

# Vorkurs Programmieren

## HTW Berlin SoSe 2016

# Organisatorisches

- Vorlesung + Übung
  - Mario Neises
  - [Mario.neises@student.htw-berlin.de](mailto:Mario.neises@student.htw-berlin.de)
  - <https://github.com/mn-io> (hello-java)
- Tutorium
  - Laura Laugwitz

# Organisatorisches

- 09:45 – 11:15 Uhr: Vorlesung + Übung
- 11:30 – 13:00 Uhr: Vorlesung + Übung
- 13:45 – 15-15 Uhr: Tutorium
- 15:30 – 17:00: Uhr Selbstlernzeit

# Einführung

# Einführung

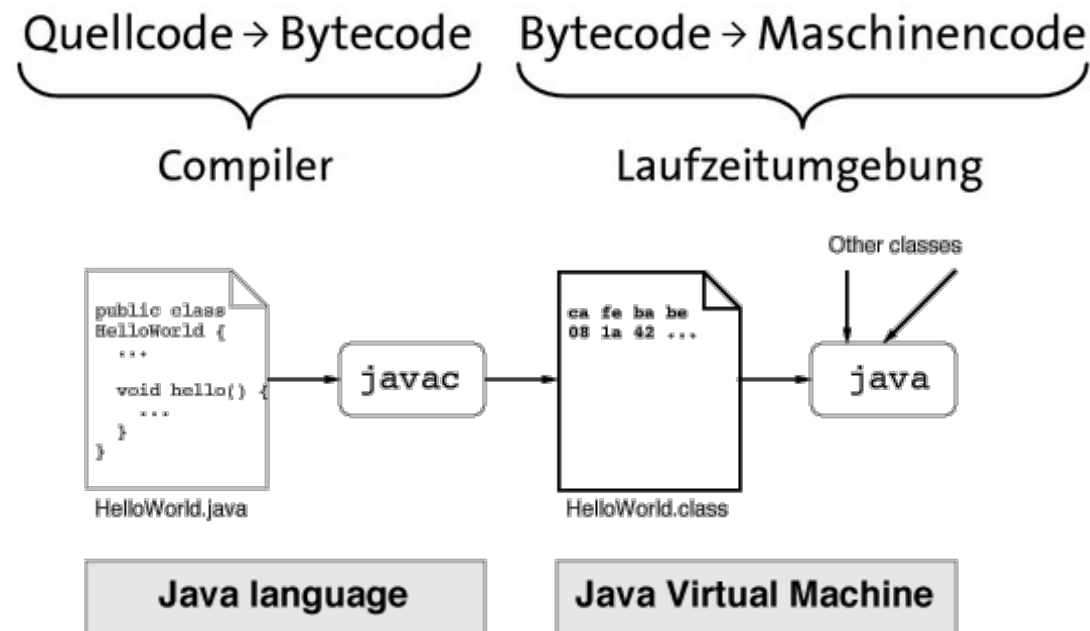
- Programmierung
  - Handwerk der Informatik
  - Testen (und spielen)

# Einführung

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

# Vom Code zum Prozess

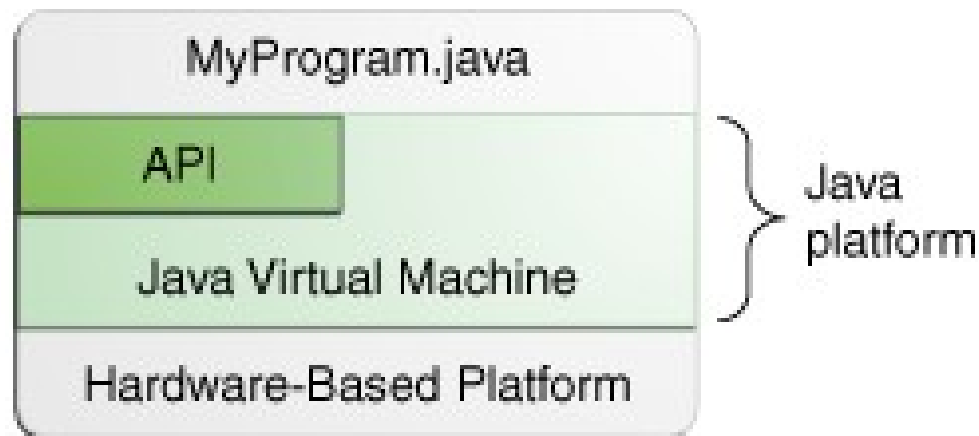
- Compiler: Javac
  - Übersetzt Programmcode in Maschinensprache



