Java-Kurs 2011: Hello World

Freitagsrunde: Wer sind wir?

- Studentische Initiative voller Studenten
- Gesamte Fak IV: ET, TI, Info
- Organisiert: Kurse, Kickerturniere, Gremienarbeit
  Einführungswochen, Klausurensammlung,…
- Treffen: Freitags 13 Uhr im FR5046

Javakurs 2011

	Donnerstag	Freitag	Montag	Dienstag
10:00 – 11:15	VL: Hello World	VL: Methoden	VL: Von der Aufgabe zum Code	VL: Vererbung
11:30 – 13:15	Übung	Übung	Übung	Übung
13:15 – 14:15	Mittagspause	Mittagspause	Mittagspause	Mittagspause
14:15 – 15:15	VL: Schleifen und Arrays	Übung	VL: Objekte und Kapselung	Übung
15:30 – 17:30	Übung	Übung	Übung	Übung

→ Vorlesungen: MA004 (17. + 18.3.) – MA005 (21. + 22.3.)
→ Übungen: TEL106/206

Motivation & Ziele des Kurses

- Für euch:
  - Java - Programme schreiben
  - Grundlagen von Java erlernen
  - Fehler im Code finden und lösen
- Für uns:
  - Üben von Vorträgen
  - Wissen vermitteln
  - Feedback zum Vortragsstil
    - Es gibt anonyme Feedback-Zettel für euch zum Ausfüllen
      - Persönliches Feedback für den Vortragenden

- Zusammen: Viel Spaß haben ☺

Es gibt keine Scheine oder Leistungsbescheinigungen!

Agenda

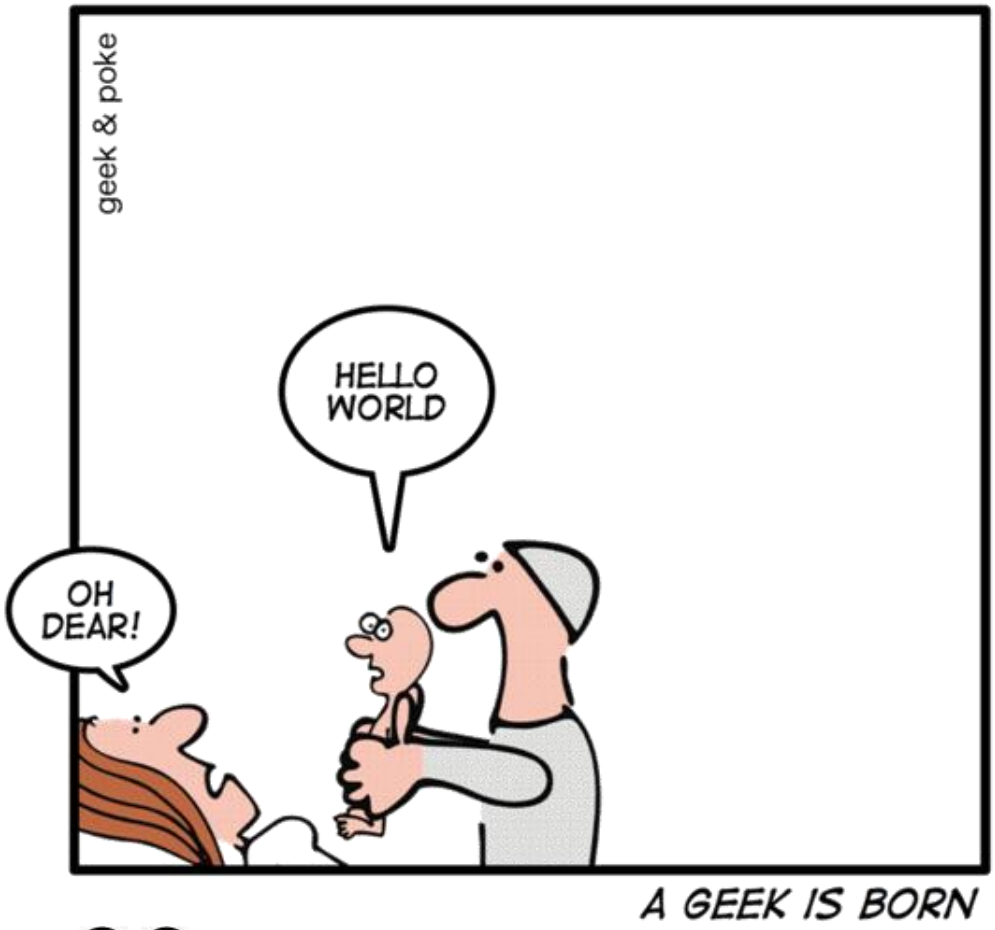
- Hello World
  - Arbeitsumgebung
  - Kompilieren und Ausführen
- Variablen und einfache Typen
- Operatoren
- Fallunterscheidungen
- Kommentare
- Fehlerbehebungen
- Zusammenfassung

Vortragender

- Student im 5. Semester
- Informatik
- Bachelor

Agenda

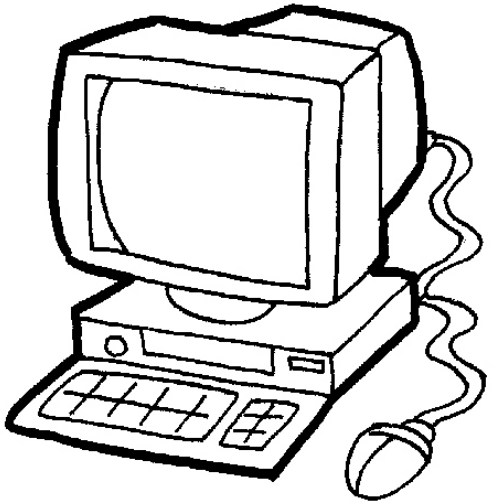
- **Hello World**
  - **Arbeitsumgebung**
  - **Kompilieren und Ausführen**
- Variablen und einfache Typen
- Operatoren
- Fallunterscheidungen
- Kommentare
- Fehlerbehebungen
- Zusammenfassung



by Oliver Widder

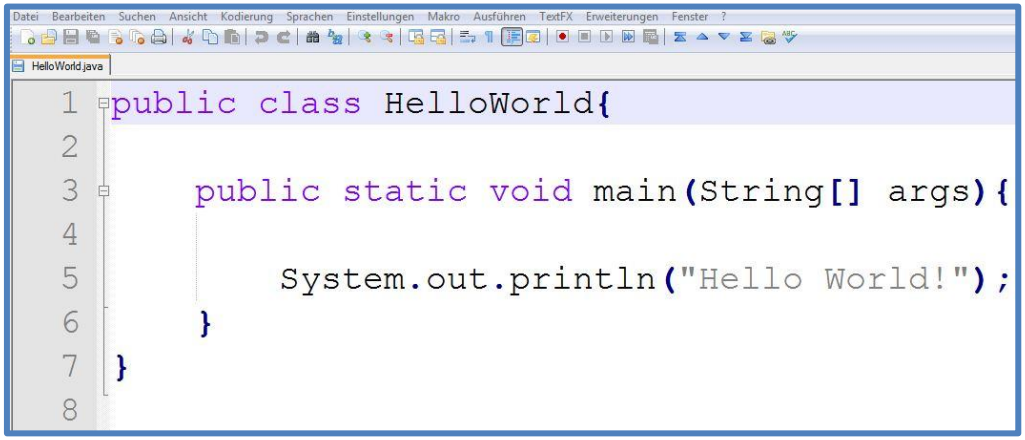
Arbeitsumgebung

- Einloggen
  - Im TEL an den Rechnern unter einem Unix-System
    (Kein Windows, weil → kein Java installiert)



Arbeitsumgebung

- Einloggen
- Editor öffnen
  - zB: **Gedit** (Gnome)
    oder **kate** (KDE)
    oder **notepad++** (win)
    oder **TextWrangler** (mac)



- und Programm schreiben

Arbeitsumgebung

Datei: HelloWorld.java

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

→ Beim Starten des Programms wird die „main“ – Methode ausgeführt.

