Einleitung

Einführung in die Programmierung 1

Sommersemester 21

TU Wien

Überblick

- Einführung in die Programmierung 1
- Java – Ein erster Überblick

Einführung in die Programmierung 1

Lernergebnisse

- Nach positiver Absolvierung der Lehrveranstaltung sind Studierende in der Lage
  - wichtige Konzepte einer aktuellen alltagstauglichen Programmiersprache (Java) zu beschreiben
  - Inhalte natürlichsprachiger Programmieraufgaben in ausführbare kleine Programme umzusetzen
  - Vorgehensweisen und Werkzeuge beim Programmieren anzuwenden
  - ausgewählte Algorithmen zu implementieren

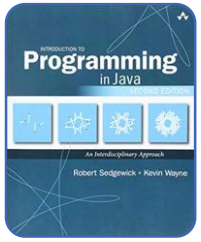
Themengebiete

Grundlagen	Variablen, Datentypen und Operatoren
	Verzweigungen
	Elementare Ein-/Ausgabe
	Schleifen
	Methoden
	Rekursion
	Arrays
Aufbauende Themen	Grundlegende Algorithmen
	Ausblick auf Objektorientierung
	Programmierstil

Literatur

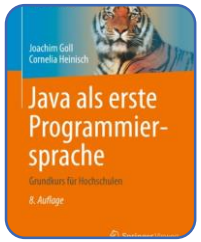
H. Mössenböck, **Sprechen Sie Java? Eine Einführung in das systematische Programmieren**, dpunkt.verlag, 5. Auflage, 2014

R. Schiedermeier, **Programmieren mit Java**, Pearson Studium, 2. Auflage, 2010



R. Sedgewick, K. Wayne, **Introduction to Programming in Java: An Interdisciplinary Approach**, Addison Wesley, 2. Auflage, 2017

Literatur



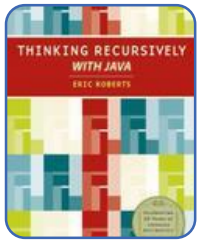
J. Goll, C. Heinisch, **Java als erste Programmiersprache**, Springer Verlag, 8. Auflage, 2016

C. Ullenboom, **Java ist auch eine Insel**, Rheinwerk Computing, 15. Auflage, 2020

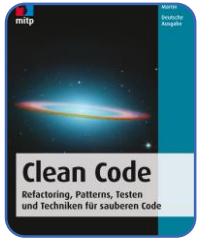
Online (12. Auflage): http://openbook.rheinwerk-verlag.de/javainsel/

M. Inden, **Java – die Neuerungen in Version 9 bis 14: Modularisierung, Syntax- und API-Erweiterungen**, dpunkt.verlag , 1. Auflage, 2020

Literatur



E. Roberts, **Thinking Recursively with Java**, Wiley, 1. Auflage, 2006



R. C. Martin, **Clean Code – Refactoring, Patterns, Testen und Techniken für sauberen Code**, mitp, 1. Auflage, 2009

R. Sedgewick, K. Wayne, **Algorithms**, Addison Wesley, 4. Auflage, 2011

Java – Ein erster Überblick

Sprache

Maschinensprache

- Wird vom Rechner „verstanden“
- Befehle stehen als Bitmuster im Speicher
- Programmierung ist aufwändig und fehleranfällig

Natürliche Sprache

- Für uns beste Wahl
- Kann von Rechnern schwer verarbeitet werden

Hochsprache (höhere Programmiersprache)

- Kompromiss
- Ein Befehl in der Hochsprache entspricht mehreren Befehlen in Maschinensprache
- Programm muss in Maschinensprache übersetzt werden

Hochsprache umwandeln

Übersetzende Programmiersprachen

- Der Quelltext (Sourcecode) eines Programms wird vor der ersten Ausführung durch einen **Compiler** (Übersetzer) in eine Zielsprache (üblicherweise Maschinensprache) übersetzt
- Hohe Ausführungsgeschwindigkeit
- Überprüfung auf syntaktische Fehler

