in möchte ich Notation der Diagramme kennen und richtig einsetzen, damit ich mich auf die Inhalte konzentrieren kann
- Als Student:in möchte ich klausurrelevante Inhalte möglichst früh kennenlernen, damit ich sie besser für die Klausur verinnerlichen kann

Ansicht der Veranstaltung im Moodle

Klassendiagramm

Agenda

- Einführung Klassendiagramm
- Klassen
- Attribute und Operationen und deren Identifikation
- Assoziationen: Grundlagen
- Objekte als Instanzen von Klassen (Objektdiagramm)
- Assoziationen: Vertiefung
- Aggregationen
- Generalisierung und Interfaces
- Datentypen
- Beispiel: Informationssystem Universitätsverwaltung
- Zusammenfassung Elemente / Rückblick und Ausblick

Die Einheit basiert mit freundlicher Genehmigung von Frau Prof. Dr. Seidl auf Inhalten der Veranstaltung „Objektorientierte Modellierung“ an der TU Wien

Klassendiagramm

Agenda

- **Einführung Klassendiagramm**
- **Klassen**
- **Attribute und Operationen und deren Identifikation**
- Assoziationen: Grundlagen
- Objekte als Instanzen von Klassen (Objektdiagramm)
- Assoziationen: Vertiefung
- Aggregationen
- Generalisierung und Interfaces
- Datentypen
- Beispiel: Informationssystem Universitätsverwaltung
- Zusammenfassung Elemente / Rückblick und Ausblick

Einführung Klassendiagramm

- **Leitfrage: Wie sind die Daten und das Verhalten meines Systems im Detail strukturiert?**

- Das Klassendiagramm…
  - bildet das „**Herzstück**“ der UML
  - basiert auf den **Prinzipien der Objekt-Orientierung** (Abstraktion, Kapselung, Vererbung, …)
  - beschreibt den **strukturellen** Aspekt eines Systems auf Typebene in Form von Klassen, Interfaces und Beziehungen

- Es ist durch seine **Vielseitigkeit in allen Phasen** eines Projekts einsetzbar
  - in der Analysephase: als Domänenmodell, versucht Abbild der Wirklichkeit darzustellen
  - in der Designphase: werden damit Datenstrukturen und die Software modelliert
  - in der Implementierungsphase: kann daraus Source-code generiert werden

- Klasse in UML: Schablone, Typ
- Objekt: Ausprägung einer Klasse

Prinzipien der Objektorientierung

Kapselung

- Kombiniert Daten und Methoden in Klassen
- Schützt den internen Zustand eines Objekts
- Fördert Wiederverwendbarkeit und Reduzierung von Komplexität

Vererbung

- Ermöglicht Klassen, Eigenschaften von anderen Klassen zu übernehmen
- Unterstützt die hierarchische Klassifizierung
- Fördert Code-Wiederverwendung und erleichtert die Wartung

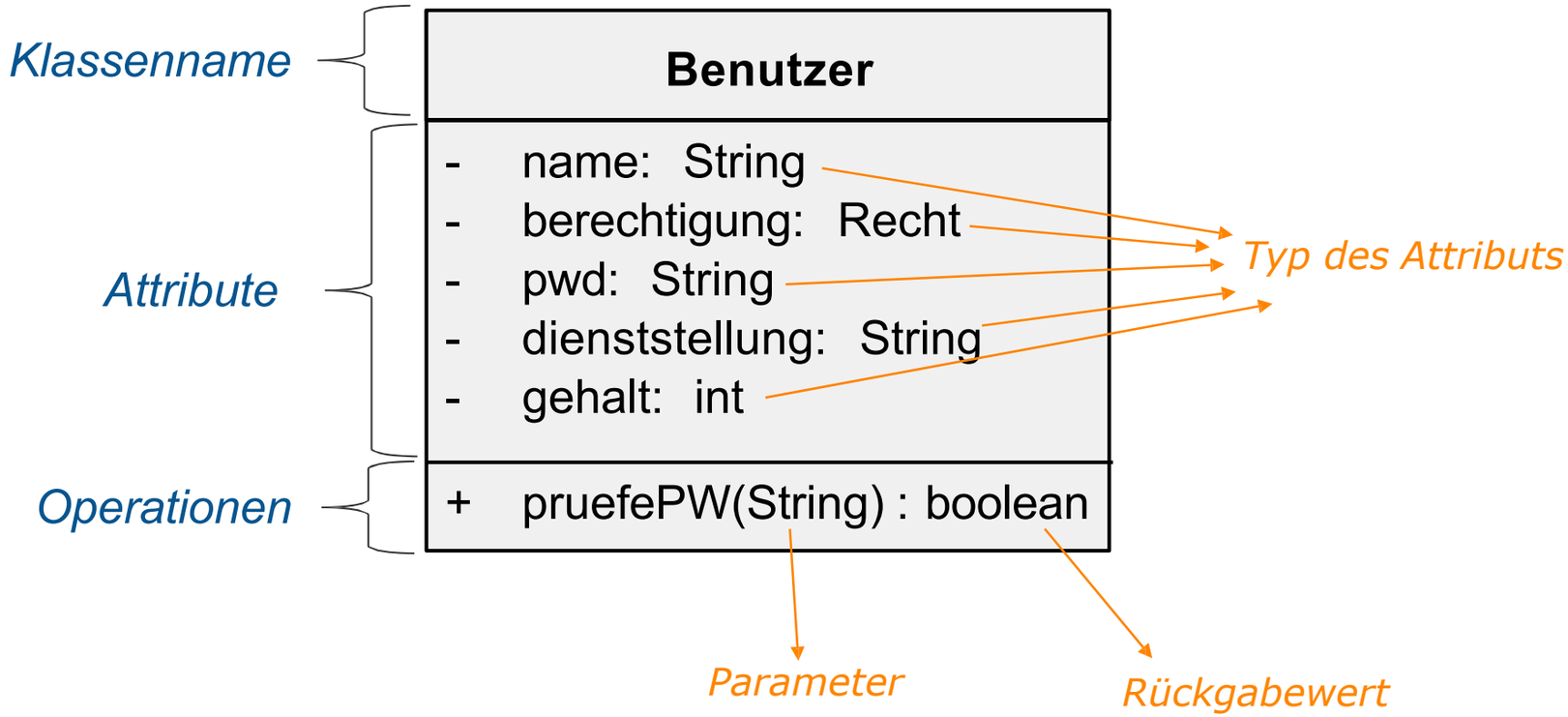
Polymorphismus

- Unterschiedliche Methodenimplementierungen unter gleichem Namen
- Überladen (@Overload: gleicher Name, unterschiedliche Parameter)
- Überschreiben (@Override: gleicher Name und Parameter, unterschiedliche Implementierung)

Abstraktion

- Vereinfacht komplexe Realitäten durch Modellierung relevanter Aspekte
- Definiert abstrakte Rahmenwerke für Implementierungsdetails
- Erhöht Flexibilität in der Softwareentwicklung

Notation für Klassen



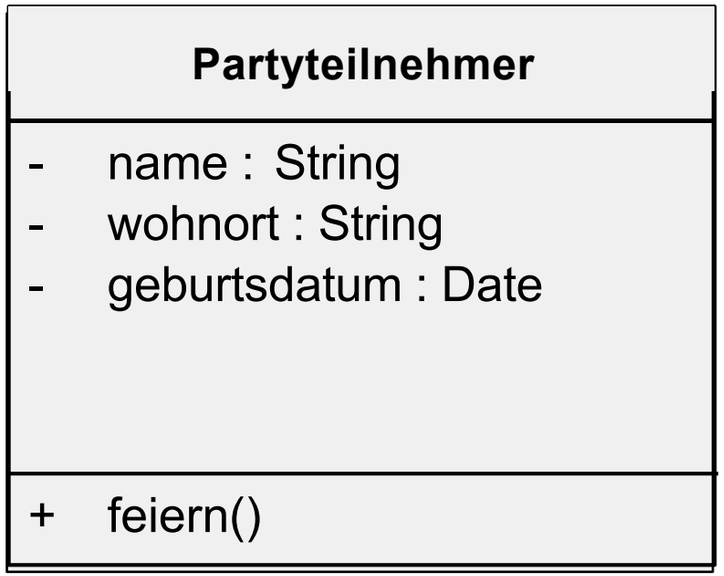
Notation für Klassen

Klasse

- Eine Menge der Objekten, die über gemeinsame Semantik, Eigenschaften und Verhalten verfügen
- Klasse in UML: Schablone, Typ

Objekt

- Ausprägung einer Klasse
- Jedes Attribut der Klasse bekommt einen Wert
- Name und Geburtsdatum sind untrennbar vom Teilnehmer



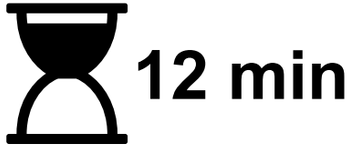
Beispiel: Identifikation von Klassen und Attributen

- Identifizieren Sie im unten stehenden Text passende Klassen und Attribute
- Überlegen Sie sich passende Datentypen für die Attribute

- Beschreibung:

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.