Arbeitsumgebung

- Einloggen
- Editor öffnen
- Programm schreiben
- Eine Shell öffnen:

- → Konsole, Terminal, Eingabeaufforderung, Kommandozeile

Kompilieren

Der Compiler übersetzt
den Quellcode in ein
ausführbares Programm

javac ist der **Java C**ompiler

```
> javac HelloWorld.java
```

Kompilieren eines Java – Programms.

Byte-Code: Ausgabe des Java-Compilers

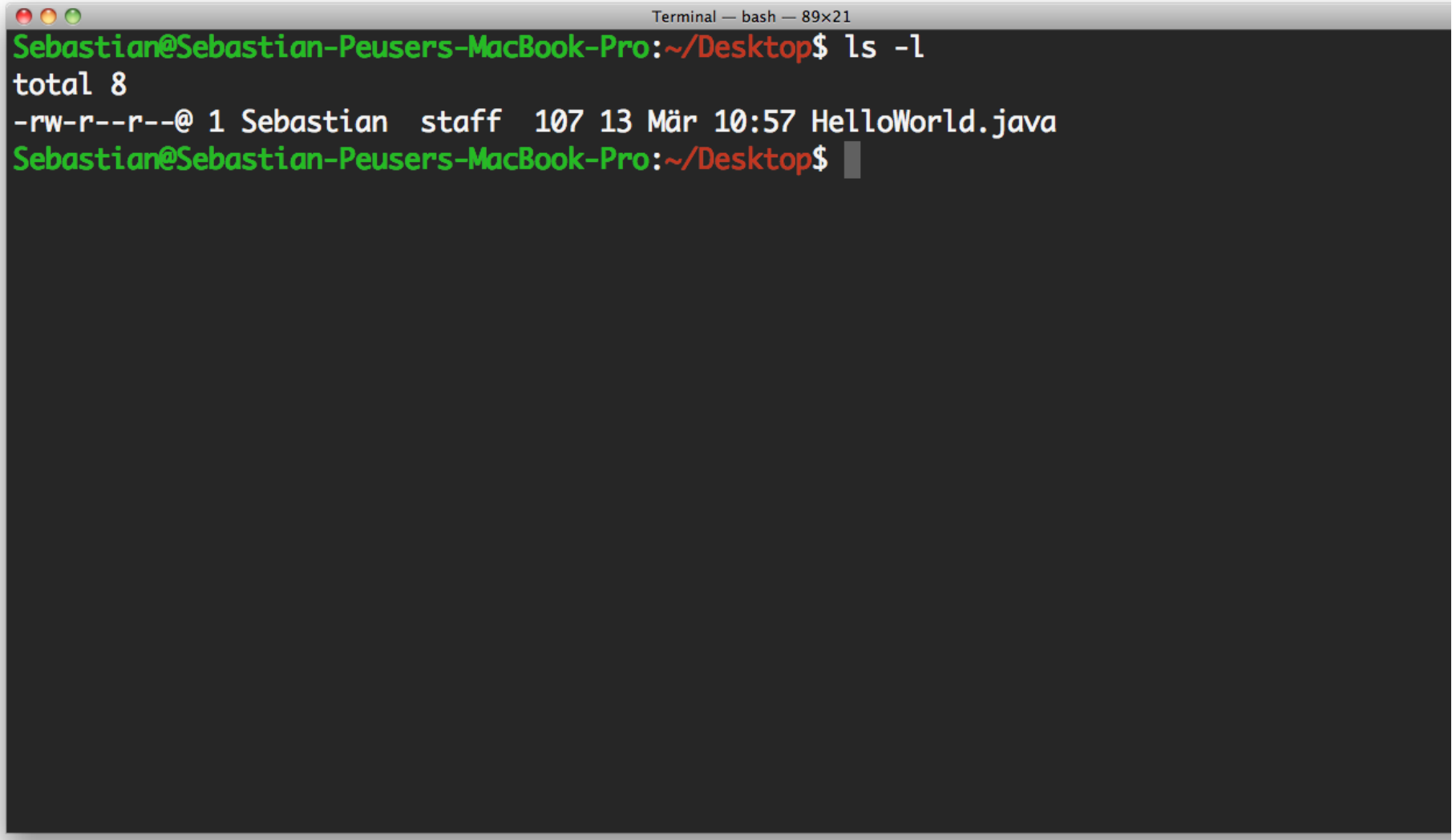
```
    Verzeichnis: C:\Users\Sebastian\Desktop\Javakurs2011


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---          11.03.2011     11:54        426 HelloWorld.class
-a---          11.03.2011     10:44        119 HelloWorld.java
```

