

Kapitel 1: Grundlagen

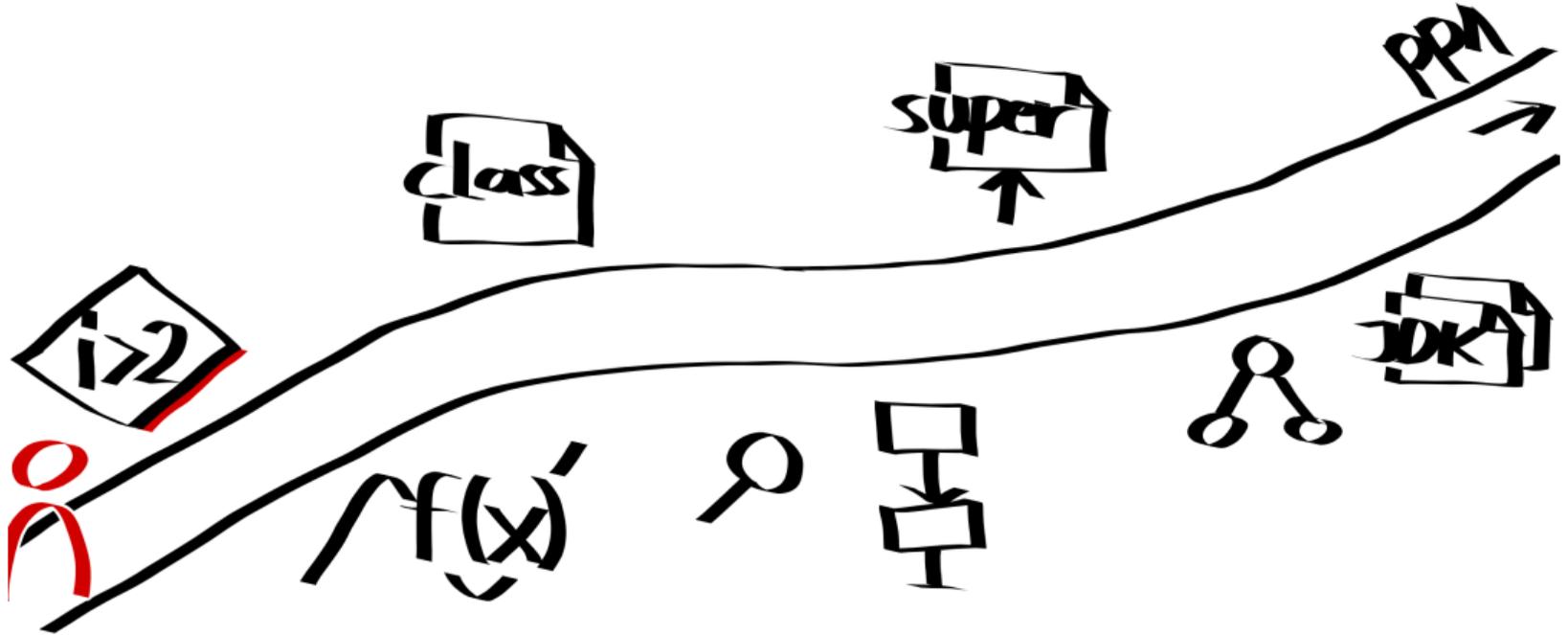
VL 1: Hello World



Die Vorlesungsfolien basieren auf den Folien von

- Markus Brenneis
- Michael Schöttner
- Stefan Harmeling
- Robert Sedgewick & Kevin Wayne,
<https://introcs.cs.princeton.edu/java/home/>

Wo stehen wir gerade?



