

Hello World!

Javakurs 2012, 1. Vorlesung

Theresa Enghardt

basierend auf der Vorlage von
Mario Bodemann und Sebastian Dyroff

www.freitagrunde.org

4. März 2012



This work is licensed under the *Creative Commons Attribution-ShareAlike 3.0 License*.

Ziele des Kurses

- ▶ Für euch:
 - ▷ Programmieren mit Java lernen
 - ▷ Probleme im Code finden und lösen
 - ▷ Keine Anwesenheitspflicht, kein Stress
- ▶ Für uns:
 - ▷ Üben von Vorträgen - Feedback
 - ▷ Arbeit als Tutor/in
 - ▷ Wissen weitergeben
- ▶ Zusammen: Viel, viel Spaß haben ...



1



- ▶ Studentische Initiative der Fakultät 4 - Studis wie du und ich
- ▶ {Diplom, Bachelor, Master} {ET, TI, Inf, WInf}
- ▶ Java-/C-Kurs, TechTalks, Linux Install Partys
- ▶ Gremienarbeit, Einführungswoche, Klausurensammlung
- ▶ Kickerturniere, LAN-Partys
- ▶ www.freitagsrunde.org, FR 5518
- ▶ Sitzung: Freitags 14 Uhr im FR 5516

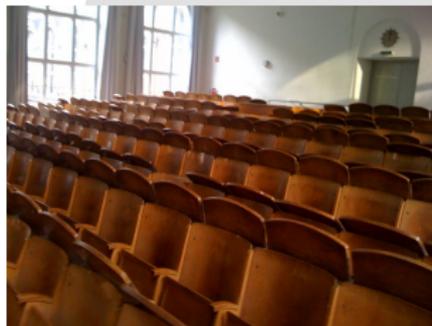
- 1 Organisatorisches
- 2 Schreiben, Kompilieren, Ausführen
- 3 Variablen
- 4 Operatoren
- 5 Fallunterscheidungen
- 6 Lesbarkeit
- 7 Typische Fehler
- 8 Zusammenfassung



4!

Ablauf

	Mo	Di	Mi	Do	Fr
10:00	<i>Hello World</i>	<i>Methoden</i>	Übung	<i>API</i>	<i>Vererbung</i>
11:30	Übung	Übung		Übung	Übung
13:15	Mittagspause				
14:15	<i>Schleifen</i>	Übung	Übung	<i>Kapselung</i>	Übung
15:30	Übung			Übung	



MA005 (Mo/Di), MA 042 (Do/Fr)



