



## Einführung



## Warum programmieren?

Mensch & Compiler - Das Dreamteam

Java

## Warum Java?

Halb kompiliert & Halb interpretiert

## Das erste Programm

Kommentare

Textausgabe

Steuerzeichen

## Vom Quellcode zum fertigen Programm

### Ein Java Programm schreiben

### Ein Java Programm ausführen

Fehlermeldungen

Fehlerarten

## Stil und Formatierung



- ▶ Lösen von Problemen mit Hilfe von Computern durch einen eigenen Algorithmus
- ▶ Problem: Computer sind rein-mathematische Geräte

Computer, addiere bitte die Zahlen 4 und 5 und teile anschließend das Ergebnis durch 3.

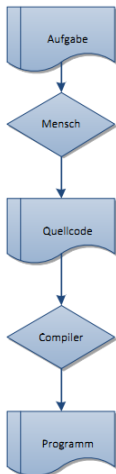
- ▶ Formalismus den Computer und Menschen verstehen?
  - ▶  $\Rightarrow$  Die Programmiersprache

# Warum programmieren?

## Mensch & Compiler - Das Dreamteam



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



- ▶ **Der Mensch** beschreibt die Aufgabe mittels Programmiersprache
- ▶ **Der Compiler** wandelt den Quellcode in ein ausführbares Programm um

# Warum programmieren?

## Java



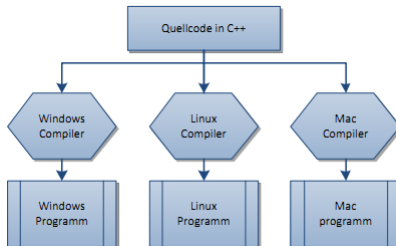
TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- ▶ Wir verwenden Java eine imperative und objektorientierte Sprache
- ▶ Arbeitssprache der ersten zwei Semester
- ▶ Aus was besteht eine Programmiersprache?
  - ▶ Java besteht aus einem Grundwortschatz aus festen, reservierten Begriffen (**Syntax**) und Regeln wie diese angeordnet werden (**Grammatik**)
  - ▶ Java bietet Grundrechenarten, vergleiche und Methoden zum Einlesen und Ausgeben von Daten

# Warum Java?



- ▶ Ein C/C++/Delphi/- Programm muss vom Quellcode in den Maschinencode für das jeweilige Betriebssystem kompiliert werden
- ▶ Eher ungünstig, wenn das Zielsystem unbekannt ist
- ▶ Java funktioniert hier anders



# Warum Java?

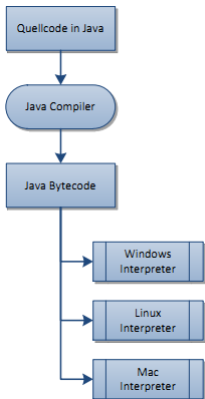
## Halb kompiliert & Halb interpretiert



- ▶ Java transformiert den Quellcode in sogenannten Bytecode. Dieser ist für alle Systeme gleich
- ▶ Anschließend wird dieser Bytecode auf dem Zielsystem interpretiert

### Hinweis

Durch den Zwischencode können Programme schneller ausgeführt werden, weil dieser schon speziell aufbereitet wurde





```
public class HelloWorld {  
    public static void main(String[] args) {  
        // Gebe Hello World aus  
        System.out.println("Hello World!");  
    }  
}
```

- ▶ **Zeile 1:** Definition der Klasse HelloWorld
- ▶ **Zeile 2:** Definition der *main* Funktion
- ▶ **Zeile 4:** Funktion aus der *Java System Library* wird aufgerufen und **Hello World** ausgegeben

## Wichtig

Jedes Statement muss mit einem Semikolon (;) abgeschlossen werden!



# Das erste Programm

## Kommentare



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- ▶ In Java gibt es zwei verschiedene Arten von Kommentaren
- ▶ **Zeilenkommentare** //
  - ▶ Beginnen bei // und enden am Zeilenende
- ▶ **Blockkommentare** /\* ... \*/

```
// Das ist der erste Kommentar
```

```
/*  
// Calculate e :)  
double e = 0;  
for(int i = 0; i <= Integer.MaxValue; i++) {  
    double temp = 1/fak(i);  
    e += temp;  
}  
*/
```

# Das erste Programm

## Textausgabe



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- ▶ Mithilfe von **System.out.print()** und **System.out.println()** kann man Text auf der Konsole ausgeben
- ▶ Unterschiede:

```
System.out.print("Hallo␣");  
System.out.print("wie␣");  
System.out.print("geht's?");  
//Ausgabe Hallo wie geht's?
```

```
System.out.println("Hallo␣");  
System.out.println("wie␣");  
System.out.println("geht's?");  
//Ausgabe  
// Hallo  
// wie  
// geht's?
```

