Sprint zur Lektion 1 und 2:

Das objektorientierte Konzept

Modul „Grundlagen der objektorientierten Programmierung mit Java“

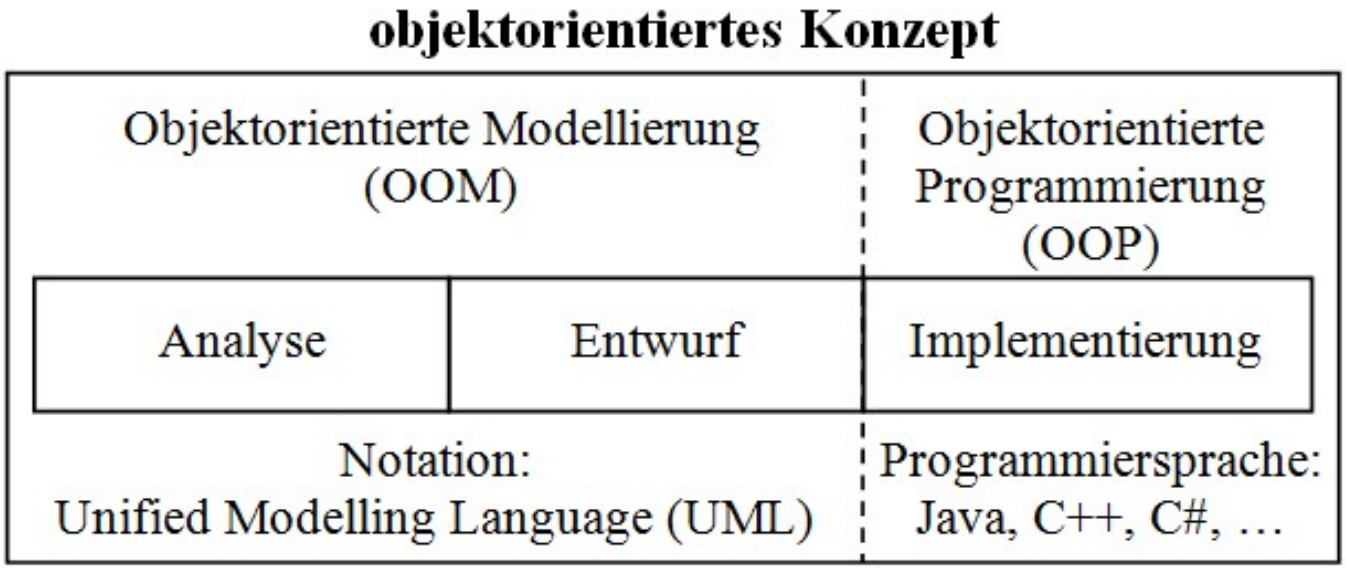
Prof. Dr. Cornelia Heinisch

Agenda

- Die Denkweise in Objekten und Klassen
- Einführung in die UML
- Übungsaufgabe

Das objektorientierte Konzept

- alternative Begriffe:
  - objektorientierter Ansatz,
  - objektorientiertes Paradigma,
  - Objektorientierung
- in Beziehung stehende Begriffe:



Das Denken in Objekten und Klassen

- ist von zentraler Bedeutung für das objektorientierte Konzept
- begleitet alle Phasen der SW-Entwicklung
- beginnt mit dem ersten Nachdenken über ein mögliches SW-System

Möchte man Aufgaben, die bisher manuell erledigt wurden, durch den Einsatz eines EDV-Systems unterstützen, so stellt man sich folgende Fragen:


Mit welchen Objekten hat man es beim Erledigen der Aufgaben zu tun?


Welche Eigenschaften dieser Objekte sind im Rahmen der zu erledigenden Aufgaben von Bedeutung?


Was wird mit den Objekten gemacht?

Beispielsystem Bibliothek

- wird bisher durch Karteikarten,
- Etiketten an Büchern und Regalen,
- sowie durch manuell gepflegte Listen verwaltet

Mit welchen Objekten hat man es beim Erledigen der Aufgaben zu tun?

Finden der relevanten Objekte und Klassen!

Beispielsystem Bibliothek: Objekte finden durch Beobachten

- viele „**Buch-Objekte**“
- werden ausgeliehen,
- werden zurückgegeben,
- …

- Personen, die Bücher mitnehmen und zurückbringen
- besitzen einen Bibliotheksausweis
- sind in der Bibliothek registriert
- → „**Ausleiher-Objekte**“

Neben existenten Gegenständen in der realen Welt (wie z. B. Bücher) können auch Wesen (wie z. B. Ausleiher) oder Konzepte (wie z. B. Versicherungs-verträge) relevante Objekte darstellen.