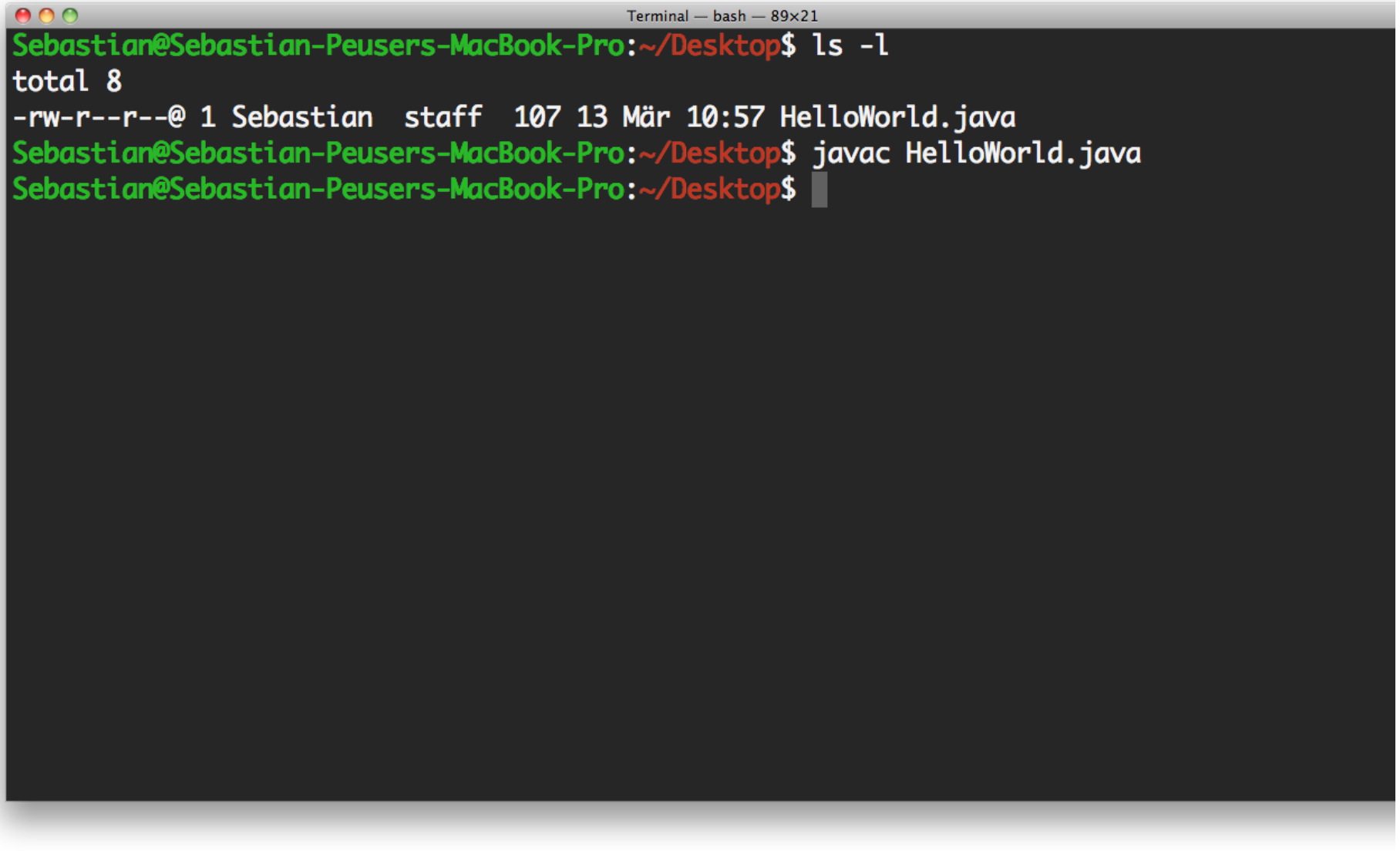
→ Kompilieren erzeugt .class-Datei in der der Bytecode liegt
→ Bytecode kann von der „**J**ava **V**irtual **M**achine“ ausgeführt werden
→ Bytecode ist maschinen**un**abhängig