## Merke

`System.out.println()` ; erzeugt einen Zeilenumbruch nach der Ausgabe

# Das erste Programm

## Steuerzeichen



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Symbol	Wirkung
\b	Backspace
\n	Zeilenumbruch
\t	Tabulator
\"	Anführungszeichen
\\	Backslash

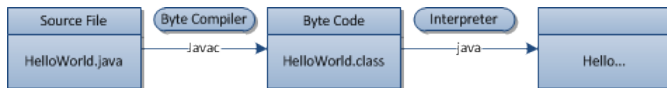
```
System.out.print("Sehr geehrte Studenten und Studentinnen,\n");  
System.out.print("willkommen zum \"Programmervorkurs 2013\"!");
```

/\* Ausgabe

```
Sehr geehrte Studenten und Studentinnen,  
willkommen zum "Programmervorkurs 2013!"  
*/
```

# Vom Quellcode zum fertigen Programm

- ▶ Dateiname des Quellprogramms hat die Endung **.java**
- ▶ Basisname der Datei ist der Klassenname (HelloWorld)



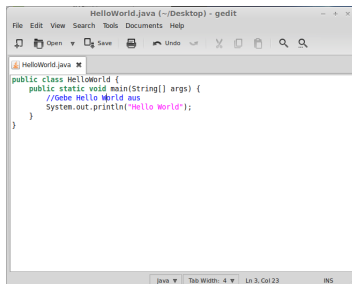
- ▶ **Kompilieren:** `javac HelloWorld.java` erzeugt Bytecode `HelloWorld.class`
- ▶ **Ausführen:** `java HelloWorld`
- ▶ Bytecode ist überall lauffähig, aber passender *Java-Interpreter* wird benötigt

# Ein Java Programm schreiben

## Live-Coding



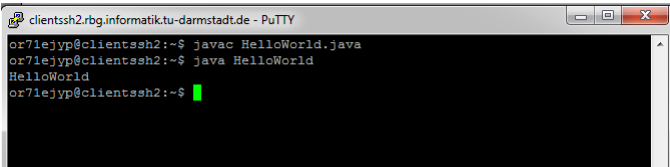
TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



```
HelloWorld.java (~/.Desktop) - gedit
File Edit View Search Tools Documents Help
[Icons] Open Save Undo Cut Copy Paste Find
HelloWorld.java
public class HelloWorld {
    public static void main(String[] args) {
        //Gebe Hello World aus
        System.out.println("Hello world");
    }
}
java Tab Width: 4 Ln 3, Col 23 INS
```

- ▶ Programm wird in einem herkömmlichen Editor geschrieben
- ▶ Poolraumrechner: gedit Editor

# Ein Java Programm ausführen



```
clientssh2.rbg.informatik.tu-darmstadt.de - PuTTY
or71ejyp@clientssh2:~$ javac HelloWorld.java
or71ejyp@clientssh2:~$ java HelloWorld
HelloWorld
or71ejyp@clientssh2:~$
```

- ▶ Kompilieren der Datei mittels **javac datei.java** in Bytecode
- ▶ Programm mit **java datei** starten

## Hinweis

Der Befehl **java** erwartet den Dateiname ohne den Suffix **.java**!

# Ein Java Programm ausführen

## Fehlermeldungen



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- ▶ Java teilt Fehler beim kompilieren mit
- ▶ Nicht immer ist der Fehler in der Zeile, die der Java Compiler erkennt

```
clientssh2.rbg.informatik.tu-darmstadt.de - PuTTY
or71ejyp@clientssh2:~$ javac HelloWorld.java
HelloWorld.java:3: ';' expected
    System.out.print("HelloWorld\n")
                   ^
1 error
or71ejyp@clientssh2:~$
```

## Frage

Welcher Fehler wurde gemacht?



### ▶ Lexikalische Fehler

- ▶ Beispielsweise Tippfehler
- ▶ `Sistem.out.print("Lexikalischer_Fehler");`

### ▶ Syntaktische Fehler

- ▶ Falsche Klammern
- ▶ Semikolon vergessen

### ▶ Semantische Fehler

- ▶ Vom Interpreter zur Laufzeit gemeldet
- ▶ Beispielsweise Division durch Null





- ▶ Gut formatierter Quellcode erhöht die Lesbarkeit
- ▶ Verwendet aussagekräftige Kommentare

```
// Bad code
public static void main( String []  args)      {
int b=6;int c=9;}

// Good code
public static void main( String []  args){
    int b = 6;
    int c = 9;
}
```