Beispielsystem Bibliothek: Klassen finden

Gleichartige Objekte werden zu einer Klasse zusammengefasst:

- viele „**Buch-Objekte**“
- → Abbildung durch die Klasse `Buch`

- viele „**Ausleiher-Objekte**“
- → Abbildung durch die Klasse `Ausleiher`

Klassen stellen die Baupläne für Objekte dar. Die **Klassen** sind die **Datentypen**, die **Objekte** die **Variablen** (Instanzen) dieser Datentypen. Ein Objekt wird gemäß dem Bauplan einer Klasse erzeugt.

Beispielsystem Bibliothek: Eigenschaften finden

Welche Eigenschaften dieser Objekte sind im Rahmen der zu erledigenden Aufgaben von Bedeutung?

Finden der relevanten Datenfelder (Eigenschaften, Attribute) der Klassen!

„Buch-Objekte“

Klasse `Buch`

- Buchnummer
- ISBN
- Titel
- Autor
- Auflage

„Ausleiher-Objekte“

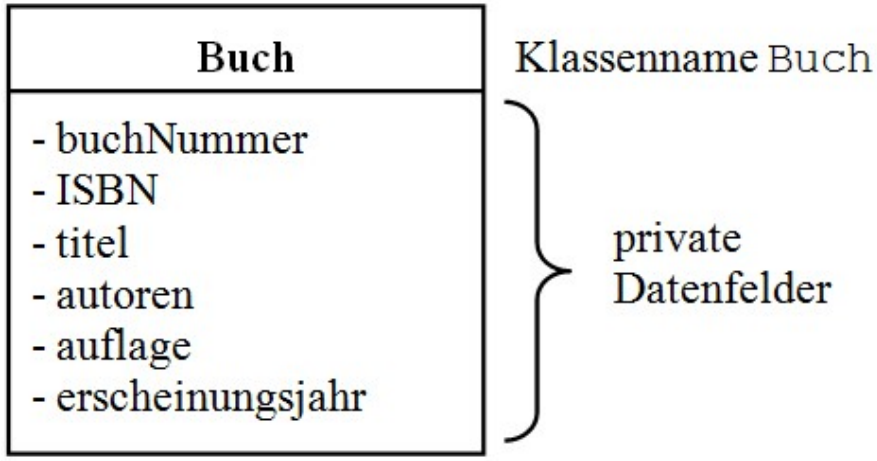
Klasse `Ausleiher`

- Ausleihernummer
- Name
- Vorname
- Anschrift

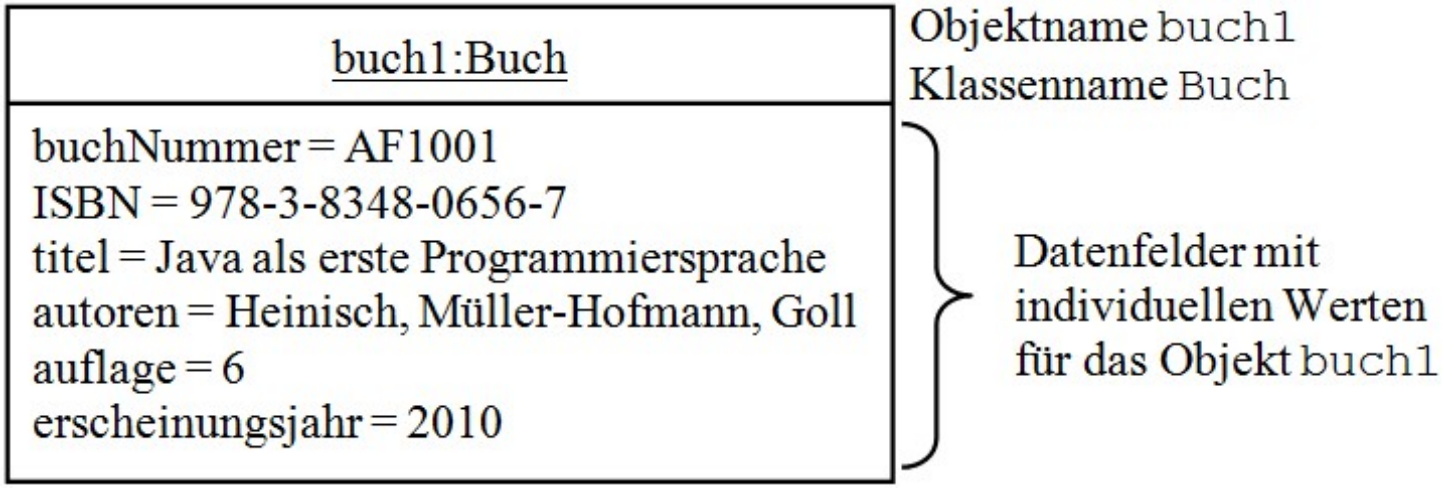
Beispielsystem Bibliothek: UML-Notation

„Buch-Objekte“

UML-Notation der Klasse `Buch`



UML-Notation für ein Objekt der Klasse `Buch`



Beispielsystem Bibliothek: Methoden finden

Was kann man mit den Buch-Objekten in der Bibliothek machen?
Was machen die Ausleiher-Objekte in der Bibliothek?