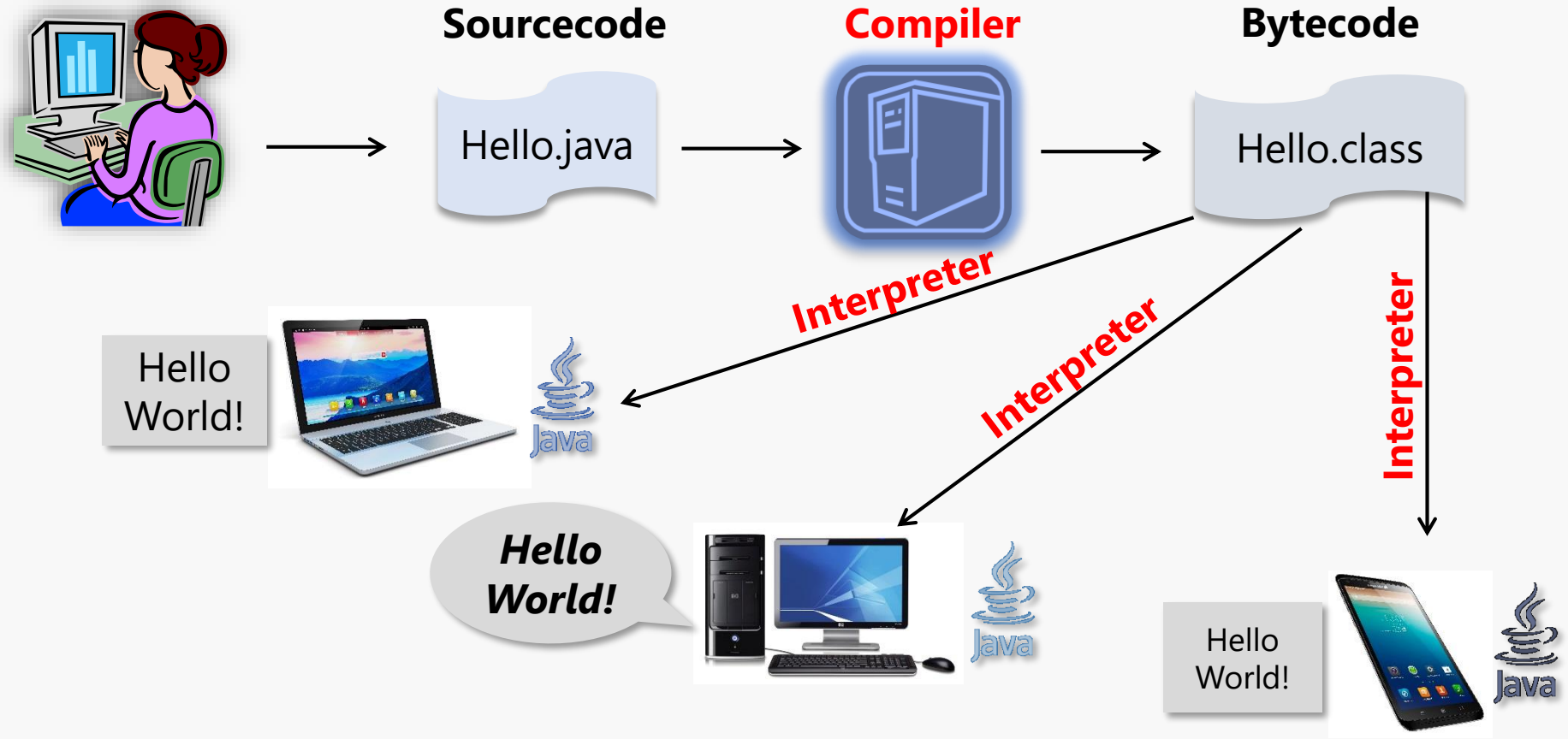
Interpretierende Programmiersprachen

- Der Quelltext eines Programms wird zur Laufzeit von einem **Interpreter** eingelesen und Befehl für Befehl abgearbeitet
- Langsamer
- Flexibler (bei kleinen Änderungen, bei unterschiedlichen Architekturen)

Mischformen (z. B. Java)

- Programm wird in einen **Zwischencode** übersetzt
- Zwischencode wird **interpretiert** (virtuelle Maschine)

Java



Mittlerweile viele Java-Versionen https://en.wikipedia.org/wiki/Java_version_history

Hello World in Java

```java
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World");
    }

}
```

Hello World in Java

Alles muss zu einer Klasse gehören (in diesem Fall zur öffentlichen (public) Klasse HelloWorld)

main ist der Startpunkt für das Programm

```
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World");
    }

}
```

„Befehle“ - zunächst unser Arbeitsplatz

Ende der Klasse

Übersetzen und Ausführen

- Auf der Kommandozeile
  - Übersetzen

    ```
    javac HelloWorld.java
    ```
  - Ausführen

    ```
    java HelloWorld
    ```
- Entwicklungsumgebungen
  - Z. B. IntelliJ
  - https://www.jetbrains.com/idea/
- IntelliJ-Demo
  - Weitere Informationen in der Übung

Beispiel (Ausblick)

- Gib die Summe aller geraden Zahlen im Intervall [2, 100] aus

```java
public class SpecialSum {

    public static void main(String[] args) {

        int sum = 0;
        int counter = 2;

        while (counter <= 100) {
            if ((counter % 2) == 0) {
                sum = sum + counter;
            }
            counter = counter + 1;
        }

        System.out.println("Sum: " + sum);
    }

}
```

Beispiel – Erklärung

```
…
int sum = 0;

int counter = 2;

while (counter <= 100) {
     if ((counter % 2) == 0) {
          sum = sum + counter;
     }
     counter = counter + 1;
}

System.out.println("Sum: " + sum);

…
```

Variable sum wird bekannt gegeben und erhält den Wert 0

Variable counter wird bekannt gegeben und erhält den Wert 2

Solange counter <= 100 ist, wird die Schleife wiederholt

Variable sum wird um den Wert von counter erhöht, **falls** counter durch 2 teilbar ist - counter wird danach um 1 erhöht

Die Summe wird auf den Bildschirm **ausgegeben**

Zusammenfassung

- EP1
  - Ziele
  - Themen
  - Literatur
- Java
  - Programmieren allgemein
  - Hello World in Java und Java-Umgebung