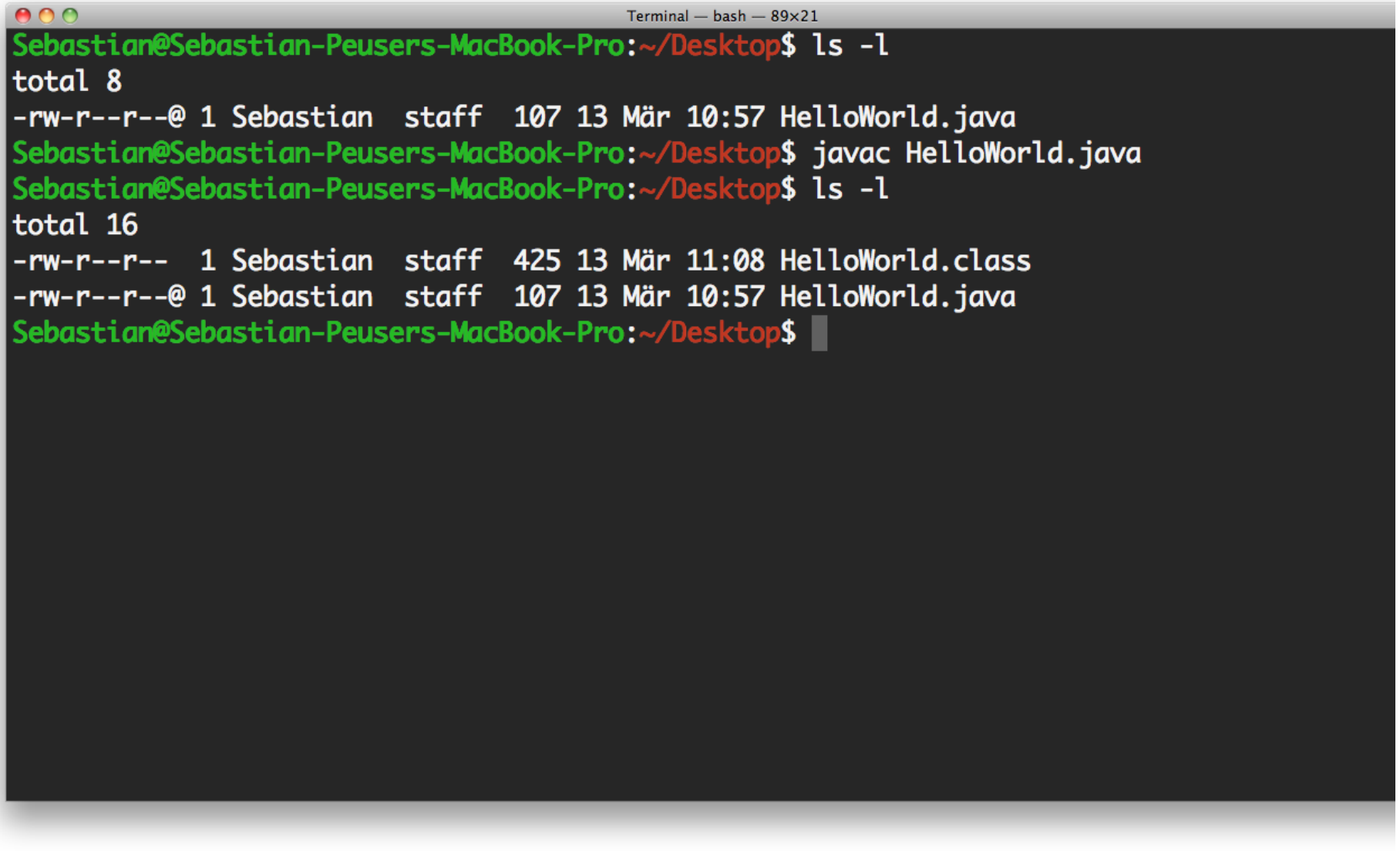
Ausführen

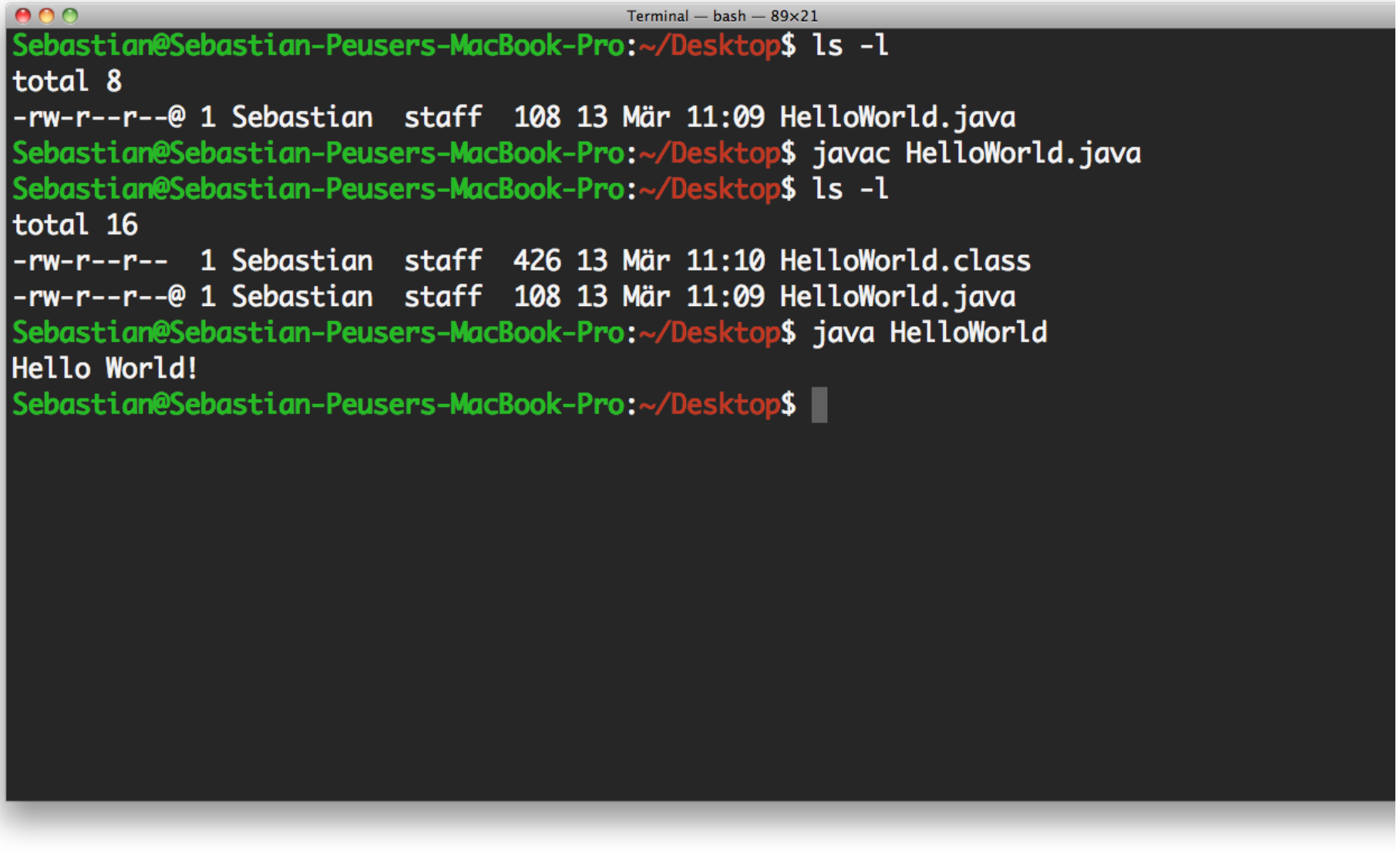
```
PS C:\Users\Sebastian\Desktop\Javakurs2011> java HelloWorld
Hello World!
```

→ „java“ ist die **Java Virtual Machine** (JVM)
→ als Parameter wird der Klassenname übergeben


Terminal — bash — 89×21
Sebastian@Sebastian-Peusers-MacBook-Pro:~/Desktop$ ls -l
total 8
-rw-r--r--@ 1 Sebastian staff 107 13 Mär 10:57 HelloWorld.java
Sebastian@Sebastian-Peusers-MacBook-Pro:~/Desktop$


Terminal — bash — 89×21
Sebastian@Sebastian-Peusers-MacBook-Pro:~/Desktop$ ls -l
total 8
-rw-r--r--@ 1 Sebastian staff 107 13 Mär 10:57 HelloWorld.java
Sebastian@Sebastian-Peusers-MacBook-Pro:~/Desktop$ javac HelloWorld.java
Sebastian@Sebastian-Peusers-MacBook-Pro:~/Desktop$


Terminal — bash — 89×21
Sebastian@Sebastian-Peusers-MacBook-Pro:~/Desktop$ ls -l
total 8
-rw-r--r--@ 1 Sebastian staff 107 13 Mär 10:57 HelloWorld.java
Sebastian@Sebastian-Peusers-MacBook-Pro:~/Desktop$ javac HelloWorld.java
Sebastian@Sebastian-Peusers-MacBook-Pro:~/Desktop$ ls -l
total 16
-rw-r--r-- 1 Sebastian staff 425 13 Mär 11:08 HelloWorld.class
-rw-r--r--@ 1 Sebastian staff 107 13 Mär 10:57 HelloWorld.java
Sebastian@Sebastian-Peusers-MacBook-Pro:~/Desktop$


Terminal — bash — 89×21
Sebastian@Sebastian-Peusers-MacBook-Pro:~/Desktop$ ls -l
total 8
-rw-r--r--@ 1 Sebastian staff 108 13 Mär 11:09 HelloWorld.java
Sebastian@Sebastian-Peusers-MacBook-Pro:~/Desktop$ javac HelloWorld.java
Sebastian@Sebastian-Peusers-MacBook-Pro:~/Desktop$ ls -l
total 16
-rw-r--r-- 1 Sebastian staff 426 13 Mär 11:10 HelloWorld.class
-rw-r--r--@ 1 Sebastian staff 108 13 Mär 11:09 HelloWorld.java
Sebastian@Sebastian-Peusers-MacBook-Pro:~/Desktop$ java HelloWorld
Hello World!
Sebastian@Sebastian-Peusers-MacBook-Pro:~/Desktop$

Sequenzielles Abarbeiten von Befehlen

Datei: Enumerate.java

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

Sequenzielles Abarbeiten von Befehlen

Datei: Enumerate.java

```java
public class Enumerate{
    public static void main(String[] args){
        System.out.println("First!");
        System.out.println("second!");
        System.out.println("third!");
    }
}
```

- Befehle werden der Reihe nach abgearbeitet
- Klassenname muss mit Dateiname übereinstimmen

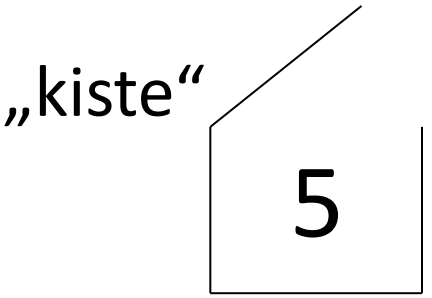
```
PS C:\Users\Sebastian\Desktop\Javakurs> java Enumerate
First!
second!
third!
```

Agenda

- Hello World
  - Arbeitsumgebung
  - Kompilieren und Ausführen
- **Variablen und einfache Typen**
- Operatoren
- Fallunterscheidungen
- Kommentare
- Fehlerbehebungen
- Zusammenfassung

Was ist eine Variable?