Finden der relevanten Methoden für die Klassen!

Beobachtbare Vorgänge in der Bibliothek:

- Ein Buch wird von einem Ausleiher ausgeliehen.
- Ein Buch wird von einem Ausleiher zurückgegeben.
- Ein Buch wird in die Bibliothek aufgenommen.
- Ein Buch wird durch den Bibliothekar in ein Regal gestellt.
- Ein Buch wird durch den Ausleiher aus einem Regal geholt.
- …

Welche der beobachtbaren Vorgänge bzw. welche Anteile sollen automatisiert werden?

Beispielsystem Bibliothek: Identifikation Anwendungsfälle

- Durch Beobachten der Abläufe werden die **Geschäftsprozesse** identifiziert.
- Für die spätere Programmierung relevant sind die so genannten **Anwendungsfälle**.

Die Anwendungsfälle werden gefunden, indem man die Geschäftsprozesse genau analysiert und festlegt, welche Teile durch das System automatisiert werden sollen.

- Der Geschäftsprozess „Buch ausleihen“, lässt sich wie folgt zerlegen:
  - Buchrecherche durchführen,
  - Buchverfügbarkeit prüfen,
  - Buch aus Regal holen,
  - Ausleiher identifizieren,
  - Buch für Ausleiher als entliehen buchen.

Software-Unterstützung naheliegend
→ Kandidaten für Anwendungsfälle

- Für die Anwendungsfälle ist zu prüfen, welche Objekte mit welchen Methoden hier unterstützen können.

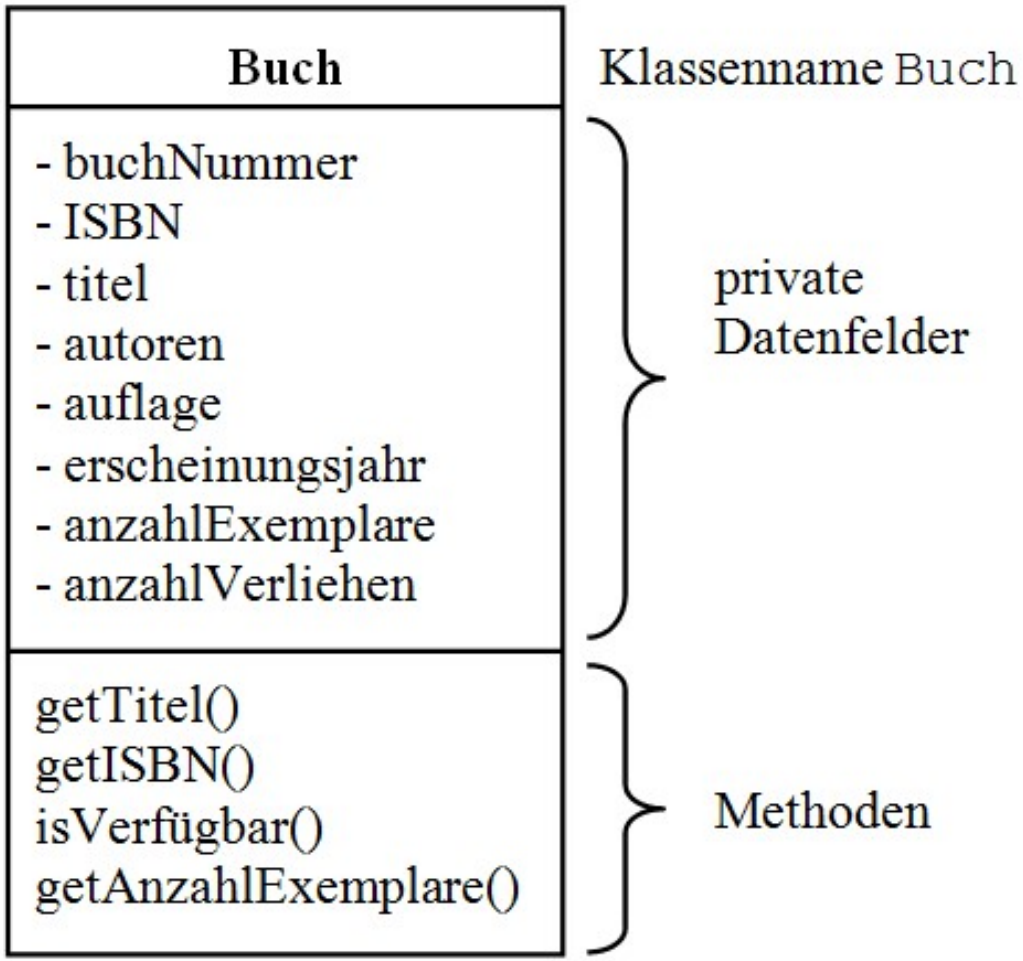
Beispielsystem Bibliothek: Methoden für die Klasse Buch