TEL 106 / TEL 206

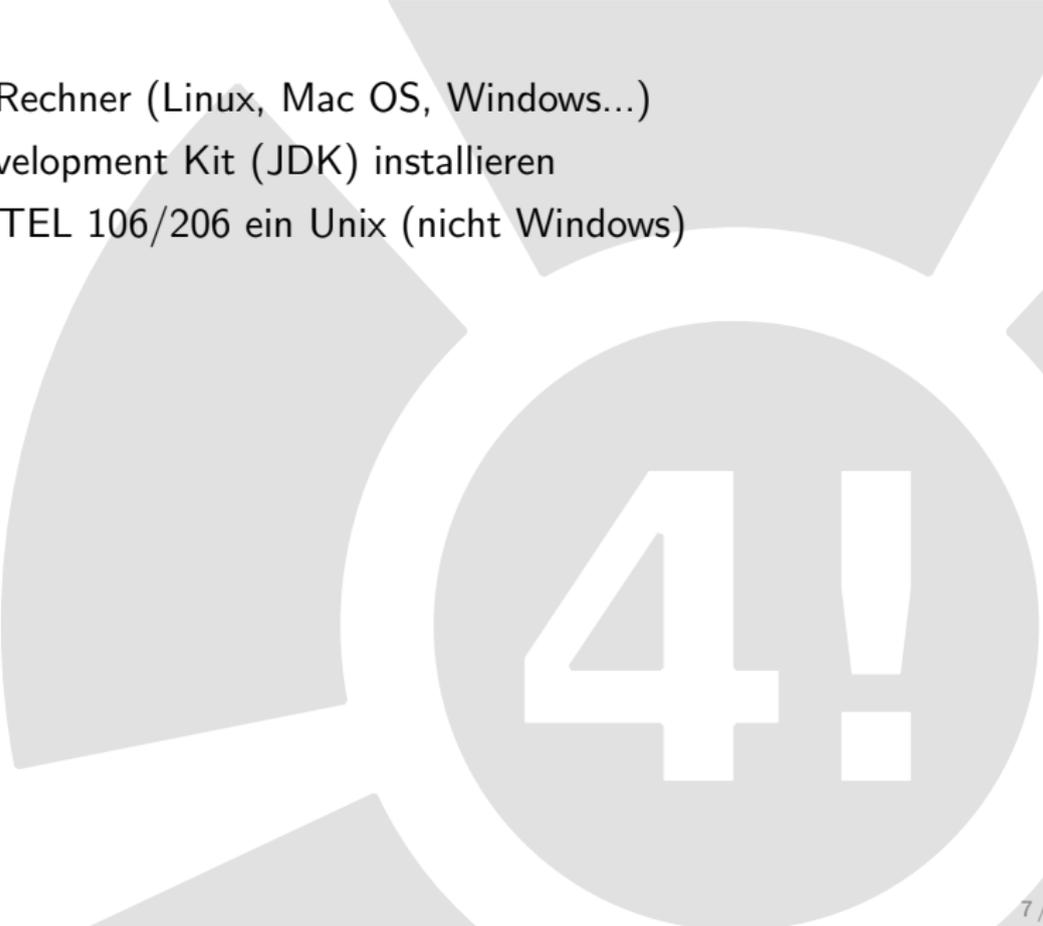
- 1 Organisatorisches
- 2 Schreiben, Kompilieren, Ausführen**
- 3 Variablen
- 4 Operatoren
- 5 Fallunterscheidungen
- 6 Lesbarkeit
- 7 Typische Fehler
- 8 Zusammenfassung



4!

Schreiben: Eure Arbeitsumgebung

- ▶ Eigener Rechner (Linux, Mac OS, Windows...)
- ▶ Java Development Kit (JDK) installieren
- ▶ oder im TEL 106/206 ein Unix (nicht Windows)

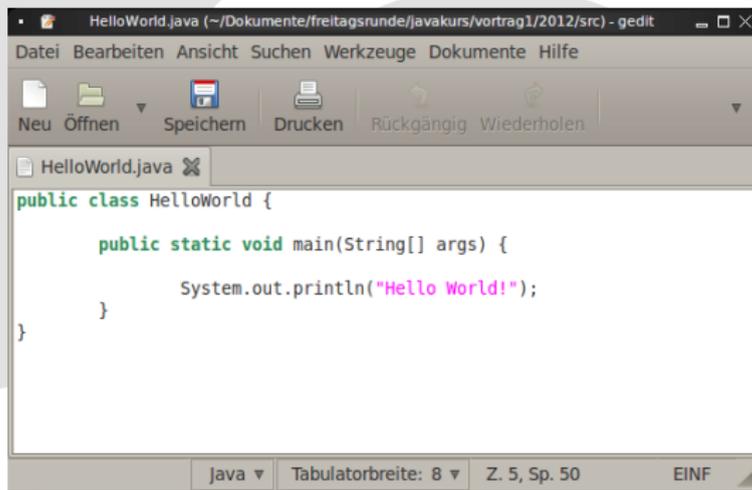


4!

Schreiben: Eure Arbeitsumgebung

- ▶ Eigener Rechner (Linux, Mac OS, Windows...)
- ▶ Java Development Kit (JDK) installieren
- ▶ oder im TEL 106/206 ein Unix (nicht Windows)
- ▶ einfacher Texteditor ³

- ▶ **gedit** (Linux)
- ▶ **nano** (Linux)
- ▶ **notepad++** (Windows)



The screenshot shows a window titled "HelloWorld.java (~/Dokumente/freitagsrunde/javakurs/vortrag1/2012/src) - gedit". The window has a menu bar with "Datei", "Bearbeiten", "Ansicht", "Suchen", "Werkzeuge", "Dokumente", and "Hilfe". Below the menu bar is a toolbar with icons for "Neu", "Öffnen", "Speichern", "Drucken", "Rückgängig", and "Wiederholen". The main editing area contains the following Java code:

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

The status bar at the bottom shows "java", "Tabulatorbreite: 8", "Z. 5, Sp. 50", and "EINF".