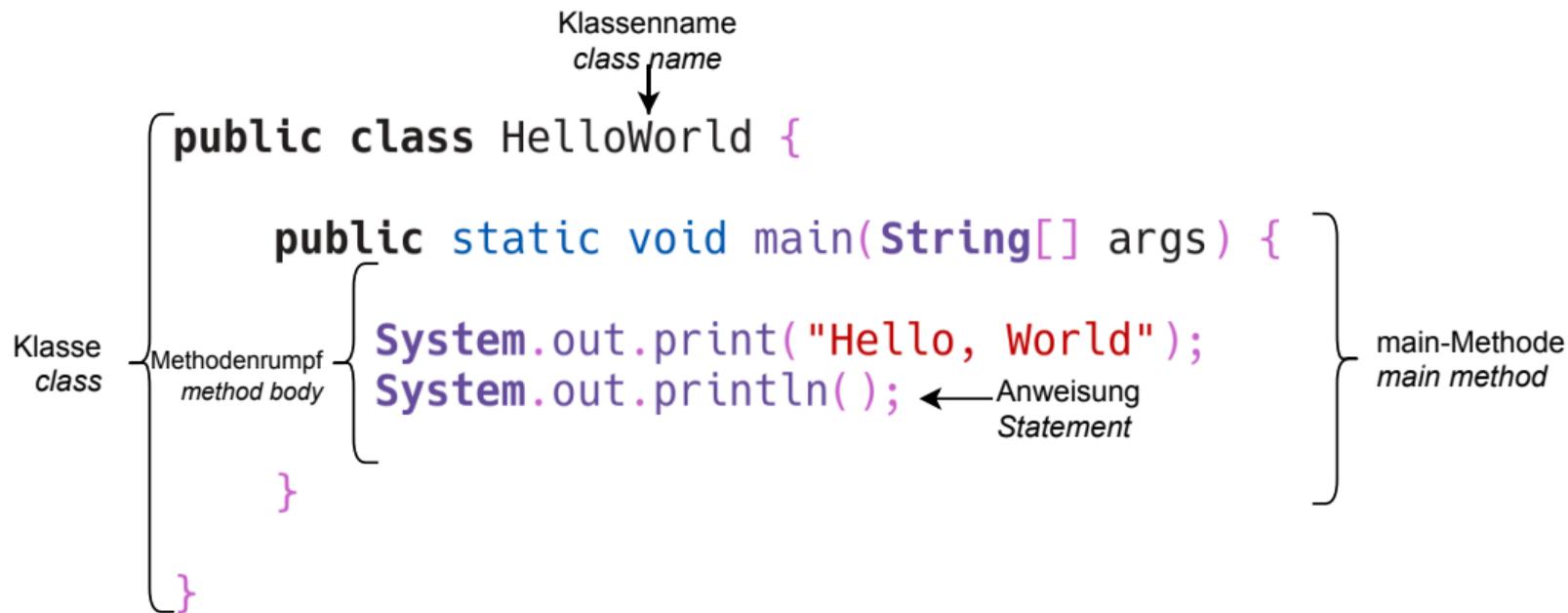
Wie schreibe ich ein Java-Programm?

- Datei `Programmname.java` mit einem Texteditor anlegen und befüllen
- Terminal öffnen und mit `cd` in das richtige Verzeichnis wechseln
- Das Programm übersetzen: `javac Programmname.java`
- Das Programm ausführen: `java Programmname`

```
1 public class HelloWorld {  
2  
3     public static void main(String[] args) {  
4         System.out.println("Hello World!");  
5     }  
6  
7 }
```

Rad 6 av 7, Kolumn 1, Ord 0/14, Tecken 0/118 Zoom: 190 % INFOGA de_DE vjuka tabulatorer: UTF-8 Java

```
~ % cd progra 9:17:36  
~/progra % ls 9:17:38  
HelloWorld.java  
~/progra % javac HelloWorld.java 9:17:39  
~/progra % ls 9:17:42  
HelloWorld.class HelloWorld.java  
~/progra % java HelloWorld 9:17:43  
Hello World!  
~/progra % 9:18:04
```



- für jetzt: Klassenname = Programmname = Dateiname ohne .java
 - später: Klassen und Methoden im Detail

Definition

Ein **Betriebssystem** ist ein Computerprogramm, das die Betriebsmittel eines Computers (Arbeitsspeicher, Prozessorlaufzeit, Festplatten usw.) verwaltet und diese Betriebsmittel Anwendungsprogrammen über eine definierte Schnittstelle zur Verfügung stellt. Beispiele für Betriebssysteme sind Apple macOS, Google Android, Microsoft Windows und Ubuntu.

¹engl.: Operating System (OS)

Definition

Die **Standardausgabe** ist ein vom Betriebssystem bereitgestellter Kommunikationskanal, über den ein Programm eine Ausgabe an seine Umgebung schicken kann. Wenn ein Programm in der Konsole ausgeführt wird, wird die Standardausgabe typischerweise als Text auf dem Bildschirm ausgegeben. In Java verwenden wir die Anweisungen `System.out.print` bzw. `println` zum Schreiben auf die Standardausgabe.