Wie kann ein Buch-Objekt die Anwendungsfälle unterstützen?

- Anwendungsfall „Buch-Recherche durchführen“
  - Ein einzelnes Buch-Objekt kann keine „Recherche durchführen“.
  - Ein einzelnes Buch-Objekt kann aber seine Daten für die Recherche bereitstellen:
    - `getTitel()`
    - `getISBN()`
- Anwendungsfall „Buchverfügbarkeit prüfen“
  - Ein Buch-Objekt muss wissen, wie viele Exemplare in der Bibliothek sind.
    → neues Datenfeld `anzahlExemplare`
  - Ein Buch-Objekt ermöglicht die Abfrage der vorhandenen Exemplare über eine Methode `getAnzahlExemplare()`
  - …

Beispielsystem Bibliothek: UML-Notation der Klasse mit Methoden

UML-Notation der Klasse Buch nach einigen Überlegungen:



Klassenname Buch

private Datenfelder

Methoden

Objektorientierte Modellierung und Programmierung

- Kernaufgabe der Objektorientierten Modellierung
  - systematisches Analysieren von Geschäftsprozessen und Anwendungsfällen,
  - Finden benötigter Klassen,
  - Finden von Eigenschaften und Methoden von Klassen.

- Kernaufgabe der Objektorientierten Programmierung

  Übersetzen der Modelle aus der Objektorientierten Modellierung in eine Objektorientierte Programmiersprache.

Um gute objektorientierte Programme zu schreiben, genügt es nicht, eine objektorientierte Programmiersprache zu beherrschen, ausschlaggebend ist, dass zuvor das richtige Modell erstellt wird.

Agenda

- Die Denkweise in Objekten und Klassen
- Einführung in die UML
- Übungsaufgabe

Was ist UML?

Definition:

UML ist eine **standardisierte Notationssprache** zur Beschreibung der **objektorientierten Modellierung** von Systemen.

Vorsicht:

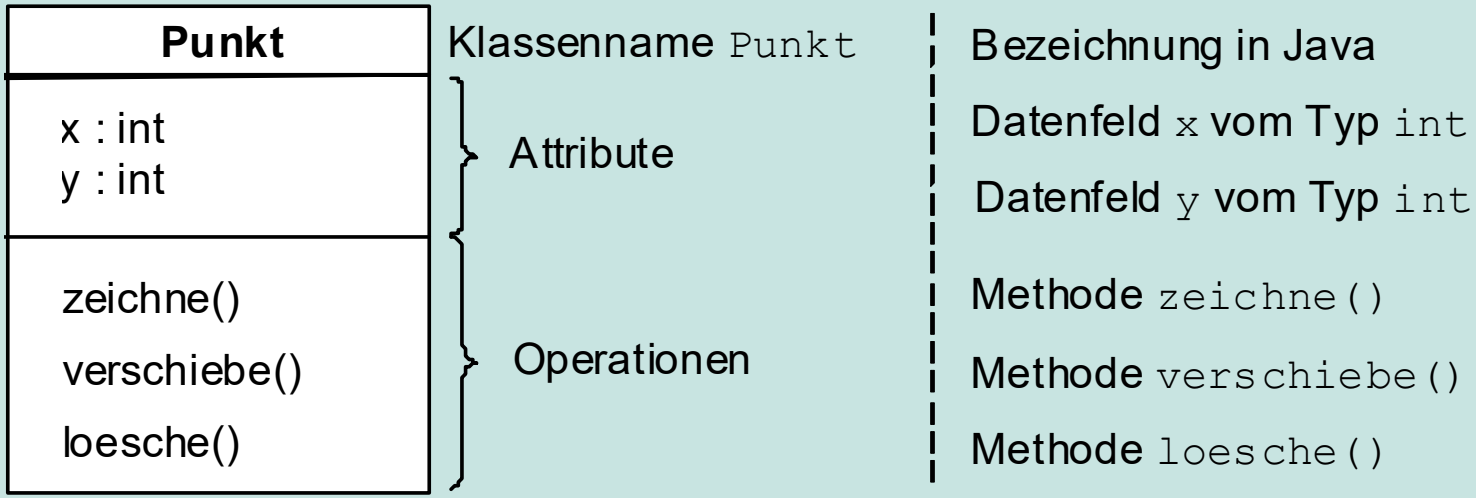
Das objektorientierte Konzept und die objektorientierte Methodik beruhen nicht auf UML, sondern können mit Hilfe von UML notiert werden.

