UNIVERSITÄT
LEIPZIG

Einführung in die Objekt-Orientierte Modellierung und Programmierung Übung 2

Wintersemester 2025/2026

Tomas Daetz

Praktikumseinteilung im Moodle!

Quiz

Gruppe A (Montag):
https://particify.uni-leipzig.de/p/83847050

Gruppe B (Donnerstag):
https://particify.uni-leipzig.de/p/63486972

Heutige Ziele

- Grundlagen: imperatives Programmieren
- Einführung objekt-orientierte Programmierung (OOP)
- Einführung Modellierung

Höhere Programmiersprachen & Programmierparadigmen

- Höhere Programmiersprachen (engl. *High-level programming language*) sind “abstrakter”
- Abstraktion: unterschiedliche Stufen und Arten

Programmierparadigma

Ein Programmierstil unter einer gewisse Betrachtung der Welt. Diese hat enorme Auswirkungen wie Software konzipiert und implementiert wird.

- Beispiele (vereinfacht)
  - *Alles kann als Funktionen dargestellt und berechnet werden*
  - *Alles ist Wissen und deren logischen Ableitungen*
  - *Alles sind Objekte und deren Beziehungen* $\rightarrow$ **objekt-orientiert**

Imperative Programmierung

- Einer der ältesten Paradigmen
- Sehr maschinennah
- Programme sind Sequenzen von Anweisungen
- Beschreibt **genau** was der Computer tun muss
- Beispiele
  - Assembler
  - C
  - Fortran

- Grundbausteine:
  - Deklaration bzw. Zuweisung von Variablen
  - Bedingte Anweisungen
  - Schleifen
  - Deklaration bzw. Aufrufe von Unterprogrammen

Was ist eine Variable?

- Im Allgemeinen nur einen Ort im Speicher
- Man kann Variablen *Werte* zuweisen sowie den gespeicherten Wert lesen
- Variablen haben eindeutigen *Bezeichner* $\rightarrow$ *Achtung: Schlüsselwörter*
- Variablen haben meistens *Modifikatoren* (später) sowie einen *Typ*
- Datentyp: Bedeutung von im Speicher gespeicherten Wert

Primitive Datentypen

- Einschränkung des Wertebereichs bzw. Semantik des binärkodierten Wert
- Ganze Zahlen (negativ und positiv):
  - `byte` (1 Byte), `short` (2 Bytes), `int` (4 Bytes), `long` (8 Bytes)
  - Beispiele: Anzahl der Studierenden im Hörsaal, Anzahl der Sitzplätze in der Tram
- Gleitkommazahl (negativ und positiv):
  - `float` (4 Bytes), `double` (8 Bytes)
  - Beispiele: Temperatur im Hörsaal, Preis eines Artikels
- Charaktere
  - `char` (2 Bytes), Beispiele: 'F', 'a', ' ' (Leerzeichen)
- Wahrheitswert
  - `boolean` (1 Bit), mögliche Werte nur `true` oder `false`

Variablen in Java

- Man muss Variablen immer **deklarieren**, d.h. die Variable *einführen*
  - Pflicht: Datentyp und Bezeichner
  - z.B. `double nameDerVariable;`
- Es gibt mehrere Varianten von Variablen
  - Als Parameter in Methoden
  - Als Attribute bzw. Klassenvariablen
  - Als lokale Variablen
- Variablen können **initialisiert** werden
  - Entweder direkt nach der Deklaration `double nameDerVariable = 1.0;`
  - oder später als eigene Anweisung `nameDerVariable = 1.0;`

Was ist eine Methode?

- Meistens gibt es Programmteile die sich **wiederholen**
- Oder Programmteile werden zu lang/komplex
- Ziel:
  - Lesbarkeit
  - Wiederverwendbarkeit
  - Wartbarkeit
  - Geringe Fehleranfälligkeit
- Widerholende Programmteile nur an einer Stelle haben $\rightarrow$ Methode
- Methoden haben einen *Bezeichner*, Eingaben (Parameter), sowie *Modifikatoren* und einen *Rückgabetyp*
- Unterschiedliche Arten von Methoden in Java (bzw. in OOP)

Was ist ein Objekt?