³Entwicklungsumgebungen, z.B. Eclipse, später

Schreiben: Hello World

Listing 1: src/HelloWorld.java

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

- ▶ Datei, darin eine Klasse, die genauso heißen muss (ohne .java)
- ▶ Klasse: später genaue Definition, jetzt einfach "Programm"
- ▶ Beim Starten des Programs wird die "main"-Methode ausgeführt
- ▶ In der Methode stehen Befehle, die nacheinander abgearbeitet werden
- ▶ Jeder Befehl endet mit einem Semikolon

Kompilieren

Der Compiler übersetzt den Quellcode in ein ausführbares Programm.

javac ist der **Java Compiler**.



- ▶ Quellcode speichern (Dateiname entspricht Klassenname + ".java")
- ▶ Shell öffnen (Kommandozeile)
 - ▷ z.B. gnome-terminal (Linux), Terminal (Mac OS), cmd (Windows)

```
$ javac HelloWorld.java  
$
```

Kompilieren eines Javaprogramms

```
$ ls -l
-rw-r--r-- 1 theri 411 Feb 28 16:13 HelloWorld.class
-rw-r--r-- 1 theri 111 Feb 28 16:18 HelloWorld.java
$
```

Quellcode und Bytecode

- ▶ Kompilieren erzeugt .class-Dateien, sog. Bytecode
- ▶ Bytecode kann mit einer **Java Virtual Machine** ausgeführt werden
- ▶ Bytecode ist maschinenunabhängig, d.h. unter jedem Betriebssystem gleich

```
$ java HelloWorld  
Hello World!  
$
```

Ausgabe unseres Programms

- ▶ **java** ist die Java Virtual Maschine
- ▶ Als Parameter wird der Klassenname übergeben
- ▶ Aufruf **ohne .class!**
- ▶ Die Ausgabe ist auf der Konsole zu sehen

Listing 5: src/HelloPlanets.java

```
1 public class HelloPlanets {
2
3     public static void main(String [] args) {
4         System.out.println(" Hello Mercure!");
5         System.out.println(" Hello Venus!");
6         System.out.println(" Hello Earth!");
7         System.out.println(" Hello Mars!");
8         System.out.println(" Hello Jupiter!");
9     }
10 }
```

- ▶ Befehle werden der Reihe nach abgearbeitet
- ▶ Name der class = Dateiname (ohne .java)

Datei Bearbeiten Ansicht Suchen Terminal Hilfe

```
theri@anghammarad:~/java$ ls -l
insgesamt 4
-rw-r--r-- 1 theri theri 270 2012-02-28 17:11 HelloPlanets.java
theri@anghammarad:~/java$
```

Datei Bearbeiten Ansicht Suchen Terminal Hilfe

```
theri@anhammarad:~/java$ ls -l
insgesamt 4
-rw-r--r-- 1 theri theri 270 2012-02-28 17:11 HelloPlanets.java
theri@anhammarad:~/java$ javac HelloPlanets.java
theri@anhammarad:~/java$
```

Datei Bearbeiten Ansicht Suchen Terminal Hilfe

```
theri@anghammarad:~/java$ ls -l
insgesamt 4
-rw-r--r-- 1 theri theri 270 2012-02-28 17:11 HelloPlanets.java
theri@anghammarad:~/java$ javac HelloPlanets.java
theri@anghammarad:~/java$ ls -l
insgesamt 8
-rw-r--r-- 1 theri theri 553 2012-02-28 17:18 HelloPlanets.class
-rw-r--r-- 1 theri theri 270 2012-02-28 17:11 HelloPlanets.java
theri@anghammarad:~/java$
```

Datei Bearbeiten Ansicht Suchen Terminal Hilfe

```
theri@anghammarad:~/java$ ls -l
insgesamt 4
-rw-r--r-- 1 theri theri 270 2012-02-28 17:11 HelloPlanets.java
theri@anghammarad:~/java$ javac HelloPlanets.java
theri@anghammarad:~/java$ ls -l
insgesamt 8
-rw-r--r-- 1 theri theri 553 2012-02-28 17:18 HelloPlanets.class
-rw-r--r-- 1 theri theri 270 2012-02-28 17:11 HelloPlanets.java
theri@anghammarad:~/java$ java HelloPlanets
Hello Mercure!
Hello Venus!
Hello Earth!
Hello Mars!
Hello Jupiter!
theri@anghammarad:~/java$
```

- 1 Organisatorisches
- 2 Schreiben, Kompilieren, Ausführen
- 3 Variablen**
- 4 Operatoren
- 5 Fallunterscheidungen
- 6 Lesbarkeit
- 7 Typische Fehler
- 8 Zusammenfassung



4!

