
Hello World


Milan


Thaddäus


4!
Freitagsrunde

Inhalte dieser Veranstaltung

- Grundlagen von Java lernen
  - LE1: Syntax, Comparatoren && Operatoren
  - LE2: Iteration; Arrays
  - LE3: Debuggen
  - LE4: Guter Code, Schlechter Code
  - LE5: Objekt-Orientierung
  - LE6: Pakete, Kapselung und Polymorphie
- Spass am Programmieren haben ;-)

1

Was habt ihr davon?

- Javakenntnisse
- Starthilfe für MPGI2
- Programmiererfahrungen durch die Übungen
- hoffentlich auch ein wenig Spaß

2

Was haben wir davon?

- Spaß an der Freude
- wir sammeln Erfahrungen im Vorträge halten
- wir werden berühmt ;-)
- wir kommen ins Fernsehen...
- ... und bekommen auch ein wenig Geld

Übungen

©TUB

Tel 106 und **Tel 206**

Übungsaufgaben:

`https://freitagsrunde.org/Javakurs`

Tagesablauf

10:00 - 11:00 erster Vortrag
11:00 - 13:00 erste Übung
13:00 - 14:00 Pause (Tipp: Mensa)
14:00 - 15:00 zweiter Vortrag
15.00 - 17:00 zweite Übung

* Es wird genügend Zeit zum Raum wechseln geben.

H1058 - Vortragsraum

Tel106/206 - Übungsraum

Noch mehr Infos

- Es gibt in den Übungen Feedbackzettel
- Es gibt auch einen IRC-Channel **#freitagsrunde** im Freenode (irc.freenode.net)

fragen? fragen? fragen? fragen? fragen? fragen? fragen? fragen?
fragen? fragen? fragen? fragen? fragen? fragen? fragen? fragen?
fragen? fragen? fragen? fragen? lagen? fragen? fragen? fragen?
fragen? tragen? fragen? fragen? fragen? fragen? fragen? fragen?
fragen? fragen? fragen? fragen? fragen? fragen? fragen? fragen?
fragen? fragen? fragen? fragen? fragen? fragen? fragen? fragen?
fragen? fragen? fragen? fragen? fragen? fragen? fragen? fragen?
fragen? fragen? fragen? fragen? fragen? fragen? magen? fragen?
fragen? fragen? fragen? fragen? fragen? fragen? fragen? fragen?
fragen? fragen? fragen? fragen? fragen? fragen? fragen? fragen?
fragen? fragen? fragen? fragen! fragen? fragen? fragen? fragen?
fragen? fragen? fragen? fragen? fragen? fragen? fragen? fragen?
fragen? fragen? fragen? fragen? fragen? fragen? fragen? fragen?
sagen? fragen? fragen? fragen? fragen? fragen? fragen? fragen?
fragen? fragen? fragen? fragen? fragen? fragen? fragen? fragen?
fragen? fragen? fragen? fragen? fragen? fragen? fragen? ja?

Antworten...

- ... gibt es auf unserer Website:
  `https://freitagsrunde.org/Javakurs`
- oder bei einem Tutor eurer Wahl ;-)

Los geht's...

Dann geht's jetzt richtig los...

8

Notizen machen?

Ja...aber nur das wichtige!

Alle Vorträge gibt es als PDF und als Screencast unter:
`https://freitagsrunde.org/Javakurs`

Inhalte & Ziele

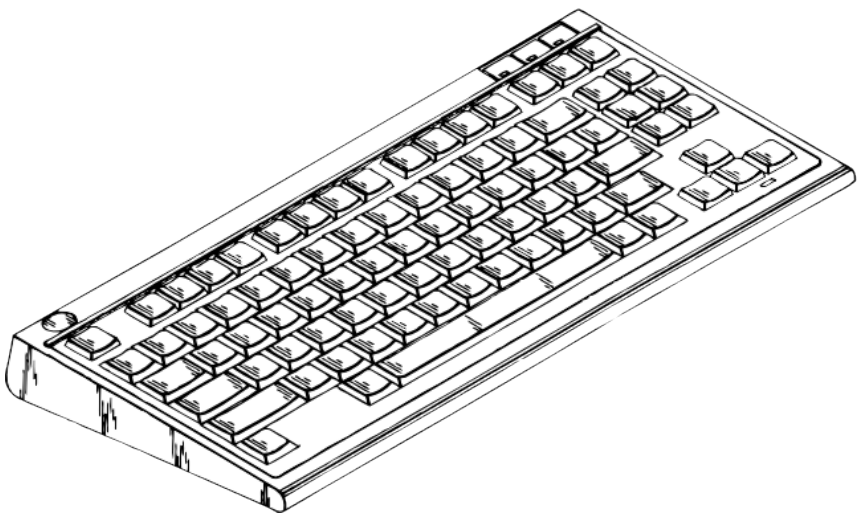
Inhalte dieser Vorlesung

- Hello World
- Kompilieren & Ausführen
- Variablen & grundlegende Typen
- Fallunterscheidungen (if)
- Kommentare
- Fehlermeldungen lesen