- Eine Variable ist eine Art Behälter/Kiste in der Werte gespeichert werden können.
- Der Name der Kiste ist der Variablenname



➢ Die Variable „kiste“ hat den Wert 5

Datentyp – int

Datei: Variable.java

```
1  public class Variable{
2
3      public static void main(String[] args){
4
5          //Deklaration einer Variablen
6          int number;
7          //Initialisierung einer Variablen
8          number = 23;
9
10         System.out.println(number);
11     }
12 }
```

→ **int** steht für (Englisch: Integer) – eine ganze Zahl
→ „=“ weist der Variablen „number“ den Wert 23 zu

Datentyp – int

```
PS C:\Users\Sebastian\Desktop\Javakurs> javac Variable.java
PS C:\Users\Sebastian\Desktop\Javakurs> java Variable
23
```

- Kompilieren und Ausführen
- Wert der Variablen wird direkt auf die Konsole geschrieben

Datentyp – String

Datei: VariableI.java

```
1  public class VariableI{
2
3      public static void main(String[] args){
4
5          int number = -10;
6          number = number + 23;
7
8          String message = "My favorite number is: ";
9
10         System.out.println(message + number);
11     }
12 }
```

Datentyp – String

Datei: VariableI.java

```java
public class VariableI{

    public static void main(String[] args){

        int number = -10;
        number = number + 23;

        String message = "My favorite number is: ";

        System.out.println(message + number);
    }
}
```

```
PS C:\Users\Sebastian\Desktop\Javakurs> javac VariableI.java
PS C:\Users\Sebastian\Desktop\Javakurs> java VariableI
My favorite number is: 13
```

Datentyp – double

Datei: VariableII.java

```
1  public class VariableII{
2
3      public static void main(String[] args){
4
5          double temperature = 23.5;
6          String message = "The temperature is: ";
7          System.out.println(message + temperature);
8      }
9  }
```

- **double** ist eine Fließkommazahl

Datentyp – double

Datei: VariableII.java

```java
public class VariableII{

    public static void main(String[] args){

        double temperature = 23.5;
        String message = "The temperature is: ";
        System.out.println(message + temperature);
    }
}
```

```
PS C:\Users\Sebastian\Desktop\Javakurs> javac VariableII.java
PS C:\Users\Sebastian\Desktop\Javakurs> java VariableII
The temperature is: 23.5
```

- **double** ist eine Fließkommazahl

Datentyp – boolean

Datei: VariableIII.java

```
1  public class VariableIII{
2
3      public static void main(String[] args){
4
5          boolean amISmart = true;
6          boolean amIAJavaHacker = false;
7
8          boolean answer = amISmart && amIAJavaHacker;
9          String message = "Am I a smart Java hacker? ";
10         System.out.println(message + answer);
11     }
12 }
```

Datentyp – boolean

Datei: VariableIII.java

```java
public class VariableIII{

    public static void main(String[] args){

        boolean amISmart = true;
        boolean amIAJavaHacker = false;

        boolean answer = amISmart && amIAJavaHacker;
        String message = "Am I a smart Java hacker? ";
        System.out.println(message + answer);
    }
}
```

```
PS C:\Users\Sebastian\Desktop\Javakurs> javac VariableIII.java
PS C:\Users\Sebastian\Desktop\Javakurs> java VariableIII
Am I a smart Java hacker? false
```

Datentypen im Überblick

Typen	Wertebereiche
`boolean answer = true;`	{true, false}
`int number = 23;`	{-2147483648 … 2147483647}
`double temperature = 23.50;`	{- 4,9 × 10^{-324} … + 1,7977 × 10^{+308}}
`String message = "Hello World";`	{„“ … endlich}

- Dies sind nur ein paar Datentypen, dafür aber die wohl wichtigsten

Konventionen

- Variablen werden im camelCase geschrieben, da keine Leerzeichen erlaubt sind
- dabei ist der erste Buchstabe klein

- Beispiel: **amIAJavaHacker**

- Es sollten kurze und aussagekräftige Namen verwendet werden!

Agenda

- Hello World
  - Arbeitsumgebung
  - Kompilieren und Ausführen
- Variablen und einfache Typen
- **Operatoren**
- Fallunterscheidungen
- Kommentare
- Fehlerbehebungen
- Zusammenfassung

Operatoren

Datei: Example.java

```
1  public class Example{
2
3      public static void main(String[] args){
4
5          int x,y;
6          x = 5;
7          y = 3;
8
9          int mult = x * y;
10         double average = (x + y)/2.0;
11
12         System.out.println(mult + " ; " + average);
13     }
14 }
15
```

Operatoren

Datei: Example.java

```java
public class Example{

    public static void main(String[] args){

        int x,y;
        x = 5;
        y = 3;

        int mult = x * y;
        double average = (x + y)/2.0;

        System.out.println(mult + " ; " + average);
    }
}
```

```
PS C:\Users\Sebastian\Desktop\Javakurs> javac Example.java
PS C:\Users\Sebastian\Desktop\Javakurs> java Example
15 ; 4.0
```

Operatoren – Überblick

Logische Operatoren

- `&&` Und `true && true ≙ true`
- `||` Oder `false || false ≙ false`
- `!` Negation `!true ≙ false`

Relationale Operatoren

- `<=` `3 <= 4 ≙ true`
- `>=` `3 >= 3 ≙ true`
- `<` `3 < 4 ≙ true`
- `>` `3 > 3 ≙ false`
- `==` `1 == 1 ≙ true`
- `!=` `1 != 2 ≙ true`

Operatoren – Überblick

Logische Operatoren

- `&&` Und — `true && true ≙ true`
- `||` Oder — `false || false ≙ false`
- `!` Negation — `!true ≙ false`

Relationale Operatoren

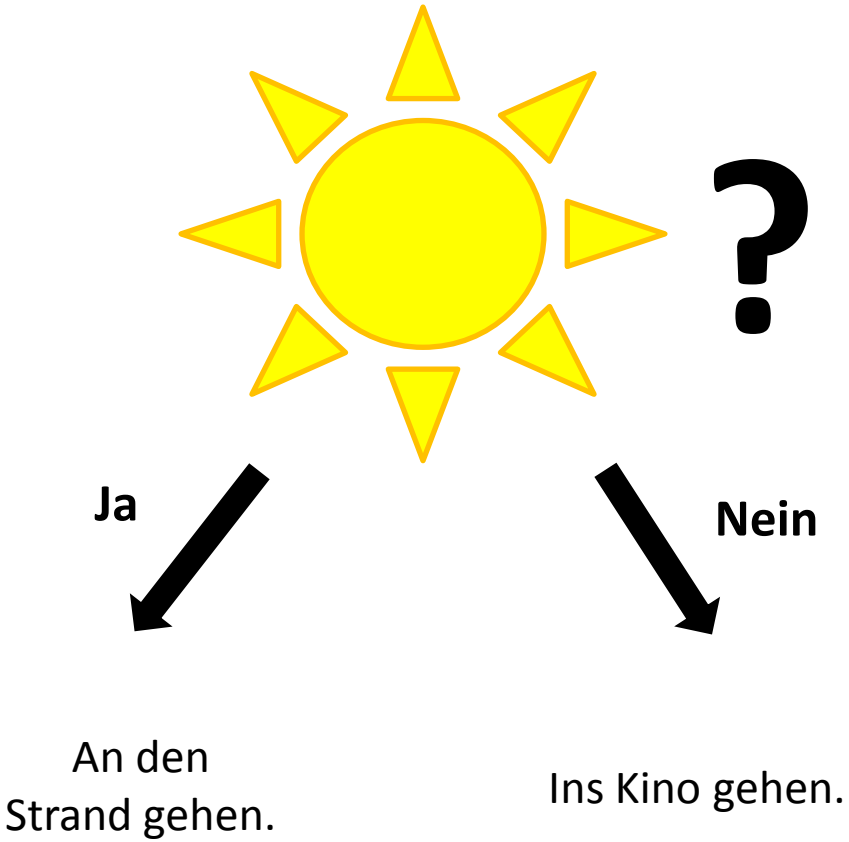
- `<=` — `3 <= 4 ≙ true`
- `>=` — `3 >= 3 ≙ true`
- `<` — `3 < 4 ≙ true`
- `>` — `3 > 3 ≙ false`
- `==` — `1 == 1 ≙ true`
- `!=` — `1 != 2 ≙ true`