Variablen

- ▶ "Kiste", in der Werte gespeichert werden können
- ▶ Hat einen Namen und einen Typ
- ▶ z.B. **int** = Integer = ganze Zahl



Listing 6: src/Variable.java

```
1  int a = 23;
2  System.out.println("Eine Zahl: " + a);
3
4  a = a + 5;
5  System.out.println("Neuer Wert: " + a);
```

```
$ javac Variable.java
$ java Variables
Eine Zahl: 23
Neuer Wert: 28
$
```

Ausgabe Variable

- ▶ Kompilieren und Ausführen
- ▶ Der Wert der Variablen wird auf die Konsole geschrieben
- ▶ Text und Zahl werden korrekt zusammengesetzt

Listing 8: src/Variable2.java

```
1 public class Variable2 {  
2     public static void main(String [] args) {  
3  
4         double aktuelleTemperatur = 22.5;  
5         boolean regnetEsGerade = false;  
6         String aktuellerMonat = "Mai";  
7     }  
8 }
```

- ▶ **double** - Fließkommazahl
- ▶ **boolean** - Wahrheitswert
- ▶ **String** - Zeichenkette

Datentypen

```
boolean result = false;  
  
int age = 20;  
  
double height = 1.75;  
  
String message = "Javakurs";
```

→ Wertebereiche im Auge behalten!

Es gibt zwar noch mehr Datentypen, aber dies sind erstmal die wichtigsten.

Wertebereich

{true, false}

{-2147483648 ...
2147483647}

{ $\pm 4,9 \cdot 10^{-324}$...
 $\pm 1,7977 \cdot 10^{308}$ }

{ ""... endlich }

Konventionen

- ▶ Variablennamen werden im sogenannten camelCase geschrieben
- ▶ Name ist Beschreibung
- ▶ Jeder Wortanfang groß



Variablen: Erster Buchstabe klein
regnetEsGerade

Klassen: Erster Buchstabe groß
ArrayIndexOutOfBoundsException

- ▶ Kurze und aussagekräftige Namen verwenden

- 1 Organisatorisches
- 2 Schreiben, Kompilieren, Ausführen
- 3 Variablen
- 4 Operatoren**
- 5 Fallunterscheidungen
- 6 Lesbarkeit
- 7 Typische Fehler
- 8 Zusammenfassung



4!

Listing 10: src/Operators.java

```
1 public class Operators {  
2     public static void main(String [] args) {  
3  
4         int a, b;  
5         a = 10;  
6         b = 2;  
7  
8         int multi = a * b;  
9         int average = (a + b) / 2;  
10    }  
11 }
```

- ▶ Zeile 4: Variablen deklariert (= bekannt gegeben)
- ▶ Zeile 5 und 6: Variablen definiert (= Werte zugewiesen)
- ▶ Auswertung von links nach rechts
- ▶ Es gelten die üblichen Rechenregeln

Listing 11: src/Operators2.java

```
1 public class Operators2 {  
2     public static void main(String [] args) {  
3  
4         double aktuelleTemperatur = 19.5;  
5         boolean regnetEsGerade = false;  
6         boolean istEsKalt = aktuelleTemperatur < 20;  
7  
8         System.out.println("Ist es kalt und regnet es? " +  
9             (istEsKalt && regnetEsGerade));  
10    }  
}
```

- ▶ istEsKalt wird erst deklariert, später definiert
- ▶ Anwendung des Vergleichsoperators, Ergebnis ist boolean
- ▶ Ausdruck in Klammern wird zuerst ausgewertet

```
$ javac Operators2.java  
$ java Operators2  
Ist es kalt und regnet es? false  
$
```

Ausgabe Variable

- ▶ boolean wird automatisch in einen String umgewandelt

Logische Operatoren

&&	UND
	ODER
^	ODER (exklusiv)
!	NICHT

Vergleichsoperatoren

<	kleiner
<=	kleiner gleich
>	größer
>=	größer gleich
==	Gleichheit
!=	Ungleichheit

Arithmetische Operatoren

+	Addition
-	Subtraktion
/	Division
*	Multiplikation
%	Modulo (Rest bei Division)

4!