- Erledigen Sie die Aufgabe in Kleingruppen
- Sie haben 12 Minuten Zeit
- Gemeinsame Ergebnisdiskussion

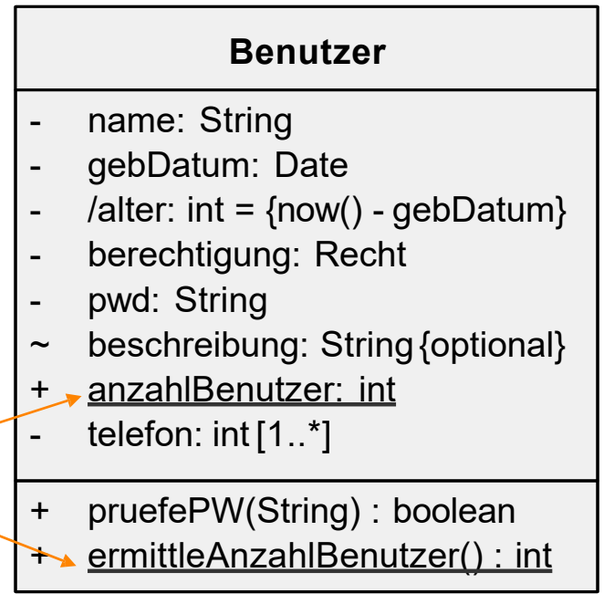


Attribute und Operationen

- 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

Alle nicht unterstrichenen: Instanzattribute/ -operationen

Klassenattribute/-operationen



- **Kann man dieses Diagramm als Entity Relationship Diagramm verwenden?**

Identifikation von Klassen – eine „Anleitung“

- **Ansatz vorgestellt in Publikation:** Linguistische Analyse der Problembeschreibung
  nach R.J. Abbott, Program Design by Informal English Descriptions, CACM, Vol. 26, No. 11, 1983

- **Hauptwörter** herausfiltern

- Faustregeln
  - 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

- **Fazit:** Ansatz nicht perfekt und nicht eindeutig, aber ein tauglicher Anfang

Identifikation von Attributen – eine „Anleitung“

- **Adjektive** und **Partizipien** herausfiltern

- Faustregeln
  - 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 im Allgemeinen nur **unvollständig** in der Anforderungsbeschreibung definiert

Identifikation von Operationen – eine „Anleitung“

- **Verben** herausfiltern

- Faustregeln
  - 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?

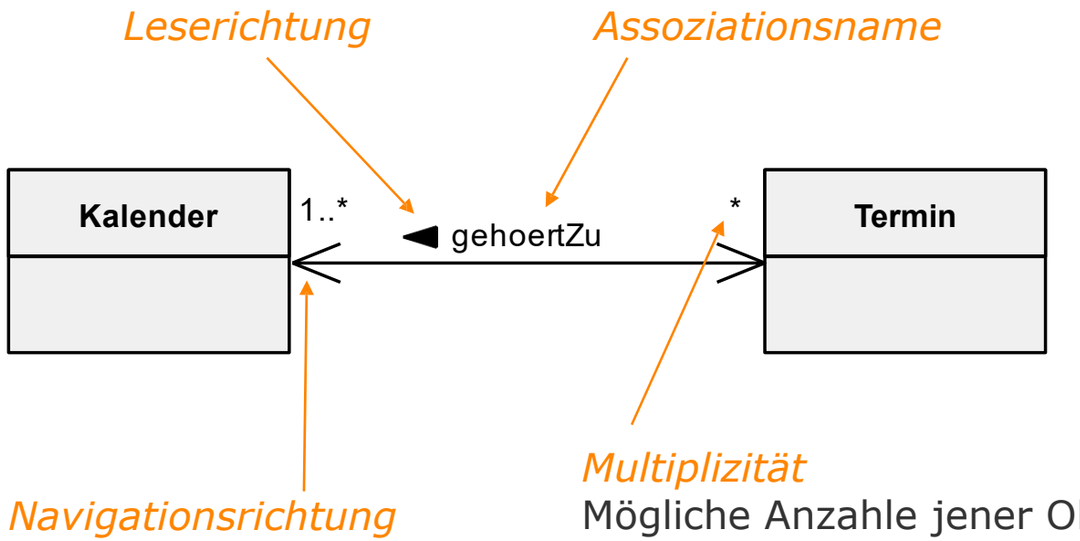
Klassendiagramm

Agenda

- Einführung Klassendiagramm
- Klassen
- Attribute und Operationen und deren Identifikation
- **Assoziationen: Grundlagen**
- Objekte als Instanzen von Klassen (Objektdiagramm)
- Assoziationen: Vertiefung
- Aggregationen
- Generalisierung und Interfaces
- Datentypen
- Beispiel: Informationssystem Universitätsverwaltung
- Zusammenfassung Elemente / Rückblick und Ausblick

Assoziation

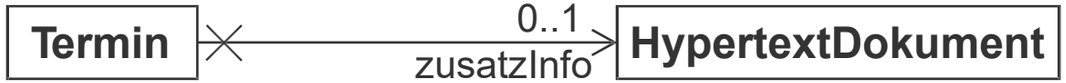
- Assoziationen zwischen Klassen modellieren mögliche **Objektbeziehungen** (Links) zwischen den **Instanzen der Klassen**



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



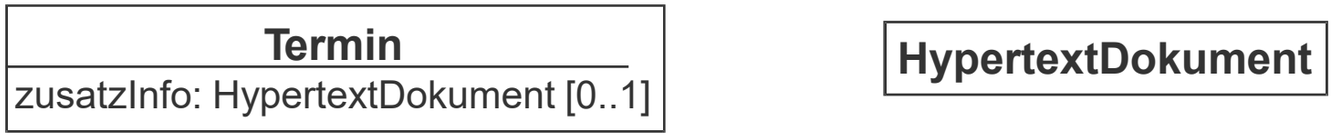
  - Navigation von einem bestimmten Termin zum entsprechenden Dokument
  - Umgekehrte Richtung - welche Termine beziehen sich auf ein bestimmtes Dokument? - wird nicht unterstützt

- **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

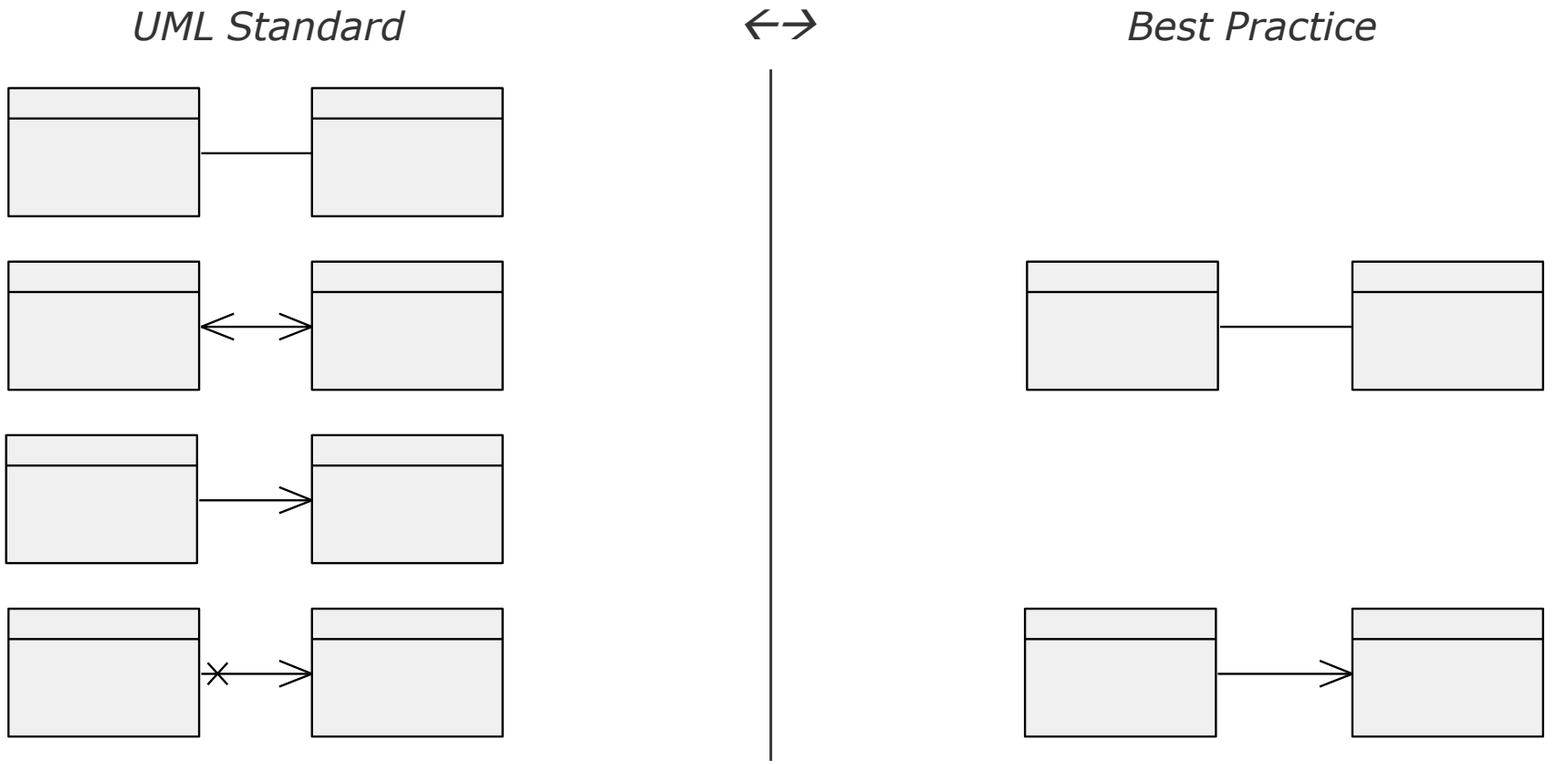
Assoziation als Attribut

- 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
- Alternative Darstellung des Beispiels der vorherigen Folie:



- **Nachteil: Schlechtere Lesbarkeit** → wird so in Vorlesung nicht weiter verfolgt

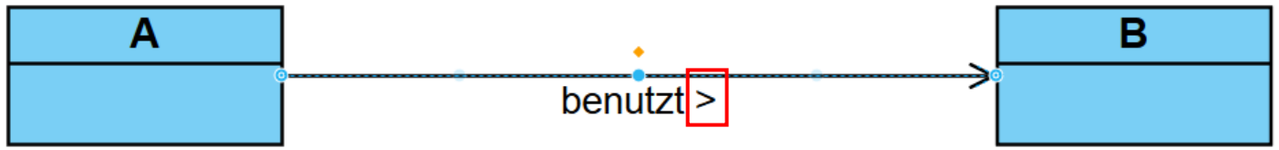
Assoziation – UML Standard vs. Best Practice