Arithmetische Operatoren

- \+ Addition
- \- Subtraktion
- * Multiplikation
- / Division
- % Modulo

Agenda

- Hello World
  - Arbeitsumgebung
  - Kompilieren und Ausführen
- Variablen und einfache Typen
- Operatoren
- **Fallunterscheidungen**
- Kommentare
- Fehlerbehebungen
- Zusammenfassung



If-Then-Else: Einfaches Beispiel

Datei: IfSimple.java

```
5           boolean sunIsShining = true;
6           if(sunIsShining == true){
7               System.out.println("Go to the beach");
8           }
9           if(sunIsShining == false){
10              System.out.println("Go to the cinema");
11          }
```

If-Then-Else: Einfaches Beispiel

Datei: IfSimple.java

```
5           boolean sunIsShining = true;
6           if(sunIsShining == true){
7               System.out.println("Go to the beach");
8           }
9           if(sunIsShining == false){
10              System.out.println("Go to the cinema");
11          }
```

Variante 2:

```
12          boolean sunIsShiningTwo = false;
13          if(sunIsShiningTwo == true){
14              System.out.println("Go to the beach");
15          }else{
16              System.out.println("Go to the cinema");
17          }
```

If-Then-Else: Einfaches Beispiel

Datei: IfSimple.java

```
5           boolean sunIsShining = true;
6           if(sunIsShining == true){
7               System.out.println("Go to the beach");
8           }
9           if(sunIsShining == false){
10              System.out.println("Go to the cinema");
11          }
```

Variante 2:

```
12          boolean sunIsShiningTwo = false;
13          if(sunIsShiningTwo == true){
14              System.out.println("Go to the beach");
15          }else{
16              System.out.println("Go to the cinema");
17          }
```

```
PS C:\Users\Sebastian\Desktop\Javakurs> javac IfSimple.java
PS C:\Users\Sebastian\Desktop\Javakurs> java IfSimple
Go to the beach
Go to the cinema
```

Bedingung mit relationalen Operatoren

Datei: IfSimpleI.java

```
5       int waterTemperature = 25;
6       if(waterTemperature >= 22){
7           System.out.println("Go to the beach");
8       }else{
9           System.out.println("Chase after pigeons");
10      }
```

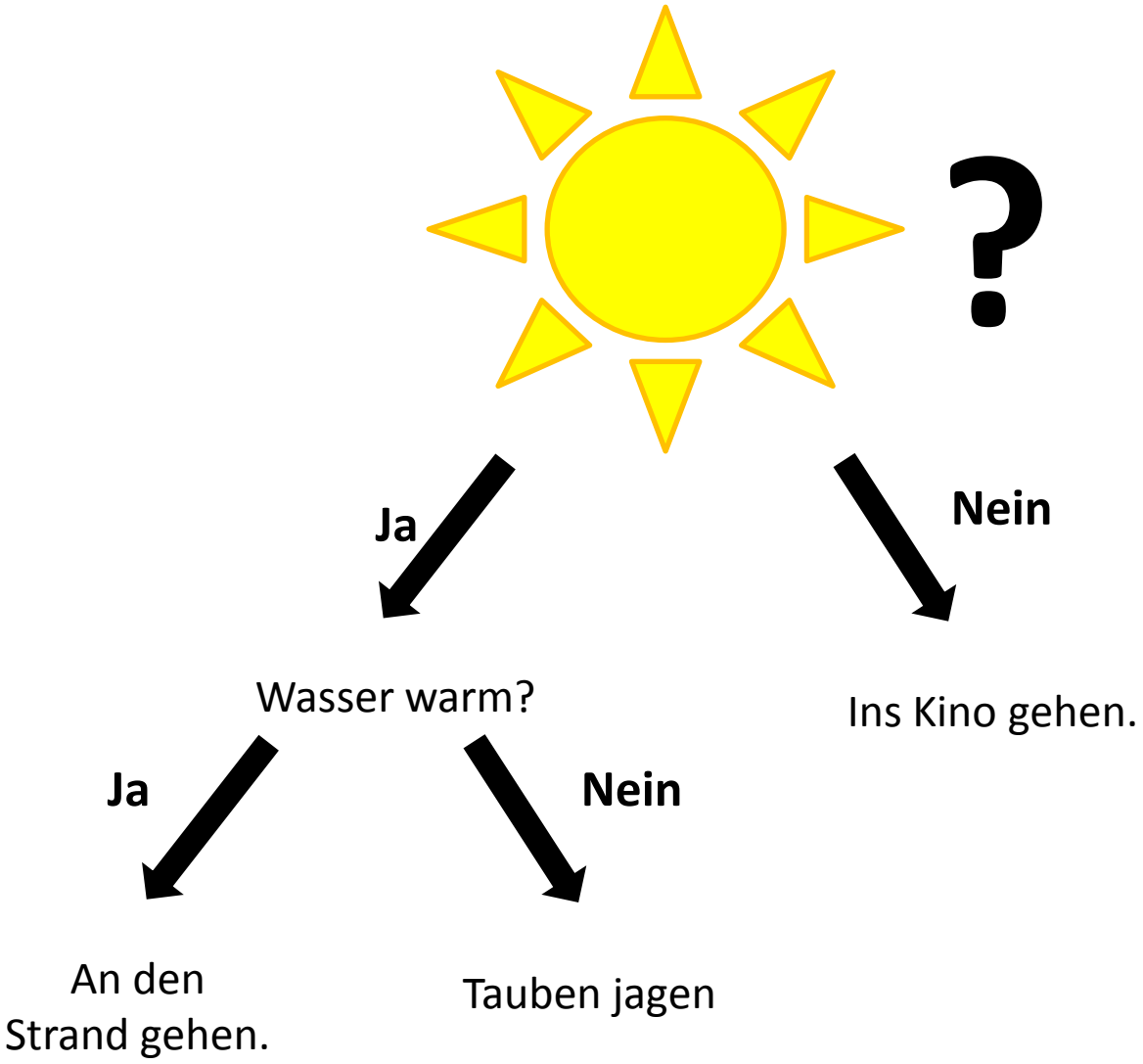
Bedingung mit logischen Operatoren

Datei: IfSimpleI.java

```
5       int waterTemperature = 25;
6       if(waterTemperature >= 22){
7           System.out.println("Go to the beach");
8       }else{
9           System.out.println("Chase after pigeons");
10      }
```

```
PS C:\Users\Sebastian\Desktop\Javakurs> javac IfSimpleI.java
PS C:\Users\Sebastian\Desktop\Javakurs> java IfSimpleI
Go to the beach
```