Inhaltsverzeichnis

- 1 Organisatorisches
- 2 Schreiben, Kompilieren, Ausführen
- 3 Variablen
- 4 Operatoren
- 5 Fallunterscheidungen**
- 6 Lesbarkeit
- 7 Typische Fehler
- 8 Zusammenfassung



4!

Listing 13: src/Entscheidung.java

```
1 public class Entscheidung {
2     public static void main(String [] args) {
3
4         boolean regnetEsGerade = false;
5
6         if (regnetEsGerade) {
7             System.out.println("Bleib lieber drinnen.");
8         }
9         else {
10            System.out.println("Geh an die frische Luft.");
11        }
12    }
13 }
```

```
$ javac Entscheidung.java  
$ java Entscheidung  
Geh an die frische Luft.  
$
```

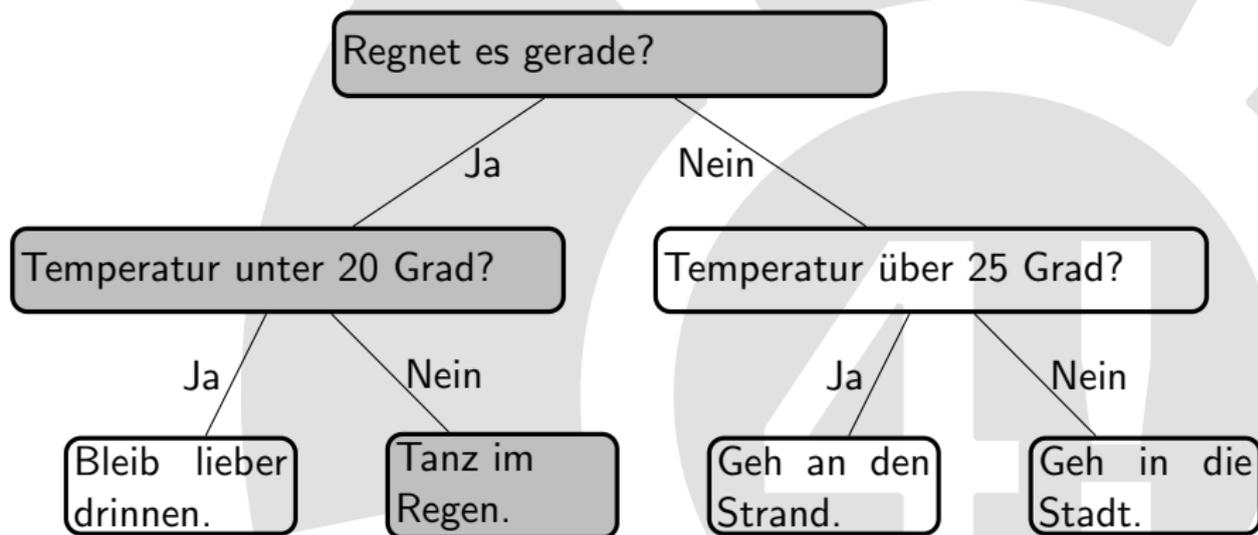
Ausgabe Entscheidung

- ▶ Bedingung `regnetEsGerade` war `false`
- ▶ Code nach dem `if` wurde daher nicht ausgeführt
- ▶ Dafür wurde der Code nach dem `else` ausgeführt

Komplexere Entscheidung

Regnet es: Ja

Temperatur: 23,5 Grad Celsius



Listing 15: src/Entscheidung2.java

```
1  double aktuelleTemperatur = 23.5;
2  boolean regnetEsGerade = true;
3
4  if (regnetEsGerade) {
5      if (aktuelleTemperatur < 20) {
6          System.out.println("Bleib lieber drinnen.");
7      } else {
8          System.out.println("Geh im Regen tanzen.");
9      }
10 }
11 else if (aktuelleTemperatur > 25){
12     System.out.println("Geh an den Strand.");
13 } else {
14     System.out.println("Geh in die Stadt.");
15 }
16 }
```

```
$ javac Entscheidung2.java
$ java Entscheidung2
Geh im Regen tanzen.
$
```

