

# Vorkurs Programmieren

## HTW Berlin SoSe 2016

# Wiederholung

# Wiederholung: Einführung

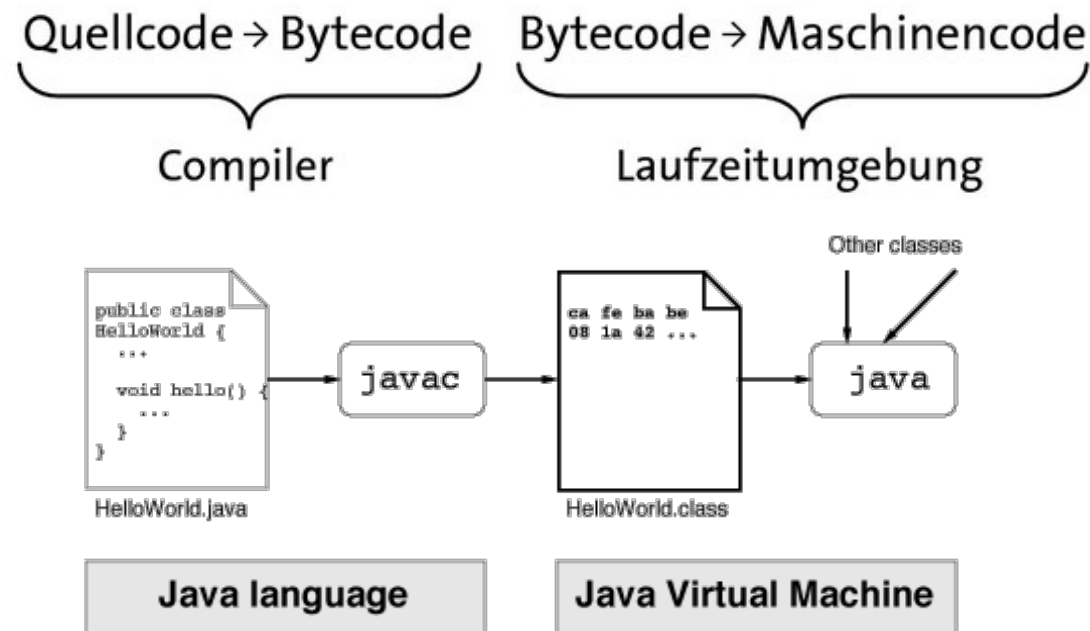
- Programmierung
  - Handwerk der Informatik
  - Testen (und spielen)

# Wiederholung: Einführung

- Hilfe zur Selbsthilfe
  - Google
    - Java ist auch eine Insel
    - Java API
    - Konventionen beachten

# Wiederholung: Code zum Prozess

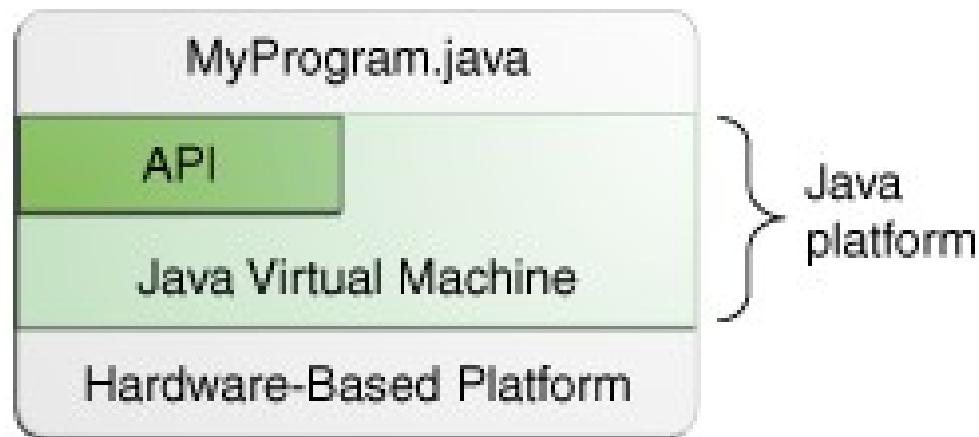
- Compiler: Javac
  - Übersetzt Programmcode in Maschinensprache