Ziel der UML ist es,

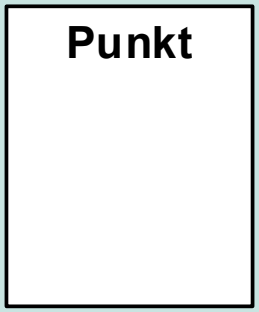
die **Analyse** und den **Entwurf** von komplexen, **objektorientierten Software-Systemen standardisiert** und **leicht verständlich zu dokumentieren**.

UML-Grundlagen – Darstellung Klasse

Klasse mit Attributen und Operationen:

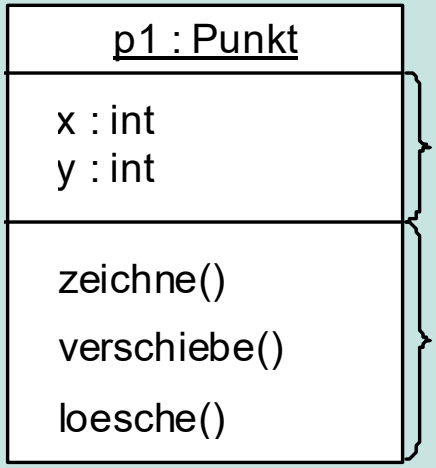


Klasse ohne Attribute und Operationen:



UML-Grundlagen – Darstellung Objekt

Konkretes Objekt `p1` der Klasse `Punkt`:



Objektname `p1`
Klassenname `Punkt`

Attribute

Operationen

`p1` ist der Name der Referenzvariablen

Datenfeld `x`

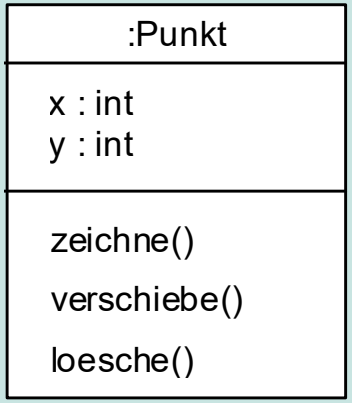
Datenfeld `y`

Methode `zeichne()`

Methode `verschiebe()`

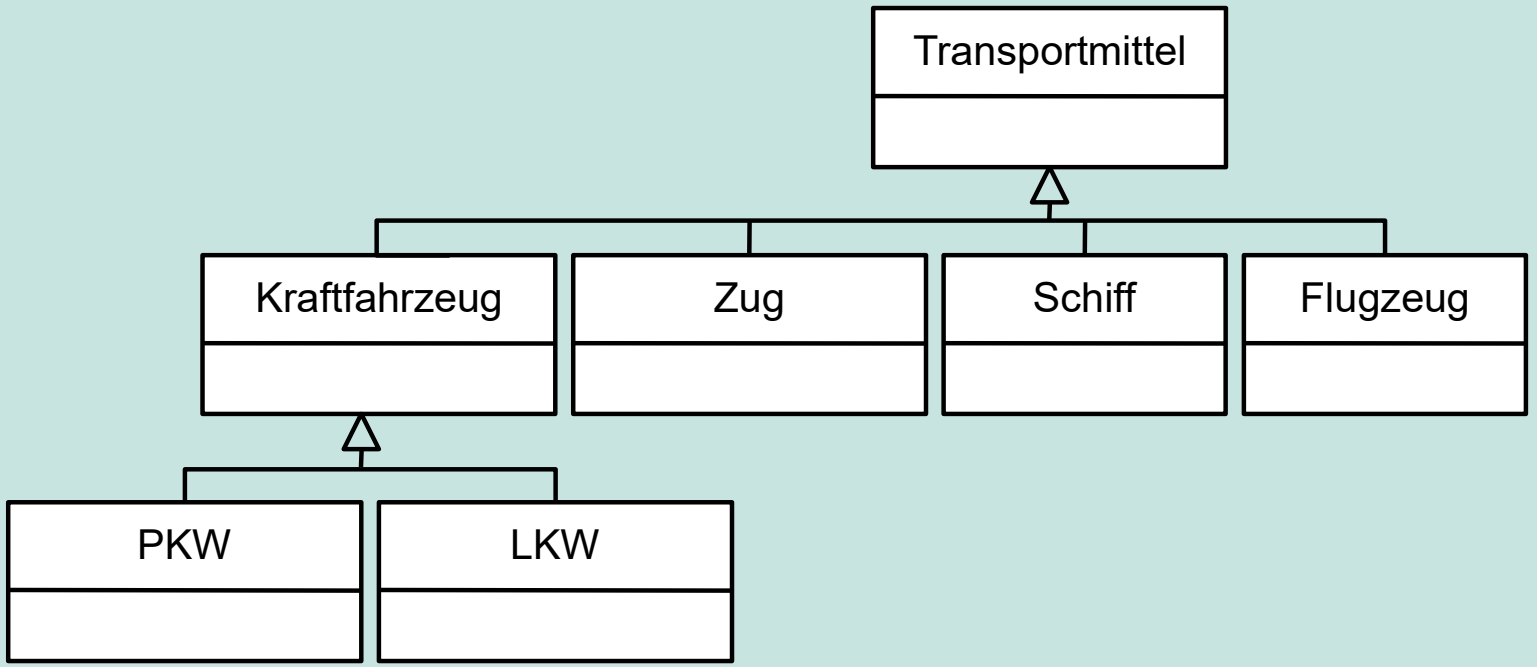
Methode `loesche()`

Beliebiges Objekt der Klasse `Punkt`:



UML-Grundlagen – Generalisierungs-Beziehung

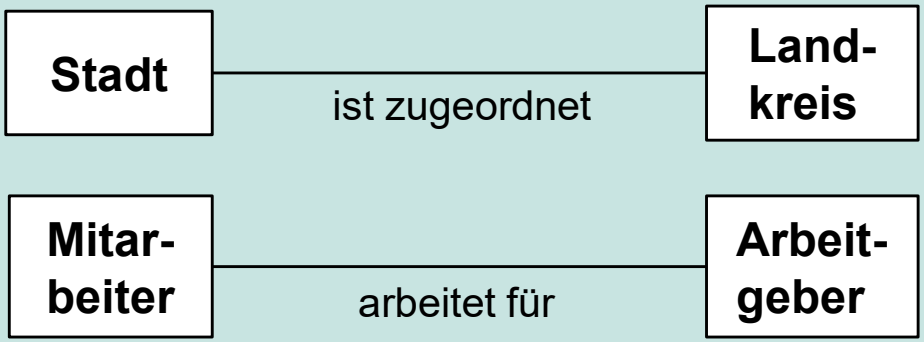
Generalisierungs-Beziehung = Vererbungsbeziehung:



Vererbung ist in der Objektorientierung ein Mechanismus für die Wiederverwendung von Programmcode!

UML-Grundlagen – Binäre Assoziation

Binäre Assoziation:



Leserichtung:

- von oben nach unten
- von links nach rechts
- mit Lesepfeil

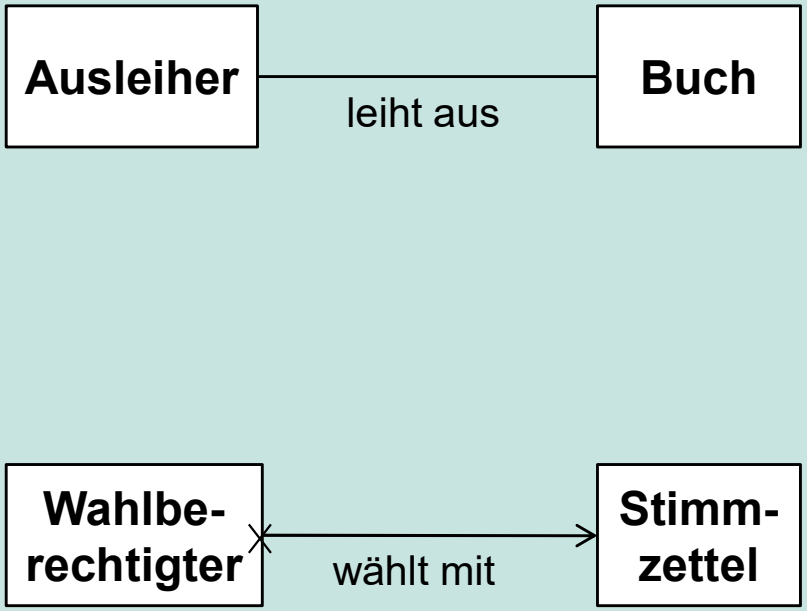
UML-Grundlagen – Assoziation mit Multiplizitäten

Multiplizitäten:

Multiplizität	Bedeutung
1	Genau ein
*	Viele, kein oder mehr, optional
1 .. *	Ein oder mehr
0 .. 1	Kein oder ein, optional
m	Genau m
m .. n	m bis n
m, n, k	m oder n oder k

UML-Grundlagen – Assoziation mit Navigation

Navigation:



Erläuterung:

- Eine Assoziation (Strich zwischen zwei Klassen) ist bidirektional oder unspezifiziert.

