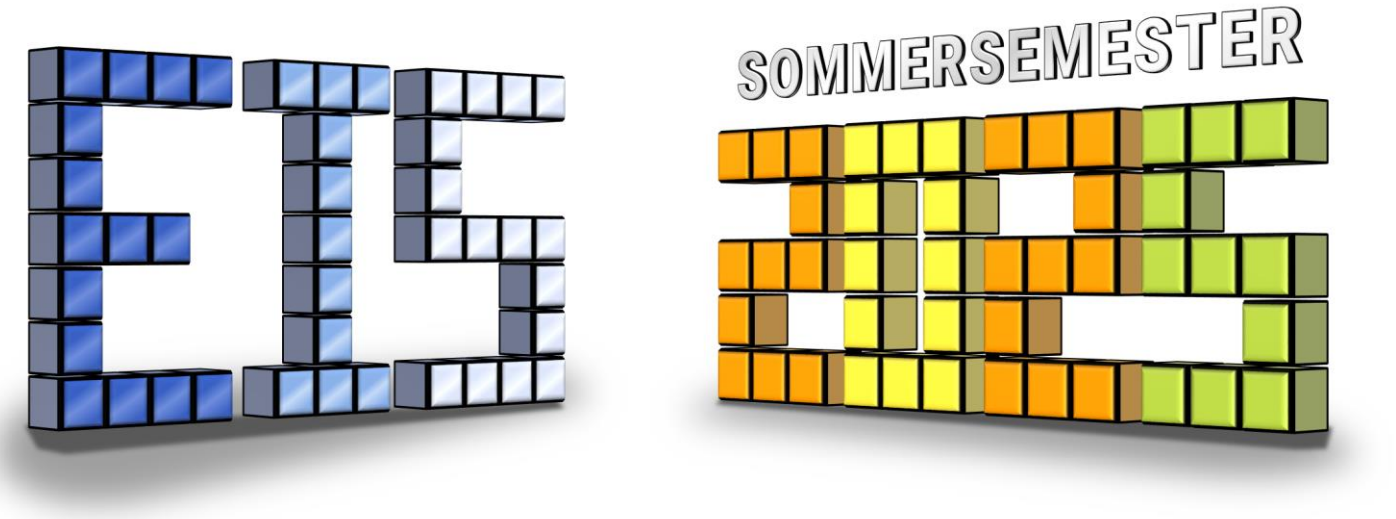
EINFÜHRUNG IN DIE SOFTWAREENTWICKLUNG

Sommersemester 2025



Foliensatz #2c

Programmiersprachen: JAVA

Michael Wand
Institut für Informatik
Michael.Wand@uni-mainz.de

Übersicht

Inhalt heute

- Programmiersprachen
  - Python + MyPy
  - C/C++
  - Java/Scala

Programmiersprachen

Java + Scala

Programmiersprachen

Java

Diskussion: Programmiersprache

Vorlage zur Diskussion

- Daten + Variablen
- Ausdrücke und Berechnungen
- Befehle
- Abstraktionen
- Systemumgebung

Was ist JAVA

Leicht zu erklären?

- Man nehme C++
- Vereinfache es stark
  - Weniger Abstraktionen: Betont vor allem OOP mit Klassen
  - Behebe Bugs und Altlasten von C++
    - Weniger „footguns“ wie „`if (a=b)`“ oder „`42;`“ uvam.
    - Vernünftiges Modulkonzept
  - Nutzt (wie Python) Garbe-Collection (GC) und bietet völlige Speichersicherheit
    - Kein Zugriff auf ungültigen Speicher!
    - Dynamische Typen sicher - keine Fehlinterpretation
- Füge fehlende Features hinzu
  - z.B. Reflection seit 1997; in C++ nun versprochen für 2026

JAVA

Vorteile

- Sehr einfache Sprache
  - Konsistent: „Es gibt für alles nur eine Lösung“
  - Erzwingt einheitlichen Stil – kann in großen Teams helfen
- Sicher
  - Keine Buffer-Overruns, keine Speicherfehler

Nachteile

- Sehr einfache Sprache
  - Fortgeschrittene können sich eingeengt fühlen
- (Etwas) weniger effizient als C/C++
  - GC schlecht für Echtzeitanwendungen

Diskussion: Programmiersprache

Vorlage zur Diskussion

- Daten + Variablen
- Ausdrücke und Berechnungen
- Befehle
- Abstraktionen
- Systemumgebung

Datentypen

Primitive Typen sind Werttypen (wie in C)

- **`byte`**, **`short`**, **`int`**, **`long`** (8,16,32,64 Bit signed integer)
- **`char`** (16 Bit Unicode-1 Zeichen, unsigned)
- **`float`**, **`double`** (32- und 64-Bit Fließkomma)
- **`boolean`** (true, false, *nicht* **`bool`** wie in C++)
- Selbstdefinierte Werttypen z.Zt. als „preview“

Wertsemantik

- Bei Zuweisung werden Inhalte kopiert
  - Auch bei Parametern für Unterprogrammen