Ausgabe Entscheidung2

- ▶ Programm folgt einem Zweig und führt dessen Code aus
- ▶ Je nach Variablenbelegung kann es bei der nächsten Ausführung ein anderer sein
- ▶ Problem: Viele Optionen → Unübersichtlicher Code

Switch Case

Listing 17: src/Entscheidung3.java

```
1 public class Entscheidung3 {  
2     public static void main(String [] args) {  
3  
4         int jahr = 2012;  
5  
6         switch(jahr) {  
7             case 1999: System.out.println("Altes Jahrtausend");  
8                 break;  
9             case 2000: System.out.println("Jahr-2000-Bug");  
10                break;  
11            case 2012: System.out.println("Ende der Welt");  
12                break;  
13            default: System.out.println("Nichts los...");  
14        }  
15    }  
16 }
```

- 1 Organisatorisches
- 2 Schreiben, Kompilieren, Ausführen
- 3 Variablen
- 4 Operatoren
- 5 Fallunterscheidungen
- 6 Lesbarkeit**
- 7 Typische Fehler
- 8 Zusammenfassung



4!

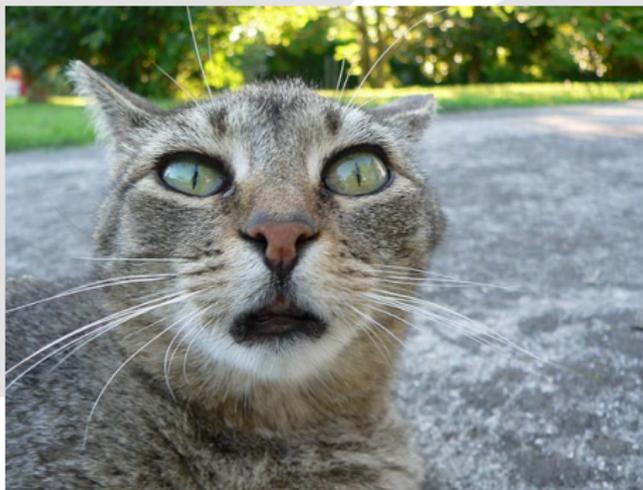
Listing 18: src/KeinKommentar.java

```
1 public class KeinKommentar{public static
2 void main(String [] args){int b=5;if (b%2==1)
3 {System.out.println("rate");}
4 else{System.out.println("mal");}}}
```

Kein Kommentar...

Listing 19: src/KeinKommentar.java

```
1 public class KeinKommentar{public static  
2 void main(String [] args){int b=5;if (b%2==1)  
3 {System.out.println("rate");}  
4 else{System.out.println("mal");}}}
```



Listing 20: src/MitKommentar.java

```
1 public class MitKommentar {
2     /* Diese Klasse soll beispielhaft zeigen,
3        wie in Java ein- oder mehrzeilige
4        Kommentare realisiert werden. */
5     public static void main( String[] args) {
6         int b = 5;
7         if (b % 2 == 1) {
8             // Ausgabe, falls b durch 2 nur mit Rest teilbar
9             // ist
10            System.out.println("rate");
11        }
12        else {
13            // Ausgabe, falls b eine ganze Zahl ist
14            System.out.println("mal");
15        }
16    }
```