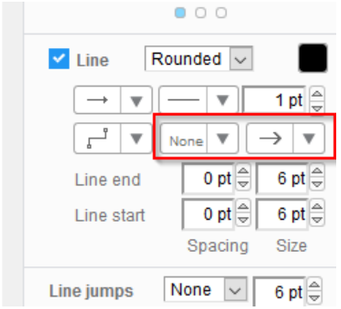


- I.d.R. Ihnen überlassen, für welche Form der Assoziationsnotation Sie sich entscheiden
- Nur wichtig: Geben Sie es an (Default Verständnis Dozent: Best Practice, ansonsten explizit angegeben)

Assoziation: Visual Paradigm Online

- Wie wenig in Praxis auf Navigations- und Leserichtung geachtet wird, wird an Visual Paradigm Online ersichtlich
- Funktionalität muss dort „ertrickst“ werden





Assoziation: Multiplizität

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

- Beispiele :

Notation

genau 1:

>= 0:

0 oder 1:

fixe Anzahl (z.B. 3):

Bereich (z.B. >= 3):

Bereich (z.B. 3 - 6):

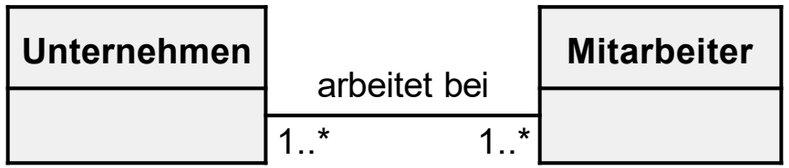
Aufzählung:

Beispiele: Assoziation und Multiplizität

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

Beispiele: Assoziation und Multiplizität

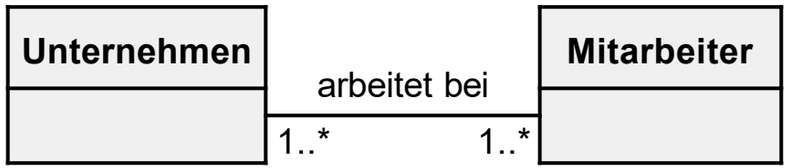
Ein Auto hat genau einen Besitzer, eine Person kann aber mehrere Autos besitzen (oder keines).



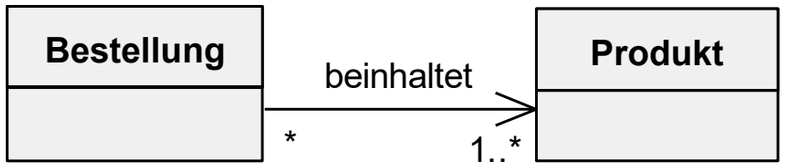
In einem Unternehmen arbeitet mind. ein Mitarbeiter, ein Mitarbeiter arbeitet mind. in einem Unternehmen

Beispiele: Assoziation und Multiplizität

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



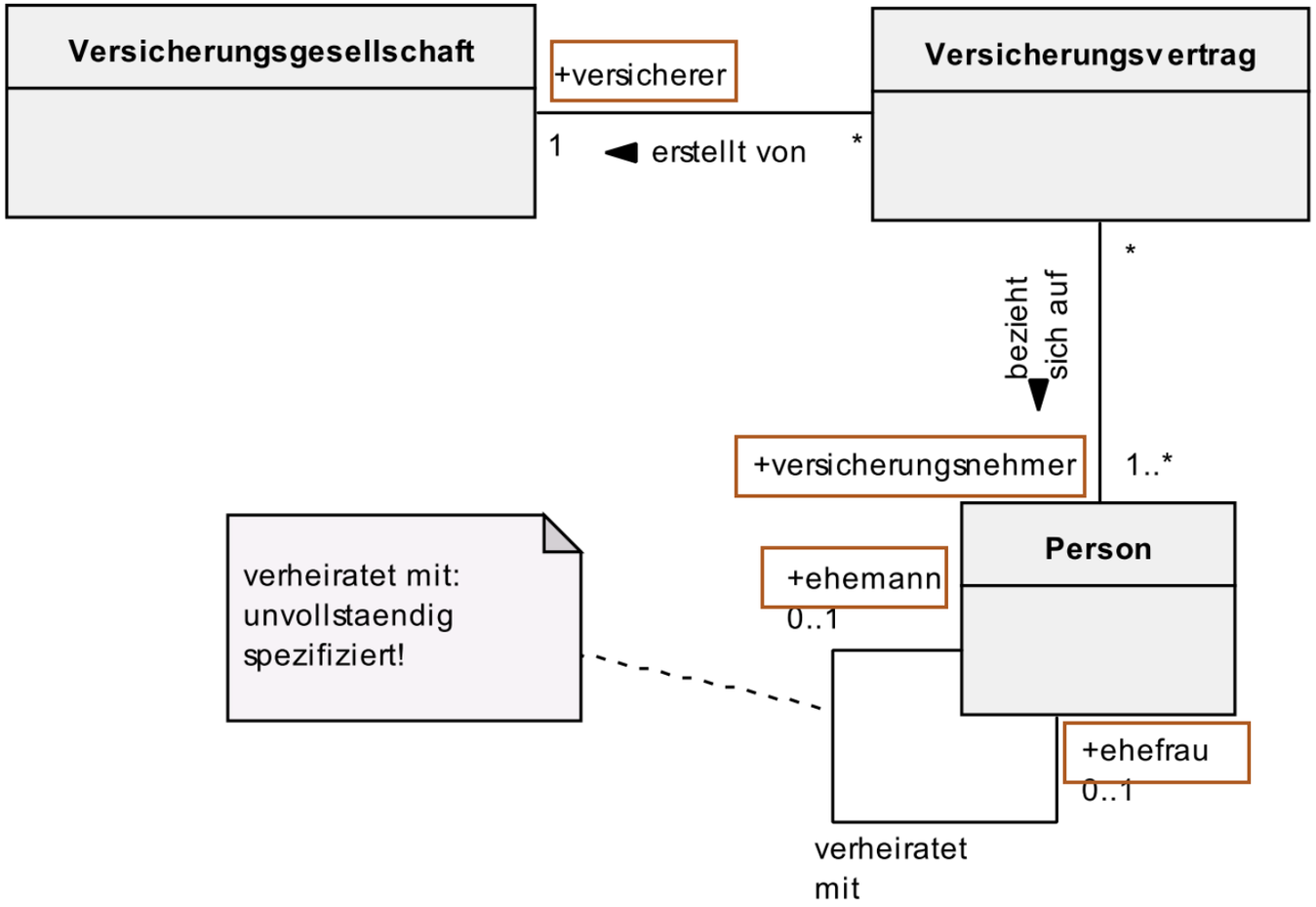
In einem Unternehmen arbeitet mind. ein Mitarbeiter, ein Mitarbeiter arbeitet mind. in einem Unternehmen



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

Assoziation: Rollen

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



Klassendiagramm

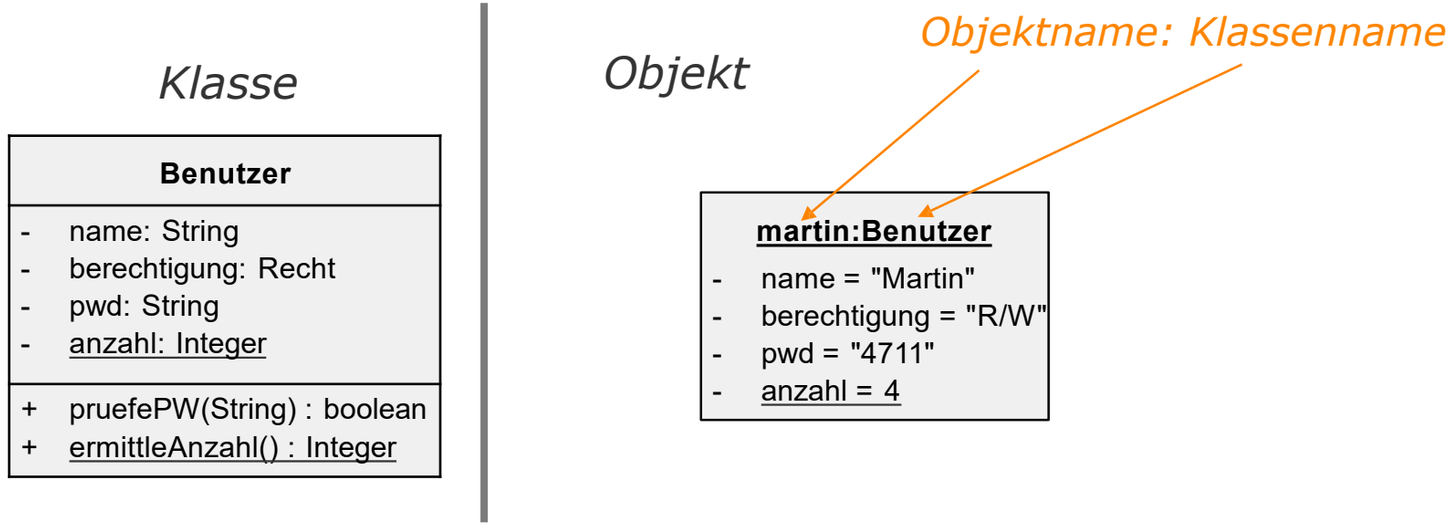
Agenda

- Einführung Klassendiagramm
- Klassen
- Attribute und Operationen und deren Identifikation
- Assoziationen: Grundlagen
- **Objekte als Instanzen von Klassen (Objektdiagramm)**
- Assoziationen: Vertiefung
- Aggregationen
- Generalisierung und Interfaces
- Datentypen
- Beispiel: Informationssystem Universitätsverwaltung
- Zusammenfassung Elemente / Rückblick und Ausblick

Objektdiagramm

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

- Beispiel:

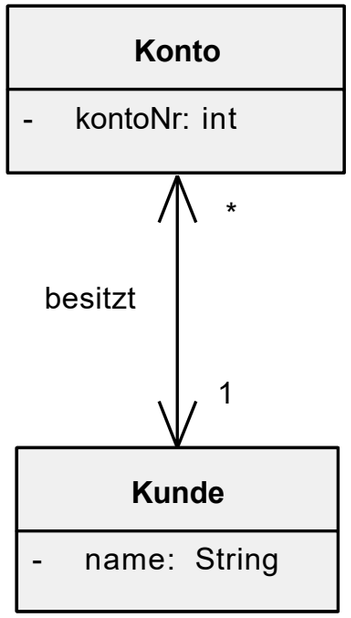


Objektdiagramm

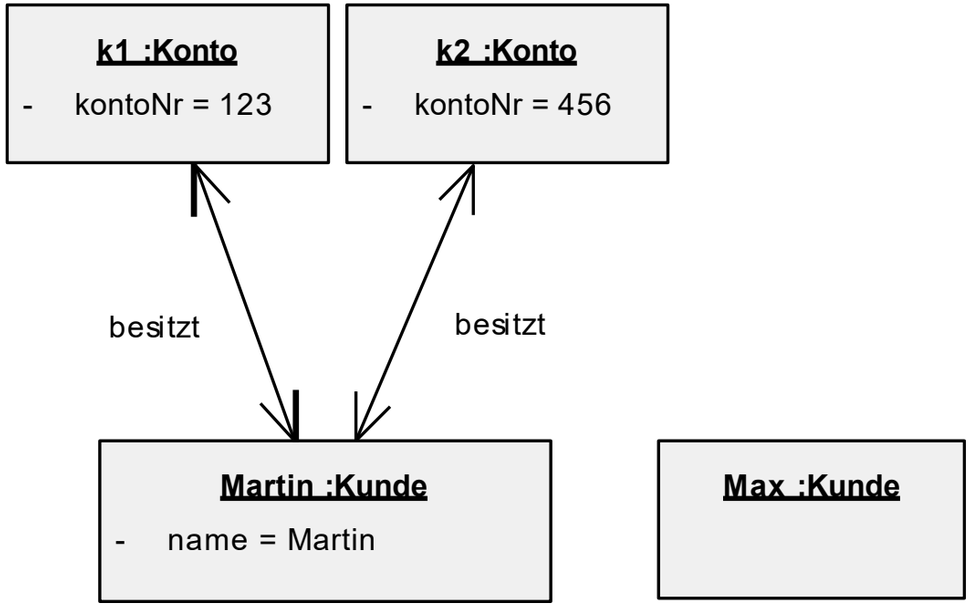
- **Basiskonzepte** des Objektdiagramms
  - Instanz einer Klasse: Objekt
  - Instanz einer Assoziation: Link
  - Instanz eines Datentyps: Wert
- **Einheitliche Notationskonventionen**
  - Gleiches Notationselement wie auf Typebene benutzen
  - Unterstreichen (bei Links optional)
- Objektdiagramm muss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

Beispiel: Objektdiagramm

Klassendiagramm



Objektdiagramm



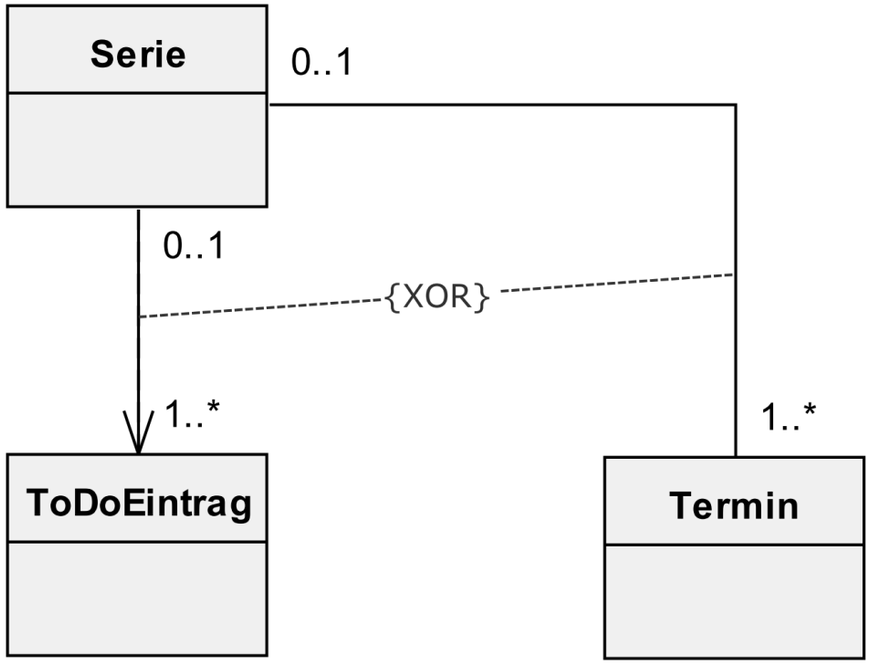
Klassendiagramm

Agenda

- Einführung Klassendiagramm
- Klassen
- Attribute und Operationen und deren Identifikation
- Assoziationen: Grundlagen
- Objekte als Instanzen von Klassen (Objektdiagramm)
- **Assoziationen: Vertiefung**
- Aggregationen
- Generalisierung und Interfaces
- Datentypen
- Beispiel: Informationssystem Universitätsverwaltung
- Zusammenfassung Elemente / Rückblick und Ausblick

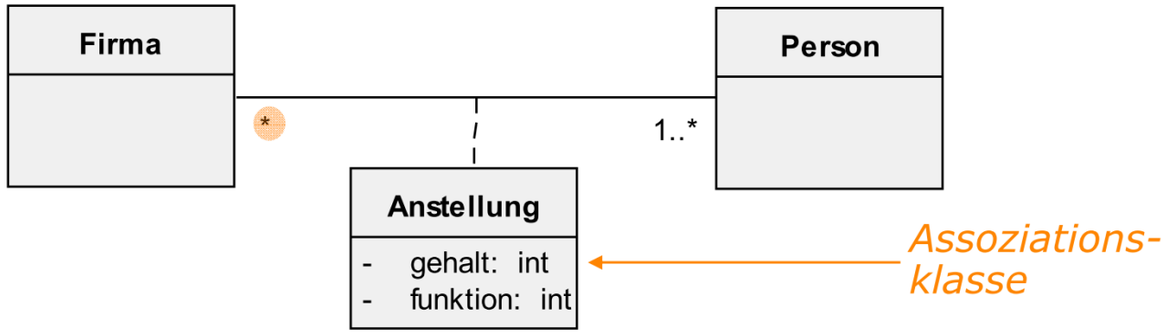
Exklusive Assoziation

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

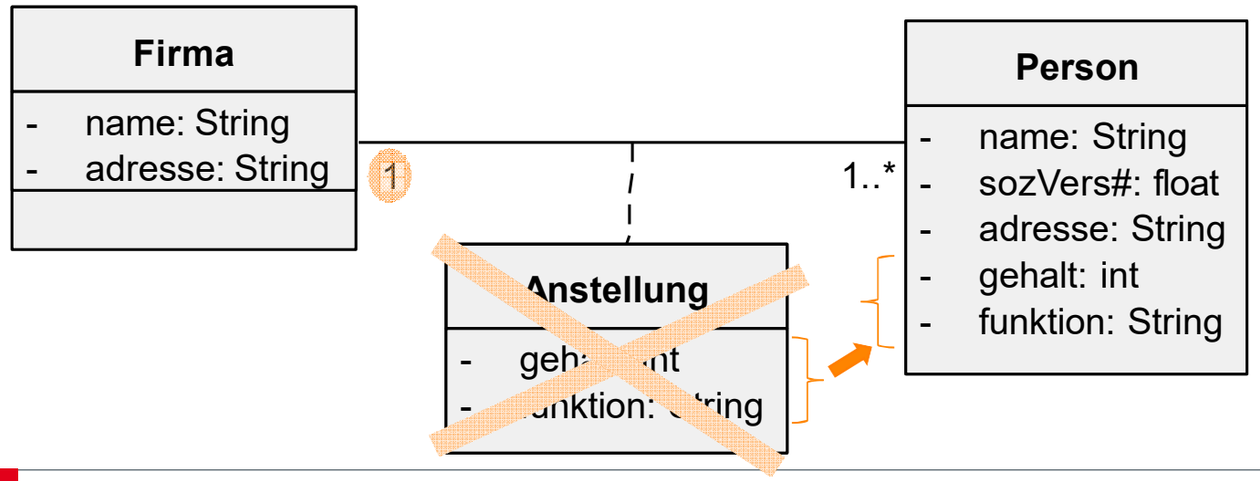


Assoziationsklasse (1/3)

- Kann Attribute der Assoziation enthalten → dient dazu, Merkmale der Beziehung von Klassen festzuhalten
  - Bei m:n-Assoziationen mit Attributen notwendig



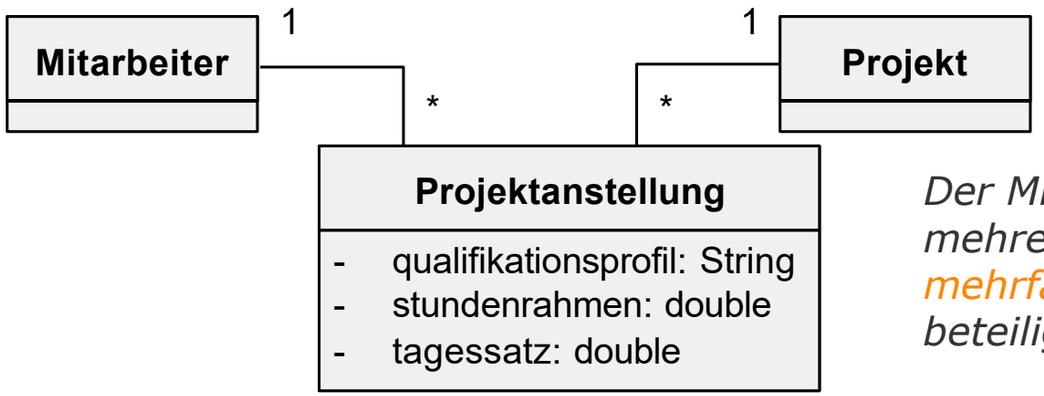
  - Bei 1:1 und 1:n-Assoziationen ggf. sinnvoll aus Flexibilitätsgründen (Änderung der Multiplizität möglich)



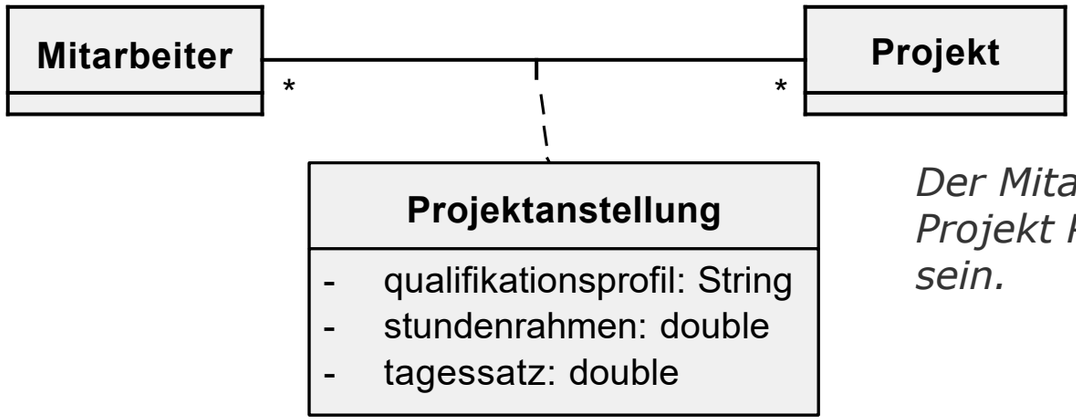
Notation:
Eine Assoziationsklasse wird durch eine Klasse und eine Assoziation dargestellt, die miteinander durch eine gestrichelte Linie verbunden sind.