Arbeitsumgebung

Arbeitsumgebung

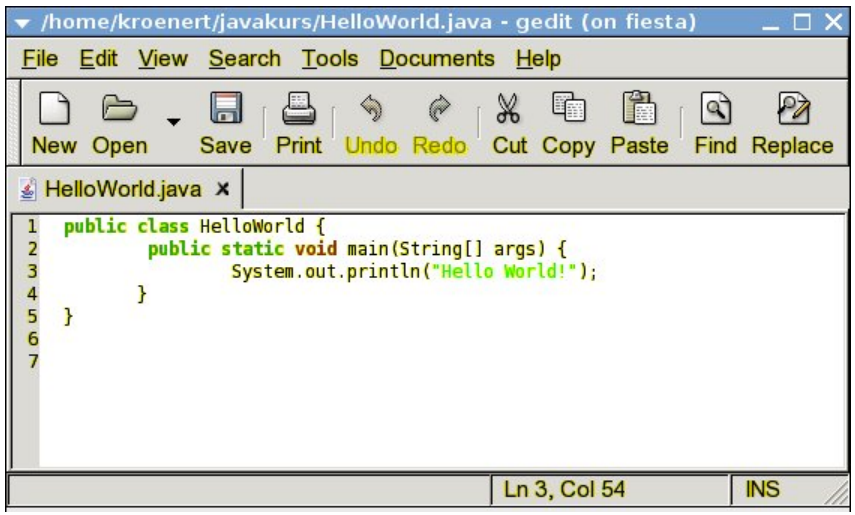
1. Einloggen



9

Arbeitsumgebung

1. Einloggen
2. einen Editor [1] öffnen



gedit

z.b. **Gedit** (Gnome) oder **kate** (KDE)

[1] Nein! Nicht Word!

Hello World

HelloWorld.java

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

Beim wird die “main” Methode ausgeführt

Abarbeiten von Befehlen

HelloSolarSystem.java

```
1  public class HelloSolarSystem {
2          public static void main(String[] args) {
3                  System.out.println("Hello Mercury!");
4                  System.out.println("Hello Venus!");
5                  System.out.println("Hello Earth!");
6                  System.out.println("Hello Mars!");
7                  System.out.println("Hello Jupiter!");
8                  System.out.println("Hello Saturn!");
9                  System.out.println("Hello Uranus!");
10                 System.out.println("Hello Neptune!");
11         }
12 }
```

Befehle werden der Reihe nach abgearbeitet

Klassennamen und Dateiname (ohne .java) müssen übereinstimmen

Kompilieren und Ausführen

Arbeitsumgebung

1. Einloggen
2. einen Editor öffnen
3. eine Shell [2] öffnen

10

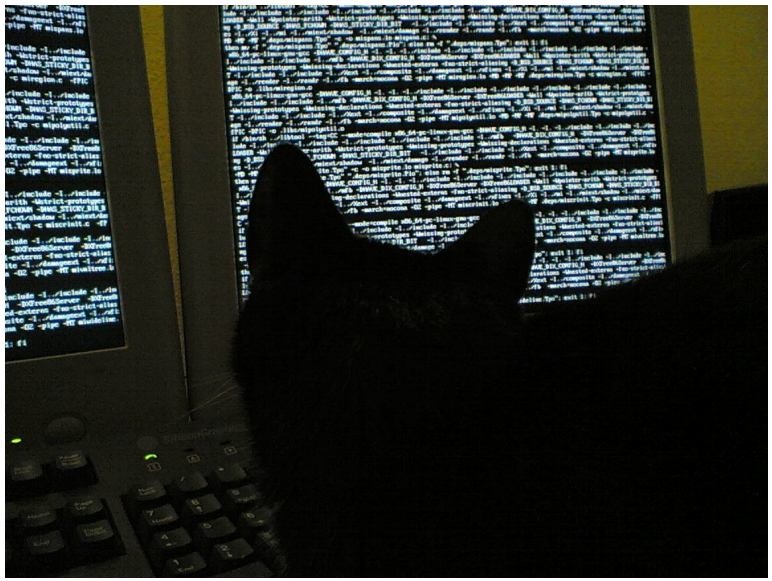
z.b. **gnome-terminal** (Gnome) oder **konsole** (KDE)

[2] Konsole, Terminal, (MS-DOS-)Eingabeaufforderung, Kommandozeile

Kompilieren

Der Compiler übersetzt den Quellcode in ein ausführbares Programm.

javac ist der **Java C**ompiler.