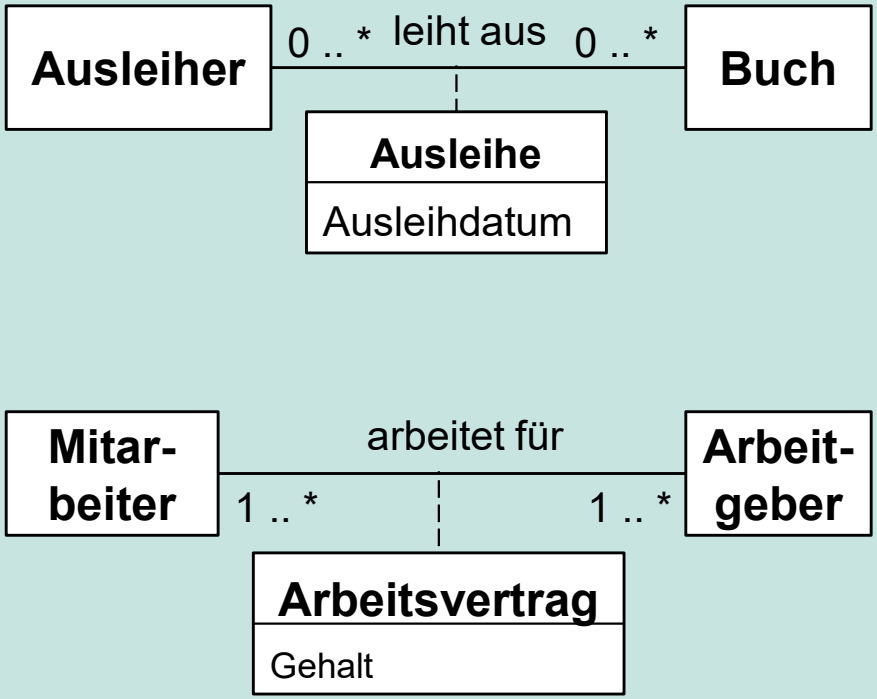
  Beispiel: Ein Ausleiher leiht ein Buch aus und das Buch wird durch den Ausleiher ausgeliehen.

- Die Navigierbarkeit kann auf eine Richtung eingeschränkt werden → unidirektionale Navigierbarkeit.

  Beispiel: Eine Person als Wahlberechtigter kann mit Hilfe eines Stimmzettels wählen, aber vom Stimmzettel aus soll es nicht möglich sein, zurück zur Person zu gelangen.