Assoziationsklasse (2/3)

- Normale Klasse **ungleich** Assoziationsklasse



Der Mitarbeiter M kann über mehrere Projektanstellungen mehrfach an dem Projekt P beteiligt sein.

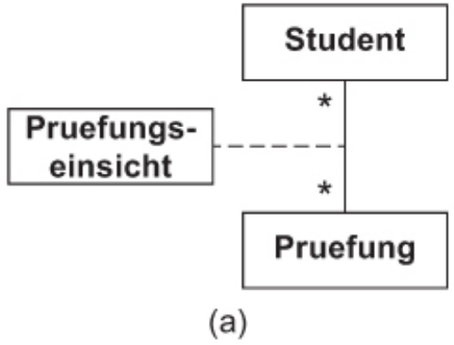


Der Mitarbeiter M kann an dem Projekt P nur einmal beteiligt sein.

Assoziationsklasse (3/3)

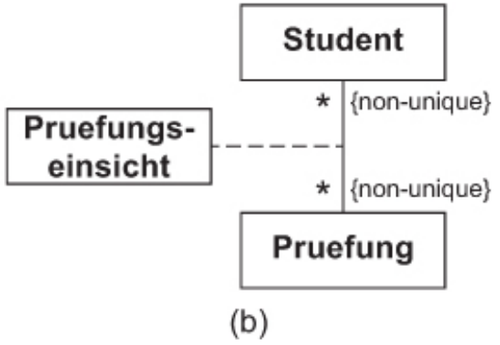
- Assoziationsklassen und Duplikate
  - Wenn Duplikate explizit gewünscht sind, müssen Assoziationsenden mit {non-unique} gekennzeichnet sein
  - Wenn keine Duplikate möglich sein sollen, gilt Defaultwert {unique} (keine explizite Kennzeichnung erfoderlich

Keine Duplikate möglich



(a)

Duplikate möglich



(b)

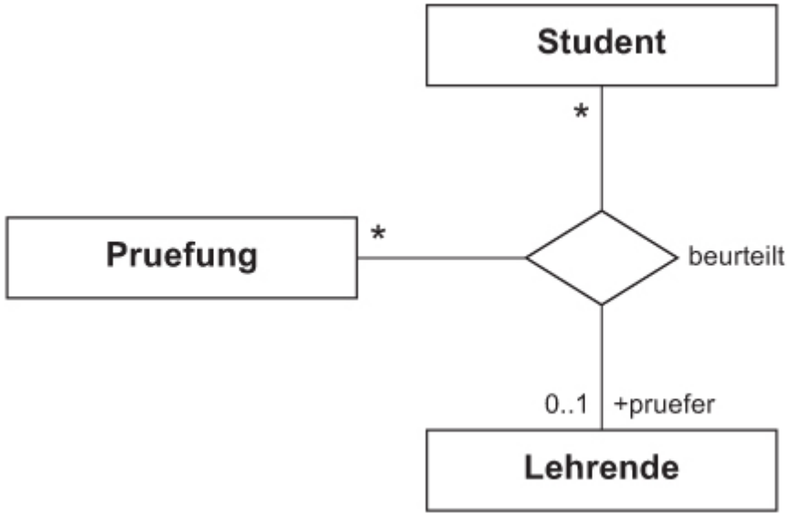
Wie oft kann ein Student für eine Prüfung Einsicht nehmen?

n-äre Assoziation (1/2)

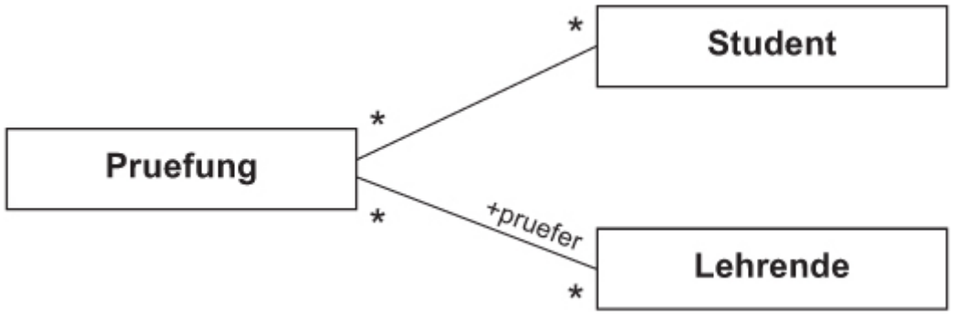
- **Beziehung zwischen mehr als zwei Klassen**
  - Navigationsrichtungen sind nicht möglich
  - Multiplizitäten und Rollenbezeichnungen sind weiterhin möglich
  - Multiplizitäten geben an, wie viele Objekte einer Rolle/Klasse einem festen (n-1)-Tupel von Objekten der anderen Rollen/Klassen zugeordnet sein können
- n-äre Assoziationen kommen in der Praxis in einem Domänen-Modell gelegentlich vor
- In der Implementierung verzichtet man aber meistens darauf und verwendet nur binäre Assoziationen

n-äre Assoziation (2/2) – Beispiel

- Festlegung der Multiplizitäten:
  - Ein bestimmter Student legt eine bestimmte Prüfung bei keinem (also gar nicht) oder bei genau einem Lehrenden ab → Multiplizität 0..1 bei Lehrende
  - Eine bestimmte Prüfung kann bei einem bestimmten Lehrenden von beliebig vielen Studenten absolviert werden. Ein bestimmter Student kann von einem Lehrenden beliebig oft beurteilt werden → Multiplizität *



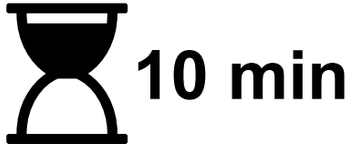
- Vergleich zu binären Assoziationen:
  - Im ternären Modell ist es nicht möglich, dass eine Prüfung von einem Studenten durch >1 Lehrende beurteilt wird.
  - Bei binären Assoziationen kann dies nicht ohne weiteres abgebildet werden



Beispiel: Assoziationen

- **Aufgabe:**
  - Modellieren Sie als Klassendiagramm, wie Personen Prüfungen entweder in Büros oder in Hörsälen durchführen und ablegen.
  - Eine Prüfung wird von einer Person durchgeführt. Eine Person kann beliebig viele Prüfungen durchführen.
  - Eine Prüfung kann von beliebig vielen Personen abgelegt werden. Eine Person kann beliebig viele Prüfungen ablegen.
  - Eine Person kann nicht gleichzeitig Prüfungen durchführen und ablegen.
  - Eine Prüfung kann nur entweder in einem Büro oder in einem Hörsaal, nicht aber gleichzeitig in beiden stattfinden.
  - In einem Büro oder einem Hörsaal können beliebig viele Prüfungen stattfinden.

- Erledigen Sie die Aufgabe in Kleingruppen
- Sie haben 10 Minuten Zeit
- Gemeinsame Ergebnisdiskussion



Klassendiagramm

Agenda

- Einführung Klassendiagramm
- Klassen
- Attribute und Operationen und deren Identifikation
- Assoziationen: Grundlagen
- Objekte als Instanzen von Klassen (Objektdiagramm)
- Assoziationen: Vertiefung
- **Aggregationen**
- Generalisierung und Interfaces
- Datentypen
- Beispiel: Informationssystem Universitätsverwaltung
- Zusammenfassung Elemente / Rückblick und Ausblick

Aggregation

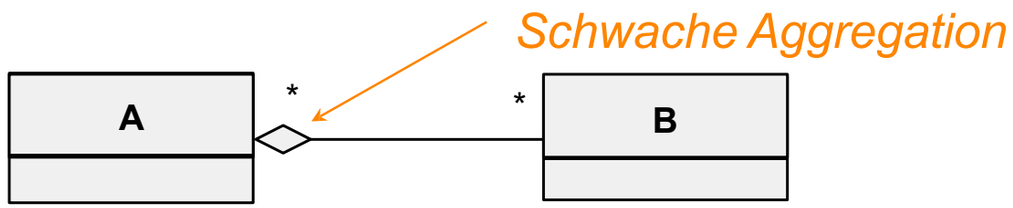
- Aggregation ist eine spezielle Form der Assoziation mit folgenden Eigenschaften:

- **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

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

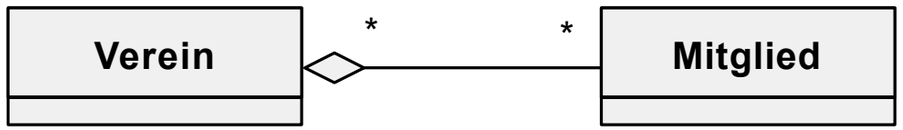
- UML unterscheidet zwei Arten von Aggregationen:
  - Schwache Aggregation (shared aggregation)
  - Starke Aggregation – Komposition (composite aggregation)