Vorteile des Kommentierens

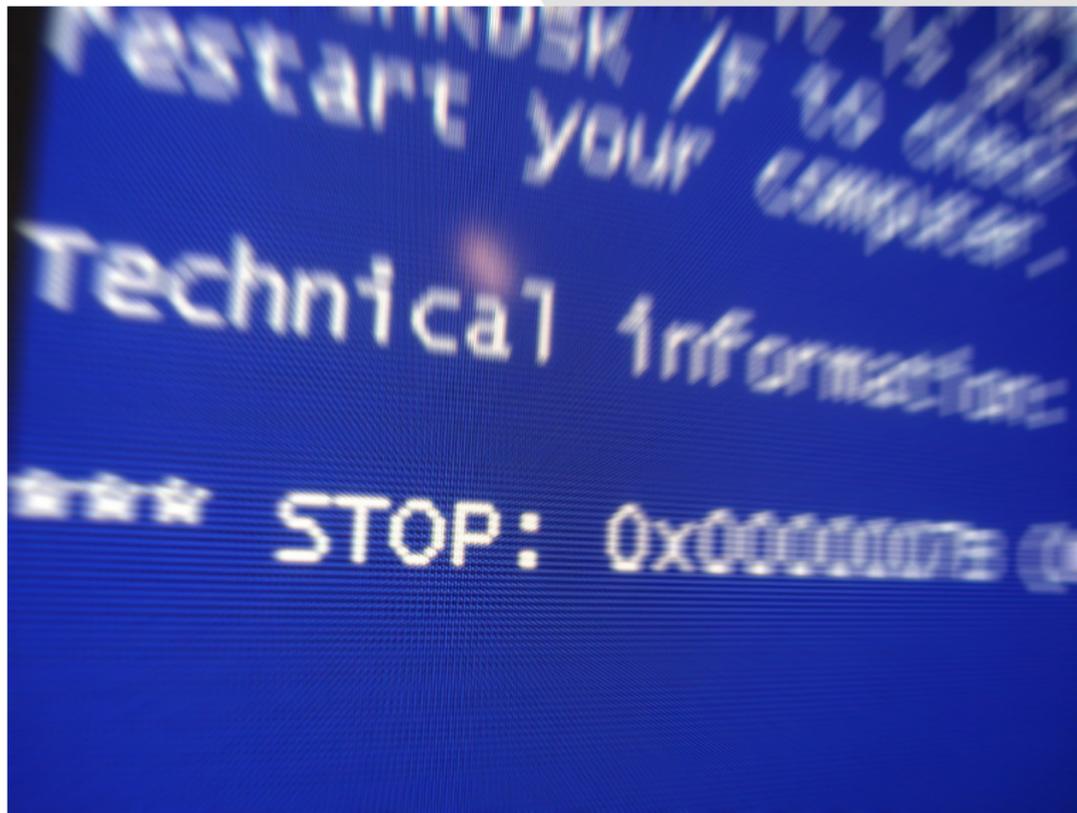
- ▶ Einrückung für Lesbarkeit
- ▶ Kommentare zur Erklärung einzelner Zeilen
 - ▷ Bitte über der betreffenden Zeile
 - ▷ Nicht übertreiben
- ▶ Hilfestellung für eure Mitmenschen... auch für jene, die korrigieren ;)
- ▶ Code, der grade kurz nicht ausgeführt werden soll?

→ Nicht löschen, auskommentieren!

- 1 Organisatorisches
- 2 Schreiben, Kompilieren, Ausführen
- 3 Variablen
- 4 Operatoren
- 5 Fallunterscheidungen
- 6 Lesbarkeit
- 7 Typische Fehler**
- 8 Zusammenfassung



4!



4

Listing 21: src/error/FehlerEins.java

```
1 public class Fehler1 {  
2     public static void main(String [] args) {  
3         System.out.println("Hello World")  
4     }  
5 }
```



4!

Fehler I

Listing 22: src/error/FehlerEins.java

```
1 public class Fehler1 {  
2     public static void main(String [] args) {  
3         System.out.println(" Hello World")  
4     }  
5 }
```

```
[java src]$ javac src/error/FehlerEins.java
```

```
-----  
1. ERROR in src/error/FehlerEins.java (at line 3)  
System.out.println("Hello World")  
                        ^
```

```
Syntax error, insert ";" to complete BlockStatements
```

```
-----  
1 problem (1 error)
```

Fehler II

Listing 23: src/error/FehlerZwei.java

```
1 public class FehlerZwei {  
2     public static void main(String [] argv) {  
3         int solution = 3 + 5;  
4         System.out.println("3 + 5 =" + soluton);  
5     }  
6 }
```

4!

Fehler II

Listing 24: src/error/FehlerZwei.java

```
1 public class FehlerZwei {
2     public static void main(String [] argv) {
3         int solution = 3 + 5;
4         System.out.println("3 + 5 =" + soluton);
5     }
6 }
```

```
[java src]$ javac src/error/FehlerZwei.java
```

```
-----
1. ERROR in src/error/FehlerZwei.java (at line 4)
System.out.println("3 + 5 =" + soluton);
                        ^^^^^^^^^
```

```
soluton cannot be resolved
```

```
-----
1 problem (1 error)
```