- Vieles kann als ein Objekt betrachtet werden (in realer Welt sowie Abstrakte Konzepte)
- Objekte haben Eigenschaften (sogenannte *Attribute*)
- Operationen können auf die Objekten angewendet werden (oder Objekte können Aktionen durchführen)

Beispiel: **Kühlschrank**

Eigenschaften:

- Größe
- Aktuelle innere Temperatur
- Aktueller Inhalt
- Stromverbrauch

Aktionen:

- Kühlschrank öffnen
- Überprüfen was im Kühlschrank ist
- Essen reinstellen
- Temperatur einstellen

Objekt-orientierte Programmierung (OOP)

- Softwareteile werden als Objekte betrachtet
- Objekte haben eigene gespeicherte Daten (Eigenschaften)
- Auf die Objekte können Methoden angewendet werden
- Normalerweise können diese Objekte sich mit diesen Methoden kommunizieren

Beispiel **Kühlschrank**

- Jeder einzelner Kühlschrank ist ein Objekt
- Jeder Kühlschrank unterscheidet sich
  - Kühlschrank 1: Marke “Bosch”, 120 L Innenraum
  - Kühlschrank 2: Marke “Boman”, 60 L Innenraum
- Aber letztendlich: Kühlschränke
- Man kann auf gleiche Art und Weise mit beiden interagieren

Klassen und Instanzen

- In Java beschreiben *Klassen* mehrere Objekte
  - *Art* der Eigenschaften (aber keine festen Werte)
  - Beschreibung des Verhaltens
  - Kann als Bauplan für einen Objekt der Klasse gedacht werden
- Objekte gehören immer zu einer Klasse
- Falls ein Objekt einer Klasse gehört $\rightarrow$ Objekt ist eine *Instanz* der Klasse
- Konstruktor: Spezielle Methode zum erstellen einer Instanz
- Instanzvariablen, Methoden sowie Konstruktoren haben eine **Sichtbarkeit**

Klassen und Instanzen: Beispiel

Beispiel: **Auto**

Eigenschaften:

- Marke
- Leistung
- (Anzahl) Sitzplätze
- Reifen

Aktionen:

- Auto starten
- Beschleunigen und Bremsen
- Hupen
- Abbiegen

Modellierung

- Entwicklung von Software kann sehr kompliziert werden
  - Große Projekte
  - Komplexe Systeme
  - Nachhaltige Erweiterung und Wartung
  - Teams von mehreren Leuten → *Entwicklungsplan*
- Erstmal wird Software konzipiert, und erst danach implementiert
- Erstellung eines *Modells*

Modell

Eine vereinfachte Beschreibung eines Ausschnitts der Welt.

- Modelle dienen auch oft als Dokumentation für die Zukunft

Modellierung

- Zum Modellieren wird in diesem Modul UML (aus dem engl. *Universal Modeling Language*) genutzt
- Graphische Darstellung des Modells
- Erzeugung:
  - Per Hand mit Stift und Papier $\rightarrow$ Scan
  - Inkscape `https://inkscape.org`
  - Dia `https://wiki.gnome.org/Apps/Dia`

Klassen, UML und Java

- Software in Java wird in Klassen unterteilt
- Normalerweise 1 Klasse $\leftrightarrow$ 1 *KlassenName.java* Datei
- Wir werden nur mit einer kleinen Teilmenge von UML arbeiten
- Für die erste Aufgabe, nur das *Klassendiagram*

KlassenName
- attribut1 : int - attribut2 : double
+ konstruktor() + methode1() : int + methode2(parameter1 : int) : double

Beispiel:

Auto
- anzahlPlaetze : int - leistung : double
+ Auto(anzahlPlaetze : int, leistung : double) + getLesitung() : double + fahreVorne() : void

Aufgabe: Radio

- Erstellen Sie ein UML für ein Radio
- Jedes Radio hat eine Lautstärke und eine Empfangsfrequenz
- Stellen Sie ein Konstruktor mit passender Parameterübergabe
- Stellen Sie für jedes Attribut ein getter und setter zur Verfügung
- Implementieren Sie das UML in Java!