Schwache Aggregation

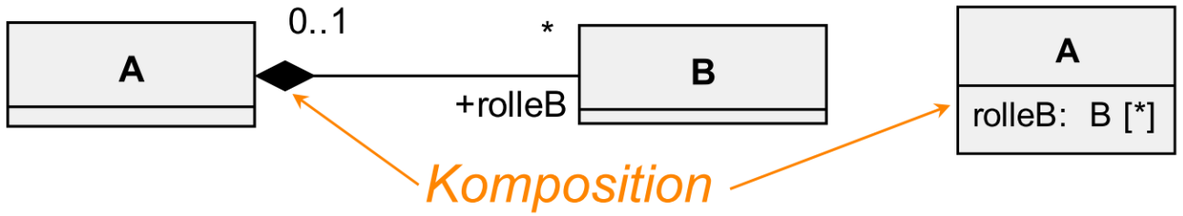


- Schwache Zugehörigkeit der Teile, d.h. Teile sind **unabhängig** von ihrem Ganzen
- Die Multiplizität des aggregierenden Endes der Beziehung (Raute) kann > 1 sein
- Es gilt nur eingeschränkte **Propagierungssemantik** → wird A gelöscht, muss B nicht automatisch auch gelöscht werden
- Die zusammengesetzten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

- Beispiel:

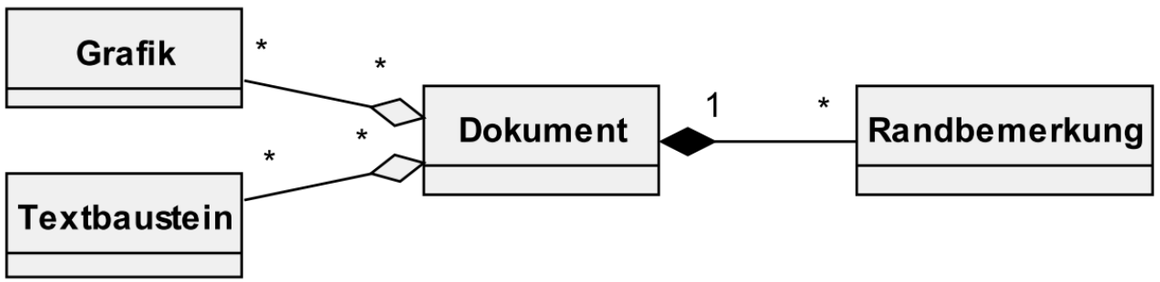


Starke Aggregation (= Komposition)



- 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 → wird A gelöscht, muss B (i.d.R.) gelöscht werden (in jedem Fall bei Multiplizität 1)
- Die zusammengesetzten Objekte bilden einen **Baum**

- Beispiel:



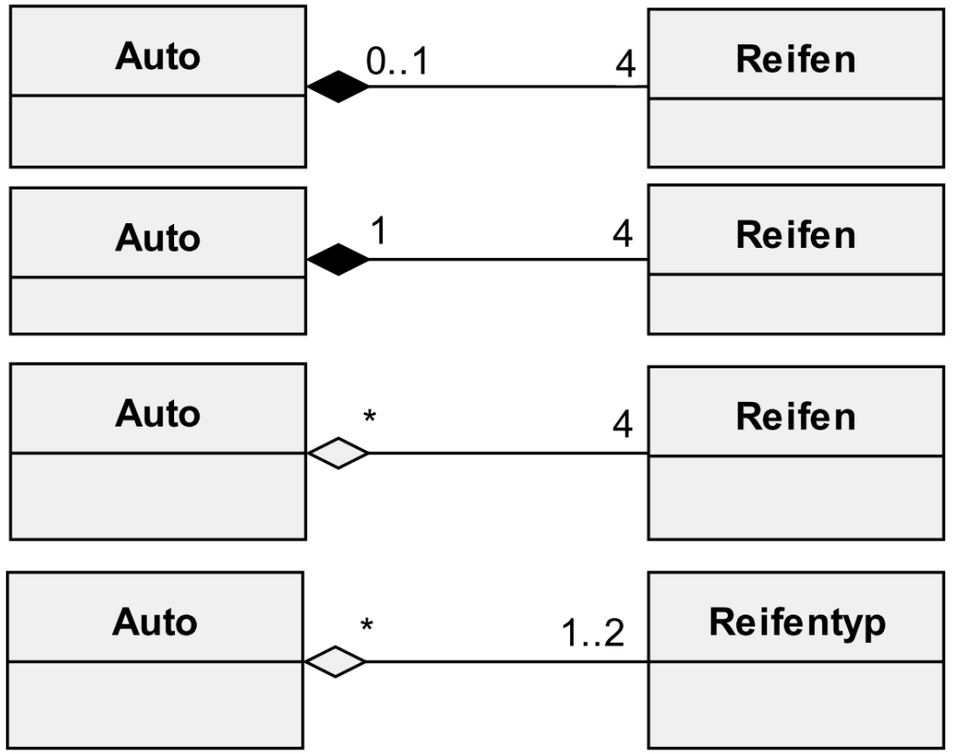
Starke Aggregation vs. Assoziation/schwache Aggregation - Faustregeln

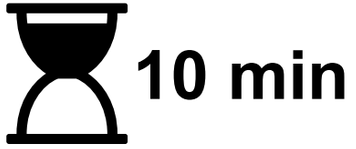
- **Einbettung**
  - Die Teile sind i.A. physisch im Kompositum enthalten
  - Über Assoziation verbundene Objekte werden über Referenzen realisiert
- **Sichtbarkeit**
  - Ein Teil ist nur für das Kompositum sichtbar
  - Das über eine Assoziation verbundene Objekt ist i.A. öffentlich sichtbar
- **Lebensdauer**
  - Das Kompositum erzeugt und löscht (i.d.R.) seine Teile (in jedem Fall bei Multiplizität 1)
  - Keine Existenzabhängigkeit zwischen assoziierten Objekten
- **Kopien**
  - Kompositum und Teile werden kopiert
  - Nur die Referenzen auf assoziierte Objekte werden kopiert

Beispiel: Komposition und Aggregation

- Welche der folgenden Beziehungen trifft zu?
  Welche trifft nicht zu?
  Begründen Sie Ihre Aussage

- Erledigen Sie die Aufgabe in Kleingruppen
- Sie haben 10 Minuten Zeit
- Gemeinsame Ergebnisdiskussion





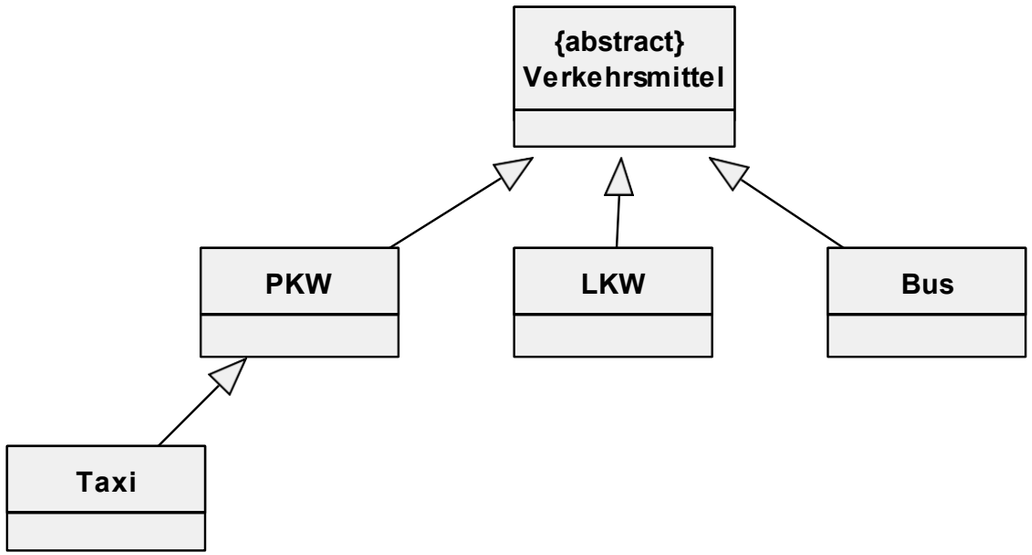
Klassendiagramm

Agenda

- Einführung Klassendiagramm
- Klassen
- Attribute und Operationen und deren Identifikation
- Assoziationen: Grundlagen
- Objekte als Instanzen von Klassen (Objektdiagramm)
- Assoziationen: Vertiefung
- Aggregationen
- **Generalisierung** und Interfaces
- Datentypen
- Beispiel: Informationssystem Universitätsverwaltung
- Zusammenfassung Elemente / Rückblick und Ausblick

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 (1/2)

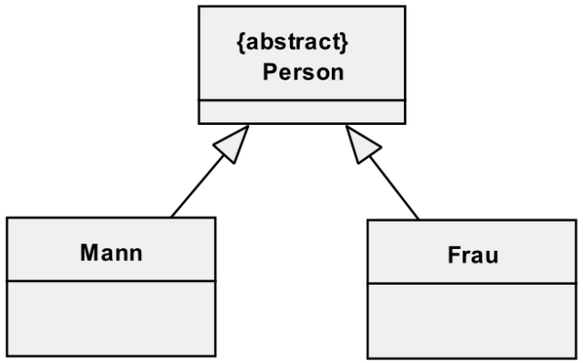
- Klasse, die **nicht instanziiert** werden kann
- **Nur in Generalisierungshierarchien** sinnvoll
- Dient zum **"Herausheben" gemeinsamer Merkmale** einer Reihe von Unterklassen