Listing 25: src/error/FehlerDrei.java

```
1  int dayOfWeek = 1;
2  switch (dayOfWeek) {
3      case 1:
4          String dayName = "Monday";
5          break;
6          // ...
7  }
8  System.out.println("Weekday = " + dayName);
```

Fehler III: Compilermeldung

```
[java src]$ javac src/error/FehlerDrei.java
```

```
-----  
1. ERROR in src/error/FehlerDrei.java (at line 10)
```

```
System.out.println("Weekday = " + dayName);
```

```
~~~~~
```

```
dayName cannot be resolved
```

```
-----
```

Listing 26: src/error/FehlerDrei.java

```
1  switch (dayOfWeek) {  
2      case 1:  
3          String dayName = "Monday";  
4          break;  
5          // ...  
6      }  
7      System.out.println("Weekday = " + dayName);
```

Listing 27: src/error/FehlerDrei.java

```
1  switch (dayOfWeek) {  
2      case 1:  
3          String dayName = "Monday";  
4          break;  
5          // ...  
6  }  
7  System.out.println("Weekday = " + dayName);
```

- ▶ Variablen gelten nur innerhalb ihres Blockes { }
- ▶ dayName gilt nur innerhalb von switch
- ▶ Lösung: Deklaration von dayName außerhalb

Listing 28: src/FehlerDreiKorrekt.java

```
1 int dayOfWeek = 1;
2 String dayName = "";
3 switch (dayOfWeek) {
4     case 1:
5         dayName = "Monday";
6         break;
7         // ...
8 }
9 System.out.println("Weekday = " + dayName);
```

Listing 29: src/FehlerDreiKorrekt.java

```
1  int dayOfWeek = 1;
2  String dayName = "";
3  switch (dayOfWeek) {
4      case 1:
5          dayName = "Monday";
6          break;
7          // ...
8  }
9  System.out.println("Weekday = " + dayName);
```

Ausgabe

```
[java src]$ java FehlerDreiSolution
Weekday = Monday
```

Listing 30: src/FehlerVier.java

```
1 System.out.println( 1 / 0 );
```

A large, light gray graphic in the background features a circle containing the number '4' followed by an exclamation mark '!', representing a runtime error (Exception).

Listing 31: src/FehlerVier.java

```
1 System.out.println( 1 / 0 );
```

Kein Compilerfehler

```
[java src]$ javac FehlerVier.java
```

```
[java src]$
```

4!

Listing 32: src/FehlerVier.java

```
1 System.out.println( 1 / 0 );
```

Kein Compilerfehler

```
[java src]$ javac FehlerVier.java  
[java src]$
```

Aber: Laufzeitfehler

```
[java src]$ java FehlerVier  
Exception in thread "main" java.lang.  
ArithmeticException: / by zero  
at FehlerVier.main(FehlerVier.java:3)
```

Inhaltsverzeichnis

- 1 Organisatorisches
- 2 Schreiben, Kompilieren, Ausführen
- 3 Variablen
- 4 Operatoren
- 5 Fallunterscheidungen
- 6 Lesbarkeit
- 7 Typische Fehler
- 8 Zusammenfassung**



4!

Was haben wir gelernt?

- ▶ Javaprogramme sind eine Folge von Befehlen, die hintereinander ausgeführt werden, in genau der Reihenfolge, wie sie da stehen.
- ▶ Klassen werden kompiliert, der entstehende Bytecode wird von der Java Virtual Machine ausgeführt.
- ▶ Variablen können an einer bestimmten Stelle während des Ausführens einen Wert annehmen, der sich an anderer Stelle ändern kann.
- ▶ Verzweigungen beeinflussen, welche Befehle ausgeführt werden.
- ▶ Kommentare erleichtern das Leben.
- ▶ Es gibt Compilerfehler und Laufzeitfehler, die Java automatisch meldet.

Und jetzt?

- ▶ Feedbackzettel ausfüllen und vorn abgeben

Üben!

- ▶ Aufgaben: <http://wiki.freitagrunde.org/Javakurs/Übungsaufgaben>
- ▶ TEL 106 / 206 (Hochhaus am Ernst-Reuter-Platz)
- ▶ Hilfe ist vorhanden
- ▶ Mittagspause von 13:15 Uhr bis 14:15 Uhr
- ▶ 14:15 Uhr - Vortrag zum Thema *Schleifen und Arrays*

4!