Kapitel 1: Hello World

Wintersemester 2021/22

Quellen

Heinrich Heine
Universität Düsseldorf

Die Vorlesungsfolien basieren auf den Folien von

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

Wie schreibe ich ein Java-Programm?

Heinrich Heine
Universität Düsseldorf

- 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`

Workflow in der Praxis

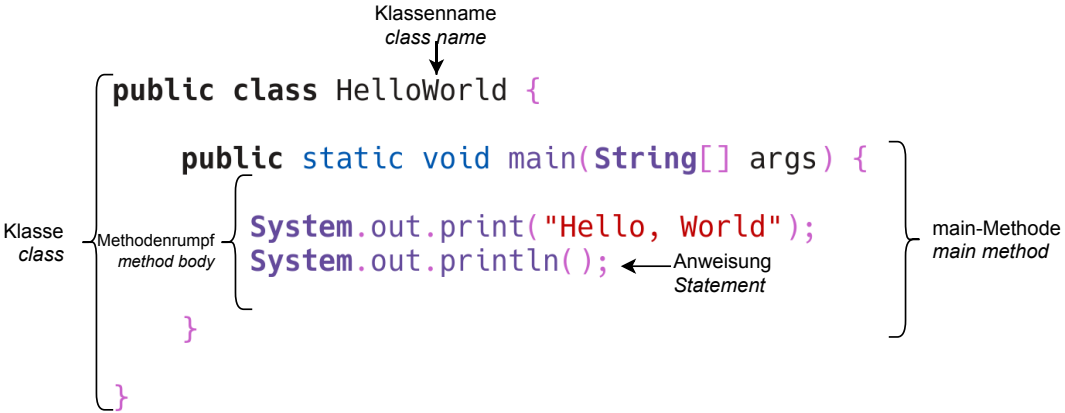
Heinrich Heine Universität Düsseldorf

```java
public class HelloWorld {

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

}
```

Rad 6 av 7, Kolumn 1, Ord 0/14, Tecken 0/118 Zoom: 190 % INFOGA de_DE vjuka tabulatorer: 4 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
```

Anatomie von HelloWorld



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

Standardausgabe[1]

Heinrich Heine
Universität Düsseldorf

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“ ($\rightarrow$ später)
- weitere Standard-Datenströme: Standardfehlerausgabe, Standardeingabe ($\rightarrow$ später)

[1] standard output

Quellcode[2]

Heinrich Heine
Universität Düsseldorf

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.

[2] Quelltext, engl.: Source Code

Compiler[3]

Heinrich Heine
Universität Düsseldorf

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.

[3] dt.: Übersetzer

Betriebssystem[4]

Heinrich Heine Universität Düsseldorf

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.

[4] engl.: Operating System (OS)

Java Virtual Machine

Heinrich Heine Universität Düsseldorf

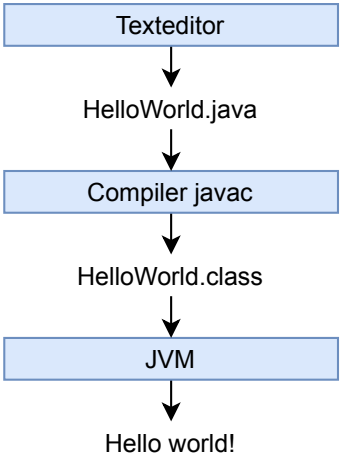
Definition

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
  - → keine direkte Abhängigkeit des Programms von Schnittstellen eines bestimmten Betriebssystems
- JVM optimiert Bytecode der class-Dateien während Laufzeit

Workflow



programmieren

Textdatei mit Quelltext

`javac HelloWorld.java`
kompiliert Quelltextdatei

übersetzter Quellcode:
maschinen-lesbarer JVM-Bytecode

`java HelloWorld`
führt Programm aus

Programm tut das, was es soll

Compiler-Fehlermeldungen verstehen I

Heinrich Heine Universität Düsseldorf

HelloWorld.java — Java

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

Compiler-Fehlermeldungen verstehen II

Heinrich Heine Universität Düsseldorf

```
% 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
```

Compiler-Fehlermeldungen verstehen III

Heinrich Heine
Universität Düsseldorf

- `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

Arten von Fehlern[6]

Heinrich Heine
Universität Düsseldorf

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 $\geq$).

[5] runtime error
[6] bugs

Formatierung des Qullecodes

Heinrich Heine
Universität Düsseldorf

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

Beispiel zur Einrückung I

Heinrich Heine Universität Düsseldorf

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();}}
```

Beispiel zur Einrückung II

Heinrich Heine Universität Düsseldorf

Besser lesbar: nur ein Statement pro Zeile

HelloWorld.java — Java

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

Beispiel zur Einrückung III

Heinrich Heine Universität Düsseldorf

Gut lesbar: tiefere Einrückung, sobald ein Block aufgeht

HelloWorld.java — Java

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

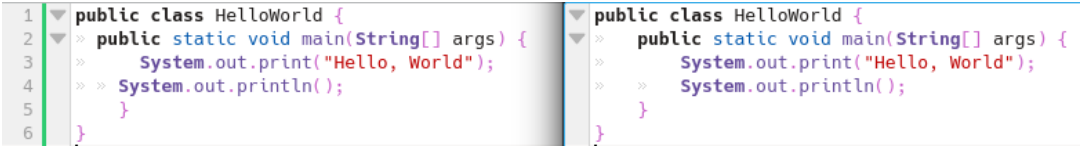
Konvention zur Formatierung und Einrückung

Heinrich Heine Universität Düsseldorf

- 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 mit 4 Leerzeichen
  - → Editor so einstellen, dass Tabulatortaste (links vom Q) 4 Leerzeichen erzeugt

Tabs vs. Spaces

- Einrückung konsequent mit Tabulator oder Leerzeichen
  - → sonst je nach Editor-Einstellung Durcheinander
- 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();
    }
}
```

Kommentare

- 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

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

Lernziele

Heinrich Heine Universität Düsseldorf

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**.

Wichtige Begriffe

System.out.println	javac	java
Standardausgabe	System.out.print	Quellcode
Compiler	Compilezeitfehler	Laufzeitfehler
Einrückung	Block-Kommentar	Zeilen-Kommentar