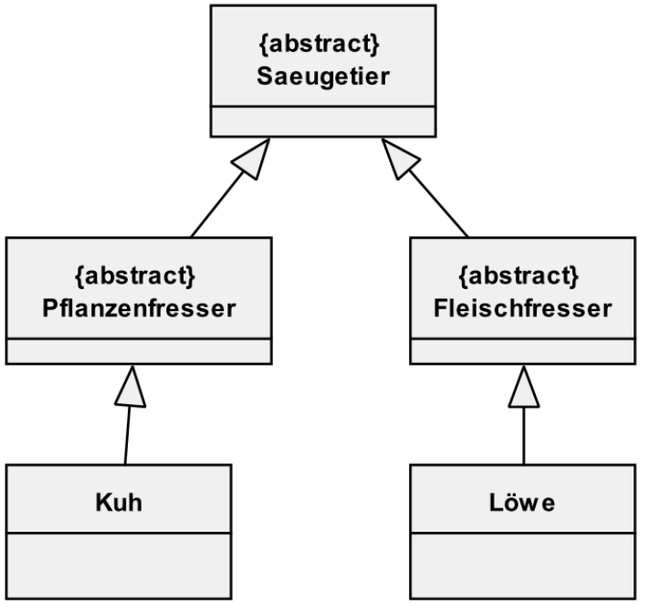
- Notation: Schlüsselwort {abstract} oder Klassenname in kursiver Schrift

- Mit analoger Notation wird zwischen konkreten (= implementierten) und abstrakten (= nur spezifizierten) **Operationen** einer Klasse unterschieden

Abstrakte Klasse (2/2)

- Beispiele

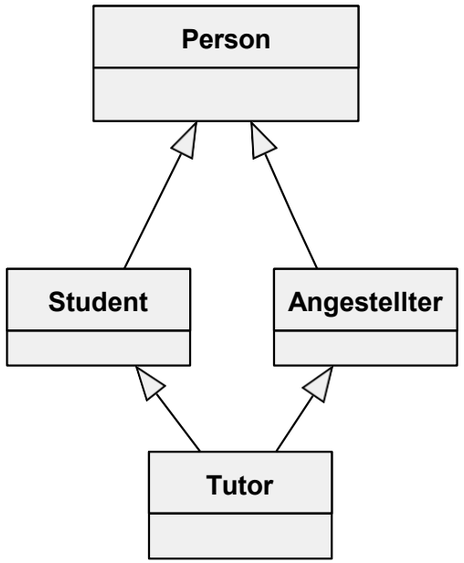




Mehrfachvererbung

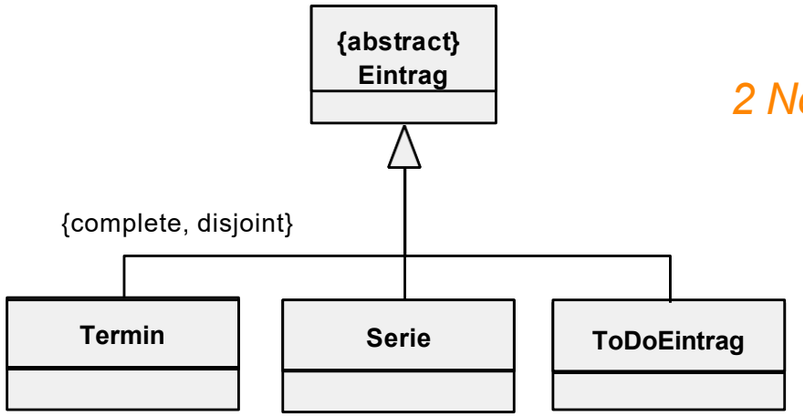
- Klassen müssen nicht nur eine Oberklasse haben, sondern können auch **von mehreren Klassen erben**
  - Falls Programmiersprache zur Umsetzung bereits feststeht, sollte diese zur Verwendung sinnvollerweise ebenfalls Mehrfachvererbung unterstützen

- Beispiel:



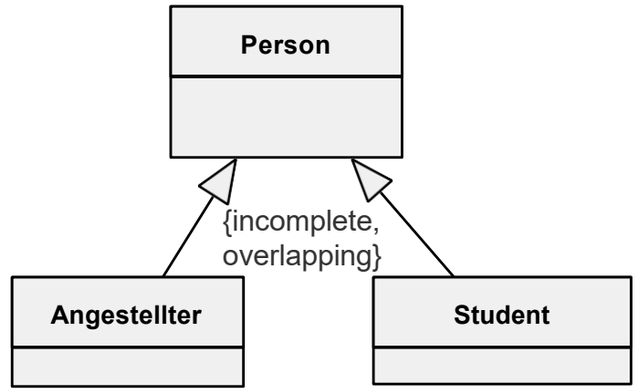
Generalisierung: Eigenschaften

- Unterscheidung kann vorgenommen werden in
  - **Unvollständig oder vollständig:**
    - In einer vollständigen Generalisierungshierarchie muss jede Instanz der Superklasse auch Instanz mindestens einer Subklasse sein
    - In einer unvollständigen Generalisierungshierarchie muss das nicht der Fall sein
  - **Überlappend oder disjunkt:**
    - In einer überlappenden Generalisierungshierarchie kann ein Objekt Instanz von mehr als einer Subklasse sein
    - In einer disjunkten Generalisierungshierarchie muss das nicht der Fall sein
  - **Default:** unvollständig, disjunkt



2 Notationsmöglichkeiten bei >1 Pfeil

„Is-a“ Beziehung



Klassendiagramm

Agenda

- Einführung Klassendiagramm
- Klassen
- Attribute und Operationen und deren Identifikation
- Assoziationen: Grundlagen
- Objekte als Instanzen von Klassen (Objektdiagramm)
- Assoziationen: Vertiefung
- Aggregationen
- Generalisierung **und Interfaces**
- Datentypen
- Beispiel: Informationssystem Universitätsverwaltung
- Zusammenfassung Elemente / Rückblick und Ausblick

Interface

- Ein Interface spezifiziert **gewünschtes Verhalten** durch **Zusammenfassung der Operationen**
  - einer Klasse
  - einer Komponente
  - eines Pakets
- und kann auch Attribute aufweisen

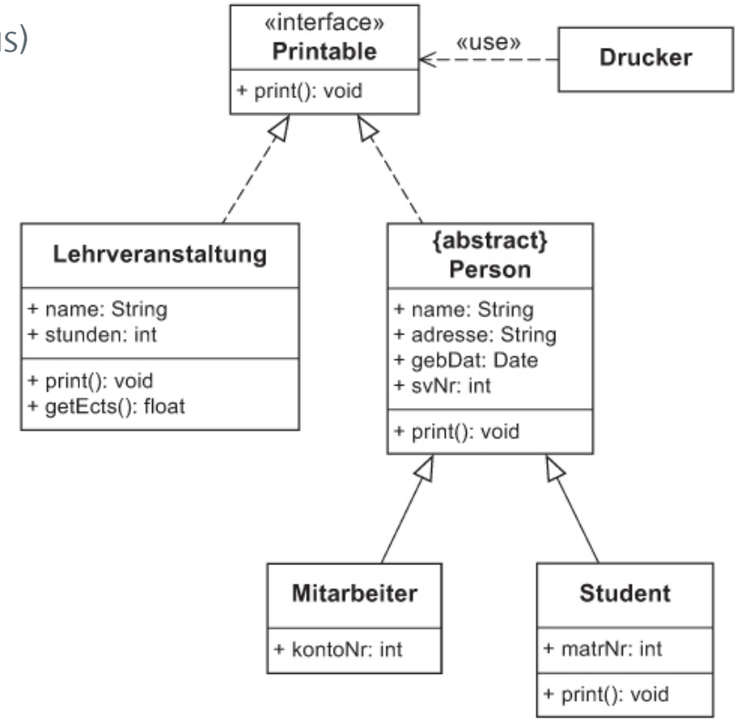
- **Unterschied zur abstrakten Klasse**
  - Abstrakte Klasse kann nur durch Subklassen realisiert werden, Interfaces durch beliebige Klassen

- Klassen, die ein **Interface realisieren - Anbieter -** können noch zusätzliche Operationen aufweisen
- Klassen, die ein **Interface benutzen - Klienten -** müssen nicht alle angebotenen Operationen tatsächlich nutzen

Interface: Beispiel

- Die Klassen Person/Lehrveranstaltung implementieren Schnittstelle Printable
  - Sie **müssen Operation print() zur Verfügung** stellen (sieht für jede Klasse anders aus)
    - Lehrveranstaltung: Druck von Name und Stundenanzahl
    - Person: Druck von Name und Adresse
- Klasse Student **erweitert geerbte Operation** print() von Person
  - Methode wird überschrieben, es wird zusätzlich Matrikelnummer mit gedruckt
- Klasse Mitarbeiter verwendet print() von Person ohne Erweiterung

- Klasse Drucker kann **jede Klasse verarbeiten**, die die Schnittstelle Printable implementiert
  - Für jede Klasse spezifische Operation print() realisierbar, **ohne** dass Klasse **Drucker verändert** werden muss



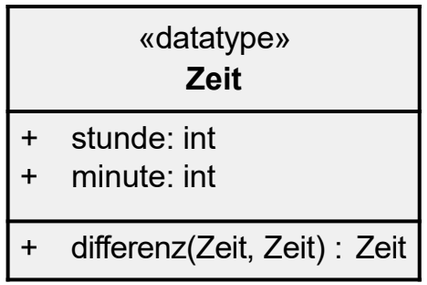
Klassendiagramm

Agenda

- Einführung Klassendiagramm
- Klassen
- Attribute und Operationen und deren Identifikation
- Assoziationen: Grundlagen
- Objekte als Instanzen von Klassen (Objektdiagramm)
- Assoziationen: Vertiefung
- Aggregationen
- Generalisierung und Interfaces
- **Datentypen**
- Beispiel: Informationssystem Universitätsverwaltung
- Zusammenfassung Elemente / Rückblick und Ausblick

Datentypen in UML

- Datentypen können wie Klassen Attribute und Operationen haben
- Aber: Instanzen eines Datentyps haben keine Identität
  - **Objekte**: Instanzen einer Klasse
  - **Werte**: Instanzen eines Datentyps (z.B. Zahl 2)
- Notation: Klassensymbol mit Schlüsselwort «datatype»
- Beispiel:



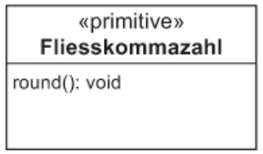
- Arten von Datentypen:
  - Primitive Datentypen
  - Aufzählungstypen