Beispiel verfeinert



Kombinierte Bedingung (else if)

Datei: IfSimpleII.java

```
5           boolean sunIsShining = true;
6           int waterTemperature = 21;
7
8           if(sunIsShining && waterTemperature >= 22){
9               System.out.println("Go to the beach");
10          }else if(sunIsShining){
11              System.out.println("Chase after pigeons");
12          }else{
13              System.out.println("Go to the cinema");
14          }
```

Kombinierte Bedingung (else if)

Datei: IfSimpleII.java

```
5           boolean sunIsShining = true;
6           int waterTemperature = 21;
7
8           if(sunIsShining && waterTemperature >= 22){
9               System.out.println("Go to the beach");
10          }else if(sunIsShining){
11              System.out.println("Chase after pigeons");
12          }else{
13              System.out.println("Go to the cinema");
14          }
```

```
PS C:\Users\Sebastian\Desktop\Javakurs> javac IfSimpleII.java
PS C:\Users\Sebastian\Desktop\Javakurs> java IfSimpleII
Chase after pigeons
```

Switch-Case für ganzzahlige Bedingungen

Datei: SwitchCase.java

```
5         int year = 2010;
6         switch(year){
7             // ...
8             case 2009: System.out.println("Jahr: 2009");
9             break;
10            case 2010: System.out.println("Jahr: 2010");
11            break;
12            case 2011: System.out.println("Jahr: 2011");
13            break;
14            // ...
15            default: System.out.println("Jahr: Unbekannt");
16        }
```

Switch-Case für ganzzahlige Bedingungen

Datei: SwitchCase.java

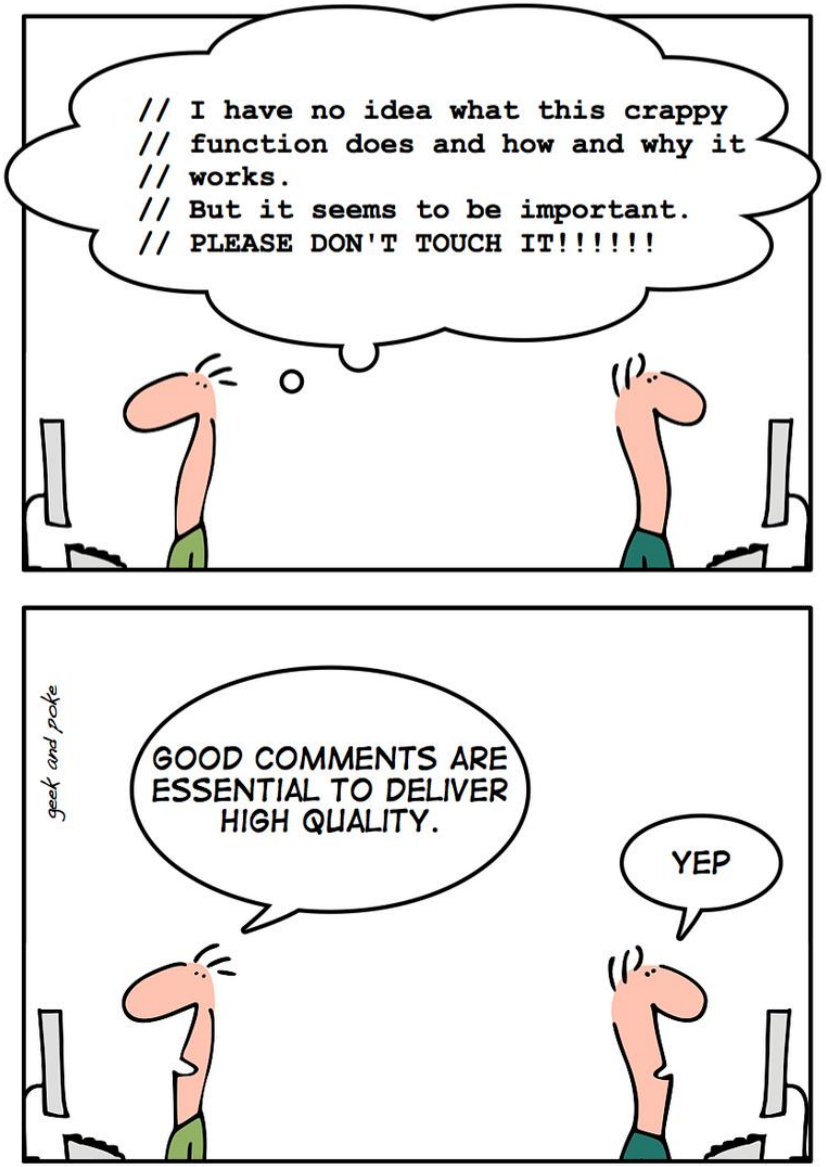
```
5         int year = 2010;
6         switch(year){
7             // ...
8             case 2009: System.out.println("Jahr: 2009");
9             break;
10            case 2010: System.out.println("Jahr: 2010");
11            break;
12            case 2011: System.out.println("Jahr: 2011");
13            break;
14            // ...
15            default: System.out.println("Jahr: Unbekannt");
16        }
```

```
PS C:\Users\Sebastian\Desktop\Javakurs> javac SwitchCase.java
PS C:\Users\Sebastian\Desktop\Javakurs> java SwitchCase
Jahr: 2010
```

- Zahl bei „switch“ bestimmt **ab** welchem „case“ der Code ausgeführt wird.
- „break“ unterbricht die weitere Ausführung und Programm springt aus dem switch-Block

Agenda

- Hello World
  - Arbeitsumgebung
  - Kompilieren und Ausführen
- Variablen und einfache Typen
- Operatoren
- Fallunterscheidungen
- **Kommentare**
- Fehlerbehebungen
- Zusammenfassung



by Oliver Widder

Kommentare helfen den Code zu verstehen

- Warum mache ich das Folgende?
- Was für Randbedingungen gibt es?
- Warum sollte ich es nicht anders lösen?

- Wichtig:
  - Kommentar vor dem eigentlichen Code
    - Hilft beim Denken
    - Hilft beim Nachvollziehen

Kommentare helfen den Code zu verstehen

- Warum mache ich das Folgende?
- Was für Randbedingungen gibt es?
- Warum sollte ich es nicht anders lösen?