- Programmausführung: Prozess
  - zur Laufzeit / Runtime

# Java

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





# Hello World

```
public class Hello {
```

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

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

```
    }
```

```
}
```

# Basis-Datentypen

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

# Basis-Datentypen

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

# Kontrollstrukturen

- Anweisungen:
  - if-else
  - if-elseif-else
  - switch-case
- return

```
boolean condition = true;  
if (condition) {  
    System.out.println("I am true");  
} else {  
    System.out.println("I am false");  
}
```

# Kontrollstrukturen

- Schleifen

- for

```
for (int i = 0; i < 10; i++) {  
    System.out.println(i);  
}
```

- while

- do while

```
int j = 0;  
while (j < 10) {  
    System.out.println(j);  
    j++;  
}
```

- return, continue, break

# Funktionen

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

```
public static void main(String[] args) {  
    int result = add(4, 3);  
    System.out.println(result);  
}
```

```
private static int add(int first, int second) {  
    int sum = first + second;  
    return sum;  
}
```

# Übung: Taschenrechner

- Schreiben Sie ein Programm mit den Funktionen `add()`, `subtract()`, `divide()`, `multiply()`
  - Welcher Rückgabewert hat `divide`?
- Das Ergebnis soll am Ende von `main()` via `System.out.println()` ausgegeben werden.
- Nehmen Sie ein `if-elseif-else`-Konstrukt um die Operationen zu bestimmen.

# Zusammenfassung: 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



# Fragen zum Verständnis

- Terminal:
  - pwd, cd, ls, cat
  - 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.

# 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.
- Zusatz: Fragen Sie und erwarten Sie Zahlen als Eingabe.

Tag 4

# Wertübergabe

- Demo / Erklärung:
  - Zuweisung von Variablen
- Übung:
  - Variablen tauschen mit/ohne Funktion
  - Elemente in Array tauschen mit/ohne Funktion
  - Was fällt auf?

# Call-by-Value

- Basis-Datentypen passen i.d.R. in eine Speicherzelle (64 bit)
- Funktionen bilden eigenen Kontext  
d.h.: Übergabe der Parameter bei Aufruf
- direkte Übergabe von dem Wert, Bsp.: int
- Was passiert bei größeren Werten? Bsp.: int[]

# Call-by-Reference

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

# Klassen und Objekte

- Neben primitiven Datentypen eine spezielle Art Datentyp
- In Java ist alles andere vom Datentyp einer Klasse
- Bsp.:
  - Klasse `String`
  - `String hi = "Hello";`
  - eigentlich: `String hi = new String("Hello");`

# Klassen und Objekte

- Klassen können Methoden haben, welche diese "gruppieren"
- Demo: Calculator-Klasse
- Was können Klassen?
  - haben Methoden, z.B.: `System.out.println()`
  - `public` und `privat` regeln Zugriff



# Klassen und Objekte

- Instanz von einer Klasse sind Objekte  
d.h.: **new** `String()` erzeugt ein Objekt/Instanz
- Was können Objekte?
  - ein Objekt ist konkret, eine Klasse allgemein
  - haben Methoden, wie `"hello".isEmpty()`
  - Existieren nur, wenn eine Referenz (=Variable) zu diesem Objekt existiert.
  - Keine `static`-Kennzeichnung bei Methoden
  - Vorteil gegenüber Klassen: **this** (Selbstreferenz)

# Klassen und Objekte

- Klassen können Methoden haben, welche diese "gruppieren"
- Demo: Calculator-Klasse
- Was können Klassen?
  - haben Methoden, wie `System.out.println()`
  - `public` und `privat` regeln Zugriff
  - Existieren immer
  - Methoden sind mit `static` gekennzeichnet

# Übung: Taschenrechner2

- Verändern Sie das Programm **Calculator**, sodass es keine main()-Methode gibt.
- Rufen Sie wie gewohnt die Methoden von einer main()-Methode auf, welche in einer anderen Klasse liegt.
- Zusatz: entfernen sie das Wort static in Calculator, erzeugen Sie ein Objekt von Calculator und speichern Sie das letzte Ergebnis in dem Objekt.

# 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