UML-Grundlagen – Assoziation mit Assoziationsklasse

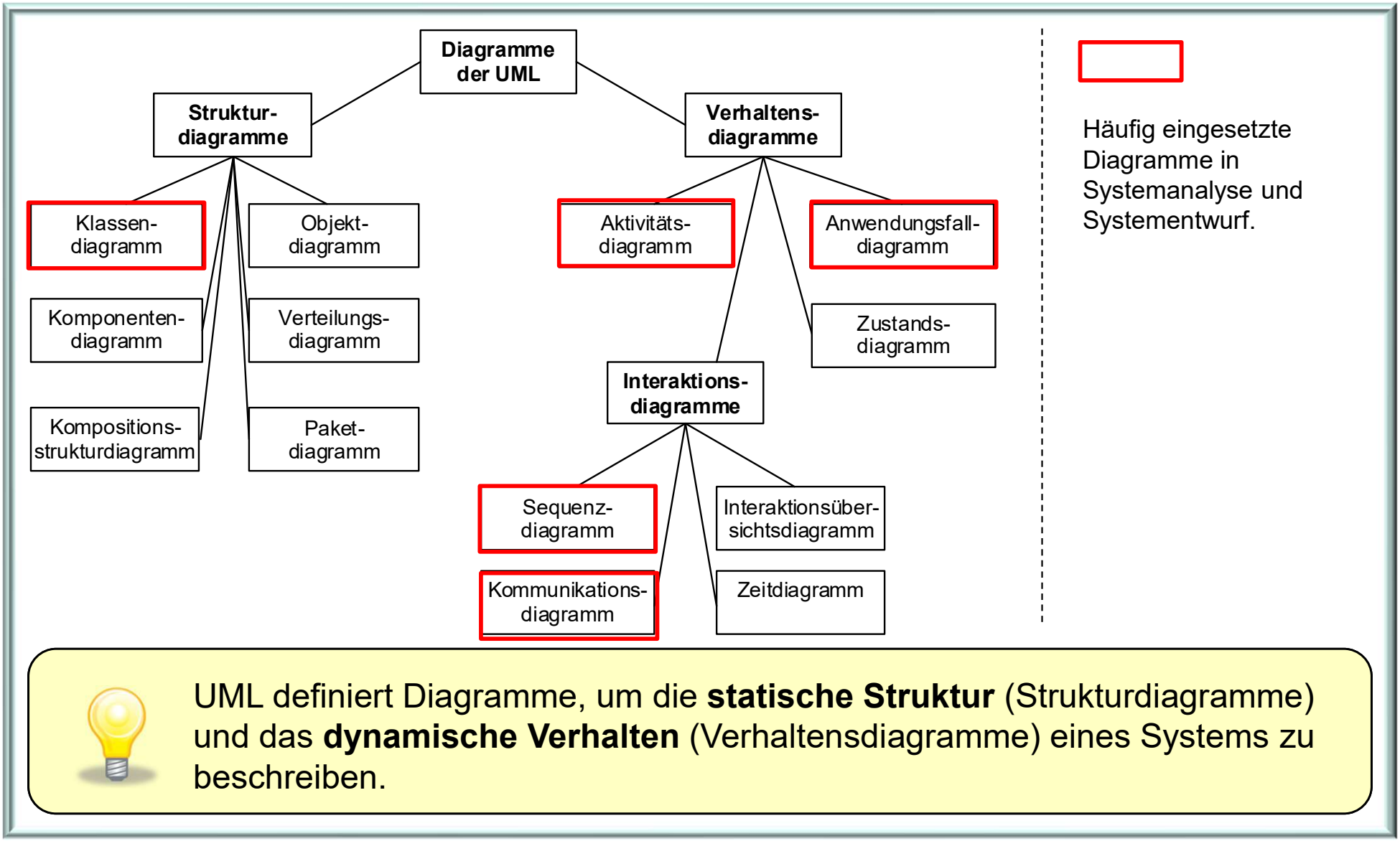
Assoziationsklasse:



Erläuterung:

- Die Assoziationsklasse fällt weg, wenn die Beziehung gestrichen wird.
- Eine Assoziationsklasse gehört also zur Assoziation und beschreibt diese näher.
- Enthält die Assoziationsklasse keine Operationen, so spricht man auch von einem Assoziationsattribut.
- Der Name der Assoziationsklasse sollte mit dem Namen der Assoziation übereinstimmen bzw. sich daraus direkt ableiten lassen:
  - leiht aus → Ausleihe
  - arbeitet für → Arbeitsvertrag

Diagramme der UML



UML definiert Diagramme, um die **statische Struktur** (Strukturdiagramme) und das **dynamische Verhalten** (Verhaltensdiagramme) eines Systems zu beschreiben.

Agenda

- Die Denkweise in Objekten und Klassen
- Einführung in die UML
- Übungsaufgabe

Übungsaufgabe

Aufgabe: Prüfungsverwaltungssystem

Es soll ein Web-basiertes Prüfungsverwaltungssystem entwickelt werden. Es sollen folgende Tätigkeiten unterstützt werden:

- Das Prüfungsamt soll die Prüfungen planen können.
- Ein Student soll sich zu Prüfungen anmelden können.
- Ein Professor soll die Prüfungsergebnisse eingeben können.
- Ein Student soll seine Prüfungsergebnisse einsehen können.

Prüfungen können sein: Seminararbeiten, Abschlussarbeiten, Pflichtprüfungen und Wahlpflichtprüfungen. Ein Professor ist für die inhaltliche Erstellung der Prüfung zuständig.

Erstellen Sie für die Klassen `Professor`, `Student`, `Prüfungsamt`, `Prüfung`, `Seminararbeit`, `Abschlussarbeit`, `Pflichtprüfung` und `Wahlpflichtprüfung` ein UML Klassendiagramm. Können Sie eine zusätzliche Assoziationsklasse identifizieren?

Vorbereitungszeit: 10 Minuten

Ergebnisdiskussion: 10 Minuten

Übungsaufgabe

Aufgabe: Prüfungsverwaltungssystem