Anmerkungen:

- Programm selbst oder Konsole kann Ausgabe „umlenken“ (→ später)
- weitere Standard-Datenströme: Standardfehlerausgabe, Standardeingabe (→ später)

²standard output

Definition

Der **Quellcode** ist ein für Menschen lesbarer Text, der ein Computerprogramm definiert. Eine **Programmiersprache** ist eine formale Sprache, die vorgibt, wie der Quellcode aussieht und welche Bedeutung er hat. Java ist eine Programmiersprache.

³Quelltext, kurz: Code, engl.: Source Code

Definition

Ein **Compiler** ist ein Computerprogramm, das Quellcode einliest und in eine Abfolge von Binärzahlen übersetzt, die der Computer (direkt oder indirekt) ausführen kann. Dabei prüft der Compiler, ob die formalen Regeln der Programmiersprache eingehalten werden. Diesen Vorgang nennt man **kompilieren**. Das Programm `javac` ist der Compiler der Programmiersprache Java.

⁴dt.: Übersetzer

Definition

Die **Java Virtual Machine** (JVM) ist ein Computerprogramm, das übersetzten Java-Programmcode ausführen kann.

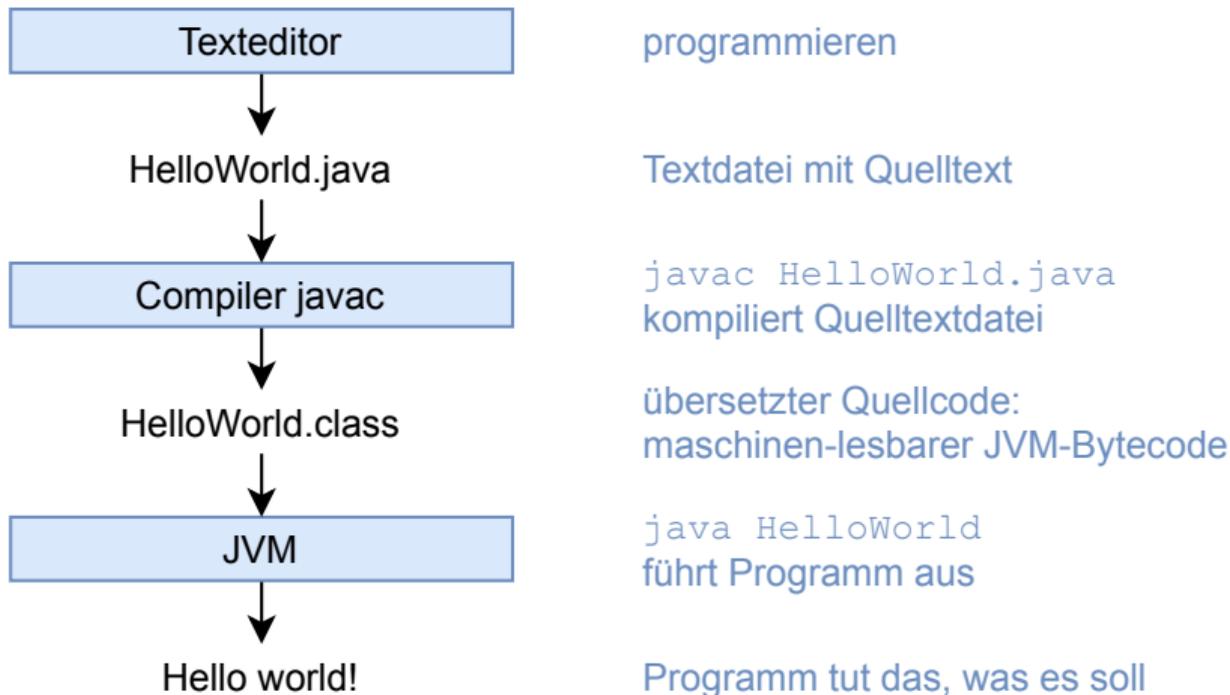
Warum generiert `javac` keinen Code, der vom Computer direkt ausgeführt werden kann?

Definition

Die **Java Virtual Machine** (JVM) ist ein Computerprogramm, das übersetzten Java-Programmcode ausführen kann.

Warum generiert `javac` keinen Code, der vom Computer direkt ausgeführt werden kann?