Datentypen

Komplexe Typen sind immer Referenztypen

- Oberklasse „`Object`“ (wie `object` in Python)
  - Schließt `String`, `Array` (ähnlich `str`, `list`) ein
- Boxing: Primitive in Objekte verpacken
  - z.B. **`int`** → `Integer`, **`double`** → `Double`, etc.
  - Nützlich z.B. für
    - Rückgabe von Werten wie in C++ (`void func(int &x)`)
    - Primitive Typen in Container für Objekte einfügen

Spezialfälle (auch Referenztypen)

- `String`: Klasse für immutable strings (wie `str` in Python)
- Arrays: Notation wie in C++
  - z.B. `int[] myArray = new int[42];`
  - „Sichere“ Objekte mit Laufzeitprüfung (Index-Überschreitung u.ä.), Member-Methoden und Feldern, etc. (…`myArray.length`…)

Diskussion: Programmiersprache

Vorlage zur Diskussion

- Daten + Variablen
- Ausdrücke und Berechnungen
- Befehle
- Abstraktionen
- Systemumgebung

Ausdrücke und Berechnungen

Nicht viel neues…

- …alles im Prinzip wie in C++
- Inkonsistenzen und Fehlerquellen behoben
- Wichtigste Änderung: Memory-Safety
  - Keine Zeigerarithmetik
    - Mit Referenzen kann man nicht rechnen
    - Nur Arrays, wenn man mit Indices rechnen will
  - Referenzen mit Garbage Collection
    - Es gibt zwar **`new`**, aber kein explizites **`delete`**
    - Null-Pointer (**`null`**) Zugriff führt zu Laufzeitfehler
  - Casting von Objektreferenzen wird zur Laufzeit geprüft

Diskussion: Programmiersprache

Vorlage zur Diskussion

- Daten + Variablen
- Ausdrücke und Berechnungen
- Befehle
- Abstraktionen
- Systemumgebung

Befehle

Befehle / Syntax wie in C++

- Änderungen für Unterprogramme
  - Alle Unterprogramme sind Methoden von Klassen
  - „**`static`**“-Methoden für solche, die nicht mit Objekten arbeiten
- Beispiel (aus Übungsblatt 01)

```
public class CountDown {   // Klasse
    public static void main(String[] args) {
        // Hauptprogram als Beispiel
        ...
    }
}
```

Diskussion: Programmiersprache

Vorlage zur Diskussion

- Daten + Variablen
- Ausdrücke und Berechnungen
- Befehle
- Abstraktionen
- Systemumgebung

Klassen

JAVA als Untermenge von C++

- Aber wichtige Features ergänzt (z.B. Reflection)
- Klassen, Vererbung und Objekte als Kernabstraktion
  - Eigentlich ähnlich zu Python, aber
    Subtyping statt Ducktyping wie in C++ (schneller, sicherer)
  - Interfaces und Traits (statt „Mehrfachvererbung“)
- Inzwischen auch
  - Generics für mehr Typsicherheit
  - Funktionszeiger und -Variablen ohne extra Klassendefinitionen
  - Patternmatching u.v.a.m.
- Alternative: JAVA mit mehr Features ≈ C#

Diskussion: Programmiersprache

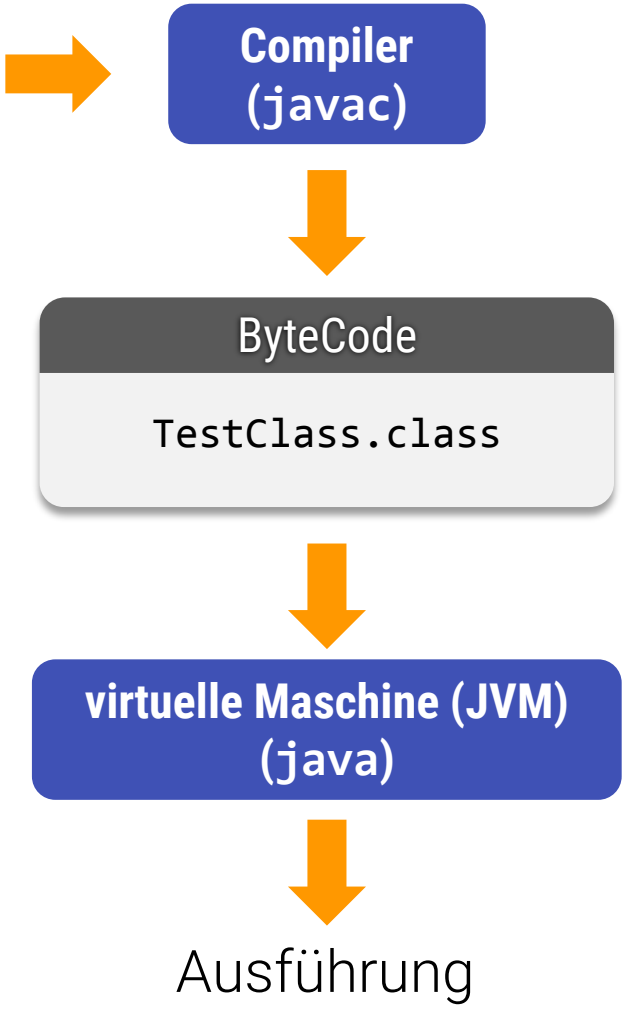
Vorlage zur Diskussion

- Daten + Variablen
- Ausdrücke und Berechnungen
- Befehle
- Abstraktionen
- Systemumgebung