Arten von Datentypen: Primitive Datentypen

- Primitive Datentypen: Datentypen ohne innere Struktur

- Von UML vordefinierte primitive Datentypen:
  - Boolean
  - Integer
  - UnlimitedNatural
  - String

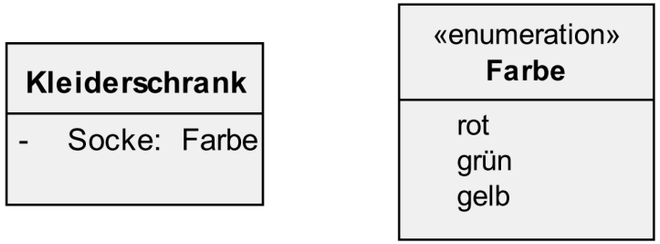
- Primitive Datentypen können auch selbst definiert werden:
  - wie beliebige Datentypen, nur mit dem Schlüsselwort «primitive»
- Primitive Datentypen können ebenfalls Operationen haben



Arten von Datentypen: Aufzählungstypen

- Festlegung der Ausprägungsmenge per Aufzählung
- Notation: Klassensymbol mit Schlüsselwort «enumeration»
- Aufzählungstypen können Attribute und Operationen haben
- Mögliche Ausprägungen werden durch benutzerdefinierte Bezeichner (Literale) angegeben

- Beispiel:



Klassendiagramm

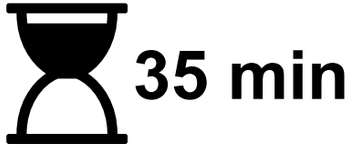
Agenda

- Einführung Klassendiagramm
- Klassen
- Attribute und Operationen und deren Identifikation
- Assoziationen: Grundlagen
- Objekte als Instanzen von Klassen (Objektdiagramm)
- Assoziationen: Vertiefung
- Aggregationen
- Generalisierung und Interfaces
- Datentypen
- **Beispiel: Informationssystem Universitätsverwaltung**
- Zusammenfassung Elemente / Rückblick und Ausblick

Beispiel: Informationssystem Universitätsverwaltung

- **Ziel der Übung:** Modellieren Sie ein Klassendiagramm für ein vereinfacht dargestelltes Universitäts-Informationssystem entsprechend der vorherigen verbalen Spezifikation

- **Vorgehensweise**
  - Erledigen Sie die Aufgabe in Kleingruppen
  - Sie haben 35 Minuten Zeit
  - Gemeinsame Ergebnisdiskussion
- **Stellen Sie sich z.B. folgende Fragen:**
  - Welche Klassen können Sie identifizieren?
  - Welche Attribute können Sie für diese Klassen identifizieren?
  - Welche Generalisierungsbeziehungen sind vorhanden?
  - Welche Assoziationsbeziehungen sind vorhanden? Gibt es Assoziationsklassen?
  - Welche Aggregationsbeziehungen existieren?



Beispiel: Informationssystem Universitätsverwaltung

Gesucht ist ein vereinfacht dargestelltes Modell für ein Universitäts-Informationssystem entsprechend der folgenden verbalen Spezifikation.

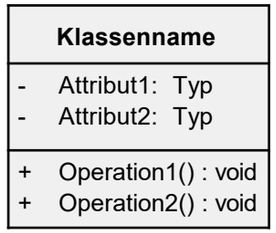
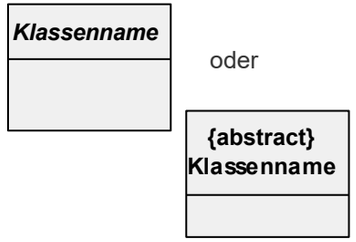
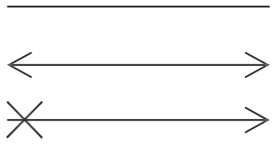
- Die Universität besteht aus mehreren Fakultäten, die sich wiederum aus verschiedenen Instituten zusammensetzen. Jede Fakultät und jedes Institut besitzt eine Bezeichnung. Für jedes Institut ist eine Adresse bekannt. Jede Fakultät wird von ihrem Dekan, einem Mitarbeiter, geleitet.
- Die Gesamtanzahl 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.
- Wissenschaftliche Mitarbeiter können 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 wissenschaftlichen Mitarbeiter werden als Vortragende bezeichnet. LVAs haben eine ID, einen Namen und eine Stundenanzahl.

Klassendiagramm

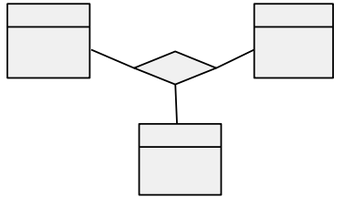
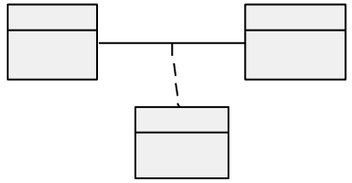
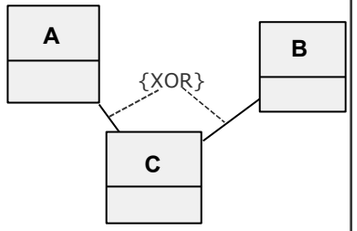
Agenda

- Einführung Klassendiagramm
- Klassen
- Attribute (Unterschied zur Datenmodellierung) und Operationen und deren Identifikation
- Assoziationen: Grundlagen
- Objekte als Instanzen von Klassen (Objektdiagramm)
- Assoziationen: Vertiefung
- Aggregationen
- Generalisierung und Interfaces
- Datentypen
- Beispiel: Informationssystem Universitätsverwaltung
- **Zusammenfassung Elemente / Rückblick**

Klassendiagramm - Elemente (1/3)

Name	Syntax	Beschreibung
Klasse		Beschreibung der Struktur und des Verhaltens einer Menge von Objekten
abstrakte Klasse		Klasse, die nicht instanziert werden kann
Assoziation		Beziehung zwischen Klassen: keine Angabe über Nav.-r.; mit Navigationsrichtung; in eine Richtung nicht navigierbar.

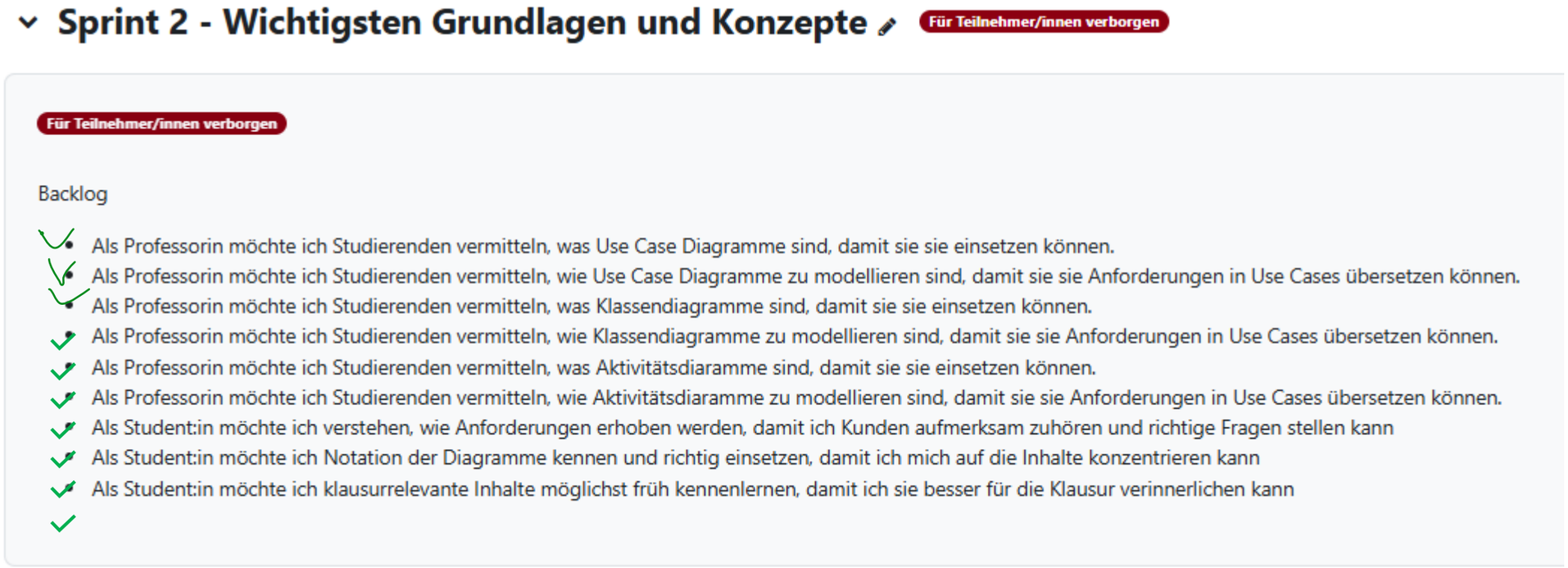
Klassendiagramm - Elemente (2/3)

Name	Syntax	Beschreibung
n-äre Assoziation		Beziehung zwischen n Klassen
Assoziations-klasse		nähere Beschreibung einer Assoziation
xor-Beziehung		Entweder steht Klasse A oder Klasse B in Beziehung zu C, nicht aber beide

Klassendiagramm - Elemente (3/3)

Name	Syntax	Beschreibung
schwache Aggregation		"Teil-Ganzes"-Beziehung
starke Aggregation = Komposition		exklusive "Teil-Ganzes"-Beziehung
Generalisierung		Vererbungsbeziehung zwischen Klassen

Willkommen im Sprint 2



- was der Unterschied zwischen einer Klasse und einem Objekt ist.
- was der Unterschied zwischen abstrakten und konkreten Klassen ist.
- was Attribute und Operationen einer Klasse sind und welche Eigenschaften diese haben können.
- dass Klassen durch Assoziationen miteinander verbunden werden und warum Assoziationen mit einer Multiplizität versehen werden.
- was Generalisierung ist und wann diese eingesetzt wird.
- was der Unterschied zwischen starker und schwacher Aggregation ist.
- was Interfaces sind.

Ansicht der Veranstaltung im Moodle