- übersetzter Code überall lauffähig, wo eine JVM verfügbar
 - Unabhängigkeit des Programms von Betriebssystem und Prozessor
- JVM optimiert Code der class-Dateien während Laufzeit



HelloWorld.java

Java

```
1 public class HelloWorld {
2     public static void main(String[] args)
3         System.out.print("Hello, World");
4         System.out.println();
5     }
6 }
```

Wo ist der Fehler?

Compiler-Fehlermeldungen verstehen II

```
% javac HelloWorld.java
HelloWorld.java:2: error: ';' expected
    public static void main(String[] args)
                        ^
HelloWorld.java:4: error: <identifier> expected
        System.out.println();
                        ^
HelloWorld.java:6: error: class, interface, or enum expected
    }
    ^
3 errors
```

- `HelloWorld.java:2` bedeutet: Fehler in Zeile 2 in Datei HelloWorld.java
- Fehler von oben nach unten abarbeiten
 - Spätere Fehler können durch vorherige ausgelöst sein
- Fehlermeldung manchmal irreführend
- Längere Programme nicht erst komplett fertigschreiben, sondern zwischendurch compilieren und testen

Definition

Ein **Compilezeitfehler** ist ein Fehler, den der Compiler beim Übersetzen feststellt (z. B. fehlendes Semikolon).

Definition

Ein **Laufzeitfehler**⁵ ist ein Fehler, der während das Programm ausgeführt wird zu einem Absturz des Programms führt (z. B. Division durch 0, Zugriff auf nicht vorhandene Datei).

Definition

Ein **Logikfehler** ist ein Fehler, wodurch das Programm etwas tut, was es eigentlich nicht tun soll, aber es (noch) nicht zu einem Programmabsturz kommt (z. B. $>$ statt \geq).

⁵runtime error

⁶bugs

- Ziel: Quellcode für Menschen gut lesbar
 - Fehler leichter auffindbar
 - Programmfluss leichter nachvollziehbar
- Nicht für Compiler relevant

Schlecht lesbar: alles in einer Zeile

HelloWorld.java

Java

```
1 public class HelloWorld{public static void  
  ↪   main(String[]args){System.out.print("Hello,  
  ↪   World");System.out.println();}}
```

Besser lesbar: nur ein Statement pro Zeile

HelloWorld.java

Java

```
1 public class HelloWorld{
2     public static void main(String[] args) {
3         System.out.print("Hello, World");
4         System.out.println();
5     }
6 }
```

Gut lesbar: tiefere Einrückung, sobald ein Block aufgeht

HelloWorld.java

Java

```
1 public class HelloWorld {
2     public static void main(String[] args) {
3         System.out.print("Hello, World");
4         System.out.println();
5     }
6 }
```

- Nach `{` und `;` neue Zeile beginnen
→ max. eine Anweisung pro Zeile
- Leerzeichen vor `{`
- in Zeile nach einer `{` eine Ebene weiter einrücken
- `}` in eine eigene Zeile und eine Ebene weniger einrücken
- Einrückung typischerweise mit 4 Leerzeichen
→ Editor so einstellen, dass Tabulatortaste (links vom Q) 4 Leerzeichen erzeugt

- Einrückung konsequent mit Tabulator oder Leerzeichen
 - sonst je nach Editor-Einstellung Durcheinander
 - Mischung oft versehentlich durch Copy&Paste
- Java-Konvention: 4 Leerzeichen

```
1 public class HelloWorld {
2   public static void main(String[] args) {
3     System.out.print("Hello, World");
4     System.out.println();
5   }
6 }
```

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

- Ziel: Erklärungen im Quelltext hinterlegen
 - für einen selbst und für andere
 - wichtig bei komplizierteren Programmen
- Compiler ignoriert Kommentare
- Kommentare auf wichtige, relevante Stellen beschränken

HelloWorld.java

Java

```
1 public class HelloWorld {
2     /* (Ich bin ein mehrzeiliger Block-Kommentar.)
3     Dieses Programm gibt Hello World auf der Standardausgabe aus. */
4     public static void main(String[] args) {
5         // Gibt Hello World aus (einzeiliger Zeilen-Kommentar)
6         System.out.print("Hello World");
7         System.out.println(); // Beginne eine neue Zeile
8     }
9 }
```

- Empfehlungen auf der Kursseite (Abschnitt zu Woche 1)
- für den Anfang *nicht* empfohlen: IntelliJ, Eclipse, ...
 - nur sinnvoll mit ausreichend Vorwissen bedienbar
 - verstecken viel
 - langsamer
 - im Programmierpraktikum ausführlich erklärt
 - Umgang mit Terminal für Berufspraxis relevant

Sie können am Ende der Woche ...