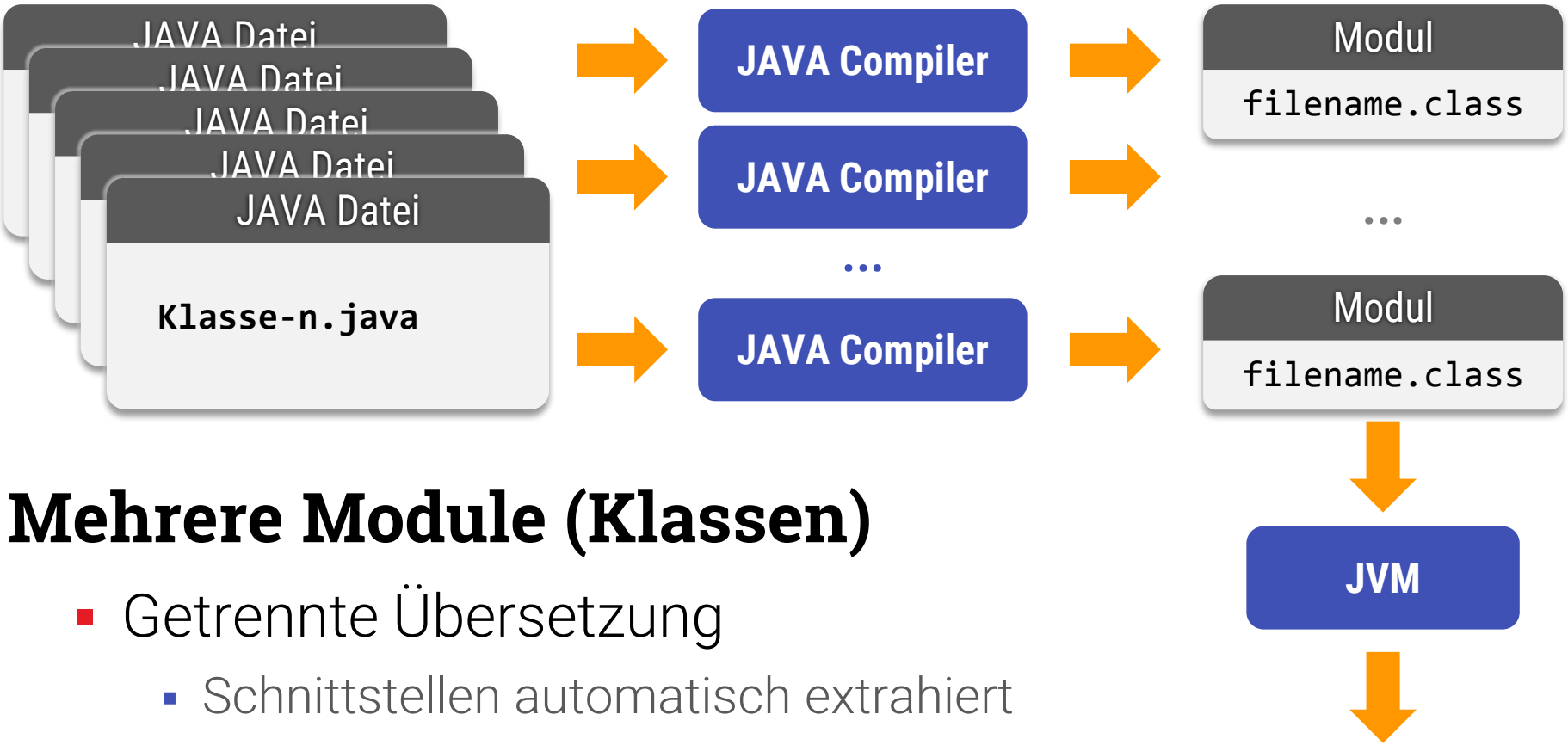
Eine Klasse pro Datei

Datei: TestClass.java

```
class TestClass {
 public static void main() {
   int var = 42;
   System.println("Hello World!");
 }
}
```



Komplexes Systeme: Dynamisches Laden



Mehrere Module (Klassen)

- Getrennte Übersetzung
  - Schnittstellen automatisch extrahiert
- Menge von „`.class`“ Dateien
  - Eine „starten“, andere werden bei Bedarf dynamisch geladen

Einbindung von Modulen

Java Module sehr ähnlich zu Python

- `import myMod.MyClass;`
  `import myMod.*;`
  - Modul / Klasse wird bereitgestellt
- Hierarchische Pakete via
  `import package.subdir.myMod;`
  - Abgebildet auf Verzeichnisstruktur
  - Deklaration von Packeten via `package myMod;`
- Dynamisches Laden zur Laufzeit bzw. Auffinden der Schnittstellen zur Übersetzungszeit
  - Umgebungsvariable CLASSPATH zeigt auf verfügbare Packages