- Programmausführung: Prozess
  - zur Laufzeit / Runtime

# Wiederholung: Java

- Java: Programmiersprache
- JVM: Hardware-Abstraktion
- JDK, SDK: Software Development Kit
  - Java SE



# Wiederholung: Hello World

```
public class Hello {
```

```
    public static void main(String[] args) {
```

```
        System.out.println("Hello World");
```

```
    }
```

```
}
```

# Wiederholung: Basis-Datentypen

- 2 Schritte:
  - Deklaration: muss
  - Initialisierung: kann
- Zahlen
  - Typen: int, long, float, double
  - Operatoren: + - \* / %
  - Kurzschreibweisen: +=, ++
  - Konstanten: z.B. Math.Pi



# Wiederholung: Basis-Datentypen

- Buchstaben
  - Typen: String, char
  - Operatoren: + (String-Konkatenation)
- Wahrheitswerte:
  - Typ: Boolean mit True und False
  - Operationen: ! & | == < >

# Wiederholung: Kontrollstrukturen

- Anweisungen:
  - if-else
  - if-elseif-else
  - ? :
  - switch-case
- Schleifen
  - for
  - while
  - do while
- return, continue, break

# Wiederholung: Funktionen

- Aufruf mit Argumenten

```
public static void main(String[] args) {  
    System.out.println(printSleep(4));  
}
```

# Wiederholung: Funktionen

- Aufruf mit Argumenten
- Deklaration mit Parametern

```
private static String printSleep(int hoursToSleep) {  
    do {  
        hoursToSleep--;  
        return hoursToSleep + " more hours to sleep";  
    } while(hoursToSleep > 0);  
}
```

# Wiederholung: Funktionen

- Aufruf mit Argumenten
- Deklaration mit Parametern
- EVA:  $f(x) = y$

```
public static void main(String[] args) {  
    System.out.println(printSleep(4));  
}
```

```
private static String printSleep(int hoursToSleep) {  
    do {  
        hoursToSleep--;  
        return hoursToSleep + " more hours to sleep";  
    } while(hoursToSleep > 0);  
}
```

# Wiederholung: Wie arbeitet ein Programm?

- Folge von Befehlen: Funktionen, Methoden, ... Aufruf
- Verarbeitung von Variablen und Konstanten
- Springen in Schleifen oder je nach Fallunterscheidung
- Code wiederverwenden mittels Funktionen

# Wiederholung: Verständnis

- Terminal: pwd, ls, cat, cd, javac, java
- Wie lege ich ein IntelliJ Projekt an?
- Wo liegt mein IntelliJ Idea Code?
- Was bedeuten rote Markierungen in der IDE?
- Wie starte ich mein Programm aus der IDE?

# Übung: Werte eingeben

- Schreiben Sie ein Programm **Input**, welches den Benutzer nach Text fragt.
- Nutzen Sie dafür die Methode aus der Datei **input.txt**
- Fragen Sie bis zu 10 mal. Bei einer leeren Eingabe soll abgebrochen werden.
- Zusatz: Fragen Sie und erwarten Sie Zahlen als Eingabe.



# Inhalt Vorlesung 2

- 1) Wiederholung und Vertiefung
- 2) Arrays
- 3) Call-by-Value, Call-by-Reference

# Arrays

# Arrays

- Motivation?

```
public class Main {  
    public static void main(String[] args) {  
        int x = 0;  
        int y = 3;  
        int z = 1;  
        printCoordinates(x,y,z);  
    }  
  
    private static void printCoordinates(int x, int y, int z) {  
        System.out.println("Coordinates: "+x+", "+y+", "+z);  
    }  
}
```

# Arrays

- Gruppieren sinnvoll
- Alle Variablen vom selben Typ

```
public class Main {  
    public static void main(String[] args) {  
        int x = 0;  
        int y = 3;  
        int z = 1;  
        printCoordinates(x,y,z);  
    }  
  
    private static void printCoordinates(int x, int y, int z) {  
        System.out.println("Coordinates: "+x+", "+y+", "+z);  
    }  
}
```