- Wichtig:
  - Kommentar vor dem eigentlichen Code
    - Hilft beim Denken
    - Hilft beim Nachvollziehen

Erklärung:
// Einzeiliger Kommentar
/* Kommentar bis */

Code ohne Kommentare

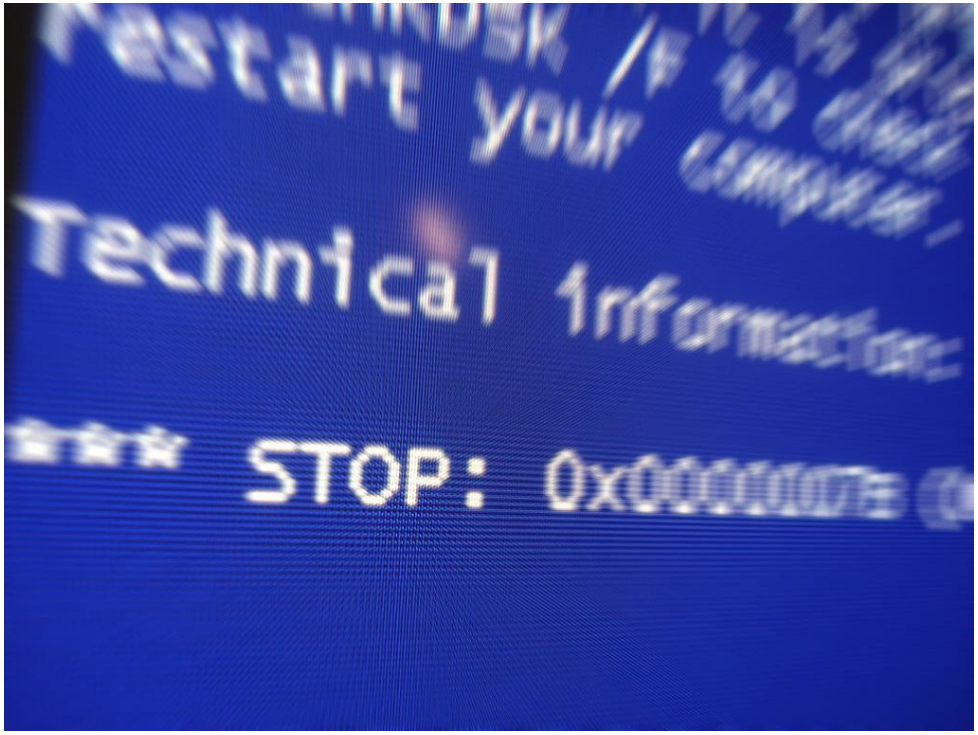
Datei: NoComments.java

```
1 public class NoComments{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");}}}
```

- Sehr schwer lesbar!

Agenda

- Hello World
  - Arbeitsumgebung
  - Kompilieren und Ausführen
- Variablen und einfache Typen
- Operatoren
- Fallunterscheidungen
- Kommentare
- **Fehlerbehebungen**
- Zusammenfassung



© by Justin Marty

Klassischer Fehler

Datei: no-compile/Error.java

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

Klassischer Fehler

Datei: no-compile/Error.java

```java
public class Error{

    public static void main(String[] args){

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

```
PS C:\Users\Sebastian\Desktop\Javakurs\no-compile> javac Error.java
Error.java:5: ';' expected
                System.out.println("Hello World")
                                                  ^
1 error
```

Klassischer Fehler II

Datei: no-compile/ErrorTwo.java

```
1 public class ErrorTwo{
2
3     public static void main(String[] args){
4
5         int solution = 3;
6         System.out.println("1 + 2 = " + soluton);
7     }
8 }
```

Klassischer Fehler II

Datei: no-compile/ErrorTwo.java

```java
public class ErrorTwo{

    public static void main(String[] args){

        int solution = 3;
        System.out.println("1 + 2 = " + soluton);
    }
}
```

```
PS C:\Users\Sebastian\Desktop\Javakurs\no-compile> javac ErrorTwo.java
ErrorTwo.java:6: cannot find symbol
symbol  : variable soluton
location: class ErrorTwo
                System.out.println("1 + 2 = " + soluton);
                                                   ^
1 error
```

Klassischer Fehler III

Datei: no-compile/ErrorThree.java

```
1  public class ErrorThree{
2
3      public static void main(String[] args){
4
5          boolean flaq = true;
6          if(flaq){
7              String message = "Hello World";
8          }
9          System.out.println(message);
10     }
11 }
```

Klassischer Fehler III

Datei: no-compile/ErrorThree.java

```java
public class ErrorThree{

    public static void main(String[] args){

        boolean flaq = true;
        if(flaq){
            String message = "Hello World";
        }
        System.out.println(message);
    }
}
```

```
PS C:\Users\Sebastian\Desktop\Javakurs\no-compile> javac ErrorThree.java
ErrorThree.java:9: cannot find symbol
symbol  : variable message
location: class ErrorThree
                System.out.println(message);
                                   ^
1 error
```

Fehler III: Fehlerquelle

```
1  public class ErrorThree{
2
3      public static void main(String[] args){
4
5          boolean flaq = true;
6          if(flaq){
7              String message = "Hello World";
8          }
9          System.out.println(message);
10     }
11 }
```

Fehler III: Lösung

```
1  public class ErrorThree{
2
3      public static void main(String[] args){
4
5          boolean flag = true;
6          String message = "";
7          if(flag){
8              message = "Hello World";
9          }
10         System.out.println(message);
11     }
12 }
```

Fehler IV: Laufzeitfehler werden vom Compiler nicht erkannt

Datei: no-compile/Laufzeitfehler.java

```
1 public class Laufzeitfehler{
2     public static void main(String[] args){
3         System.out.println(1/0);
4     }
5 }
```

Fehler IV: Laufzeitfehler werden vom Compiler nicht erkannt

Datei: no-compile/Laufzeitfehler.java

```java
public class Laufzeitfehler{
    public static void main(String[] args){
        System.out.println(1/0);
    }
}
```

```
PS C:\Users\Sebastian\Desktop\Javakurs\no-compile> javac Laufzeitfehler.java
PS C:\Users\Sebastian\Desktop\Javakurs\no-compile>
```

Fehler IV: Laufzeitfehler werden vom Compiler nicht erkannt

Datei: no-compile/Laufzeitfehler.java

```java
public class Laufzeitfehler{
    public static void main(String[] args){
        System.out.println(1/0);
    }
}
```

```
PS C:\Users\Sebastian\Desktop\Javakurs\no-compile> javac Laufzeitfehler.java
PS C:\Users\Sebastian\Desktop\Javakurs\no-compile>
```

Aber: Es tritt ein Laufzeitfehler auf!

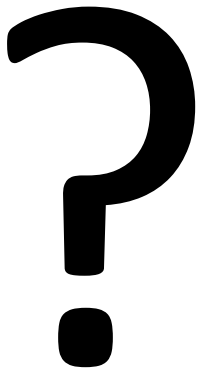
```
PS C:\Users\Sebastian\Desktop\Javakurs\no-compile> java Laufzeitfehler
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Laufzeitfehler.main(Laufzeitfehler.java:3)
```

Agenda

- Hello World
  - Arbeitsumgebung
  - Kompilieren und Ausführen
- Variablen und einfache Typen
- Operatoren
- Fallunterscheidungen
- Kommentare
- Fehlerbehebungen
- **Zusammenfassung**

- Erzeugen und Kompilieren eines Javaprogramms
- Einfache Konstrukte
  - Variablen
  - Bedingungen
  - Kommentare
- Kompilerfehler und deren Lösung

Fragen?



Zu den Übungen

- TEL am Ernst-Reuter-Platz
- Räume 106/206

➢ Den Tutoren folgen
  ➢ bzw. immer der Menschenmenge hinterher ☺

➢ Um 13:15 Uhr: 2. Vortrag hier! (MA004)