- Java-Programme **schreiben**, die auf die Standardausgabe schreiben.
- ein Java-Programm mithilfe des Terminals **kompilieren** und **ausführen**.
- anhand von Meldungen zu Laufzeit- und Compilezeitfehlern **erkennen**, wo ein Programmierfehler ist.
- den Unterschied zwischen Laufzeit- und Compilezeitfehlern **erklären**.
- eigenen Code **kommentieren**.

`System.out.print`

Standardausgabe

Compiler

Einrückung

`javac`

`System.out.print`

Compilezeitfehler

Block-Kommentar

`java`

Quellcode

Laufzeitfehler

Zeilen-Kommentar

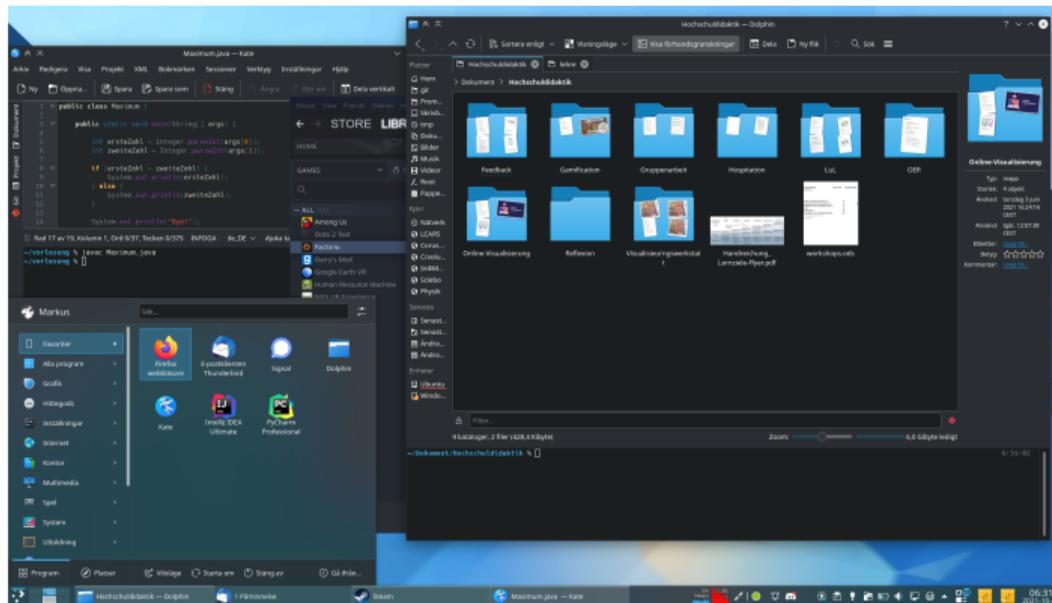
- Hello World
- Textaufgabe: Compilieren & Ausführen

- Gewählte, studentische Vertretung
- Hilfe bei Problemen (mit Profs, Stundenplan, ...); Veranstaltungen (Spieleabende, ...)
- Raum: 25.12.O1.18
- öffentliche Sitzung: wöchentlich
- Website: <https://fscs.hhu.de>
- Instagram: <https://www.instagram.com/hhufscs>
- Twitter: @hhufscs
- E-Mail-Adresse: fscs@hhu.de
- Telefon: +49 211 81 14846
- Rocket.Chat: <https://rocketchat.hhu.de/channel/fscs>
- Discord der InΦMa: <https://fscs.hhu.de/discord>
- Ersti-Stick-FAQ: <https://fscs.hhu.de/ersti-stick-faq/>

- Fr., 13.10.: Spieleabend
- Mi., 18.10.: Linux Install Party
- Fr., 20.10.: Der gestiefelte Kater
- Fr., 27.10.: Casinoabend
- Fr., 3.11.: Spieleabend
- Fr., 10.11.: Pubquiz
- Fr., 17.11.: Karaokeabend

alles kostenlos, in der Uni, Details unter <https://fscs.hhu.de>

Linux Install Party der Fachschaft Informatik



Mittwoch, 18.10.23, 16:30 Uhr, fcs.hhu.de