# Arrays

- Gruppieren sinnvoll
- Alle Variablen vom selben Typ

```
public class Main {  
    public static void main(String[] args) {  
        int[] coordinate = new int[] {0,3,1};  
        //     int[] point = new int[3];  
        //     point[0] = 3; ...  
        printCoordinates(coordinate);  
    }  
  
    private static void printCoordinates(int[] p) {  
        System.out.println("Coordinates:" + p[0] + "," + p[1] + ","  
            + p[2]);  
    }  
}
```

# Arrays

- Lesen / Schreiben via Index in [ ]
- Start: 0

```
public class Main {  
    public static void main(String[] args) {  
        int[] coordinate = new int[] {0,3,1};  
        //    int[] point = new int[3];  
        //    point[0] = 3; ...  
        printCoordinates(coordinate);  
    }  
  
    private static void printCoordinates(int[] p) {  
        System.out.println("Coordinates:" + p[0] + "," + p[1] + ","  
            + p[2]);  
    }  
}
```

# Arrays

- 2 Arten der Initialisierung
- Helfer: Arrays.toString(myArray)

```
public class Main {  
    public static void main(String[] args) {  
        int[] coordinate = new int[] {0,3,1};  
        //    int[] point = new int[3];  
        //    point[0] = 3; ...  
        printCoordinates(coordinate);  
    }  
  
    private static void printCoordinates(int[] p) {  
        System.out.println("Coordinates:" + p[0] + "," + p[1] + ","  
            + p[2]);  
    }  
}
```

# Arrays

- Feste Anzahl an Elementen
- Standardwerte?

```
public class Main {  
    public static void main(String[] args) {  
        int[] coordinate = new int[] {0,3,1};  
        //    int[] point = new int[3];  
        //    point[0] = 3; ...  
        printCoordinates(coordinate);  
    }  
  
    private static void printCoordinates(int[] p) {  
        System.out.println("Coordinates:" + p[0] + "," + p[1] + ","  
            + p[2]);  
    }  
}
```



# Übung: Liste speichern

- Schreiben Sie ein Programm **ListInput**, welches den Benutzer nach Zahlen fragt und in einem Array speichert.
- Nutzen Sie dafür die Methode aus der Datei **input.txt**
- Speichern Sie bis zu 10 Werte und geben Sie dann die Liste aus.
- Zusatz: Geben Sie die Liste nur aus bis zur letzten Eingabe.

# Wertübergabe

- Wann werden Werte übergeben?
  - Blöcke
  - Funktionen

# Call-by-Value

- Basis-Datentypen passen i.d.R. in eine Speicherzelle (64 bit)
- direkte Übergabe von dem Wert, Bsp.: int
- Was passiert bei größeren Werten? Bsp.: int[]
- Demo...

# Call-by-Reference

- Übergabe von Referenz auf Daten
- Können verändert werden unabhängig von return
- Bsp.: `swap(int[] arr, int i, int j)`

# Übung: Liste suchen

- Schreiben Sie ein Programm **ListFind**, welches den Benutzer nach einer Zahl fragt und in einem Array speichert.
- Nutzen Sie dafür die Methode aus der Datei **input.txt**
- Speichern Sie bis zu 10 Werte und geben Sie das größte Element der Liste aus.
- Zusatz: Überprüfen Sie ob die Liste **sortiert** ist.
- Zusatz: Sortieren Sie die Liste

# Übung: Taschenrechner

- Schreiben Sie ein Programm **Calc**, welches zwei ganze Zahlen akzeptiert und diese verrechnet. Das Programm soll mit “**java Calc 1 - 2**” aufrufbar sein.
- Erweitern Sie ihr Programm um die anderen Rechenoperationen: +, \*, / und %
- Extrahieren Sie die Rechenoperationen in eine eigene Funktion, welche auf Ein- und Ausgabe getestet werden soll (EVA).
- Die Funktion soll einen String zurück geben mit dem Ergebnis oder einer Fehlernachricht, z.B. Operator nicht gefunden.
- Testen Sie auf Division durch 0.