11

Shell

```
1 bueno kroenert: javac HelloWorld.java
2 bueno kroenert:
```

Bytecode

Shell

```
1   bueno kroenert: ls -l
2  -rw------- 1 kroenert 426 Mar 10 13:51 HelloWorld.class
3  -rw------- 1 kroenert 106 Mar 10 13:51 HelloWorld.java
```

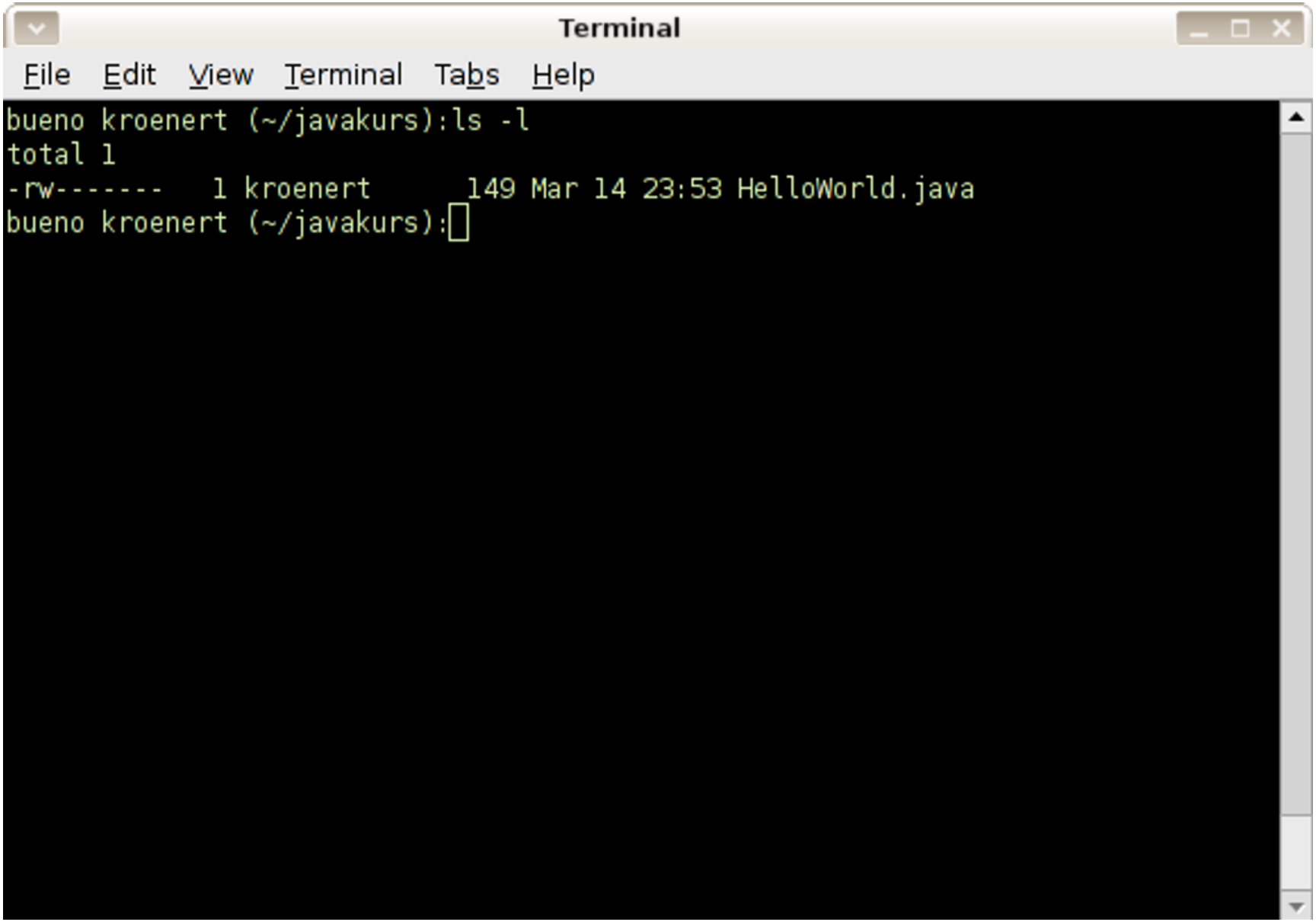
- Compilieren erzeugt .class Dateien, sog. Bytecode
- Bytecode kann mit einer **Java Virtual Machine** ausgeführt werden
- Bytecode ist maschinen**un**abhängig

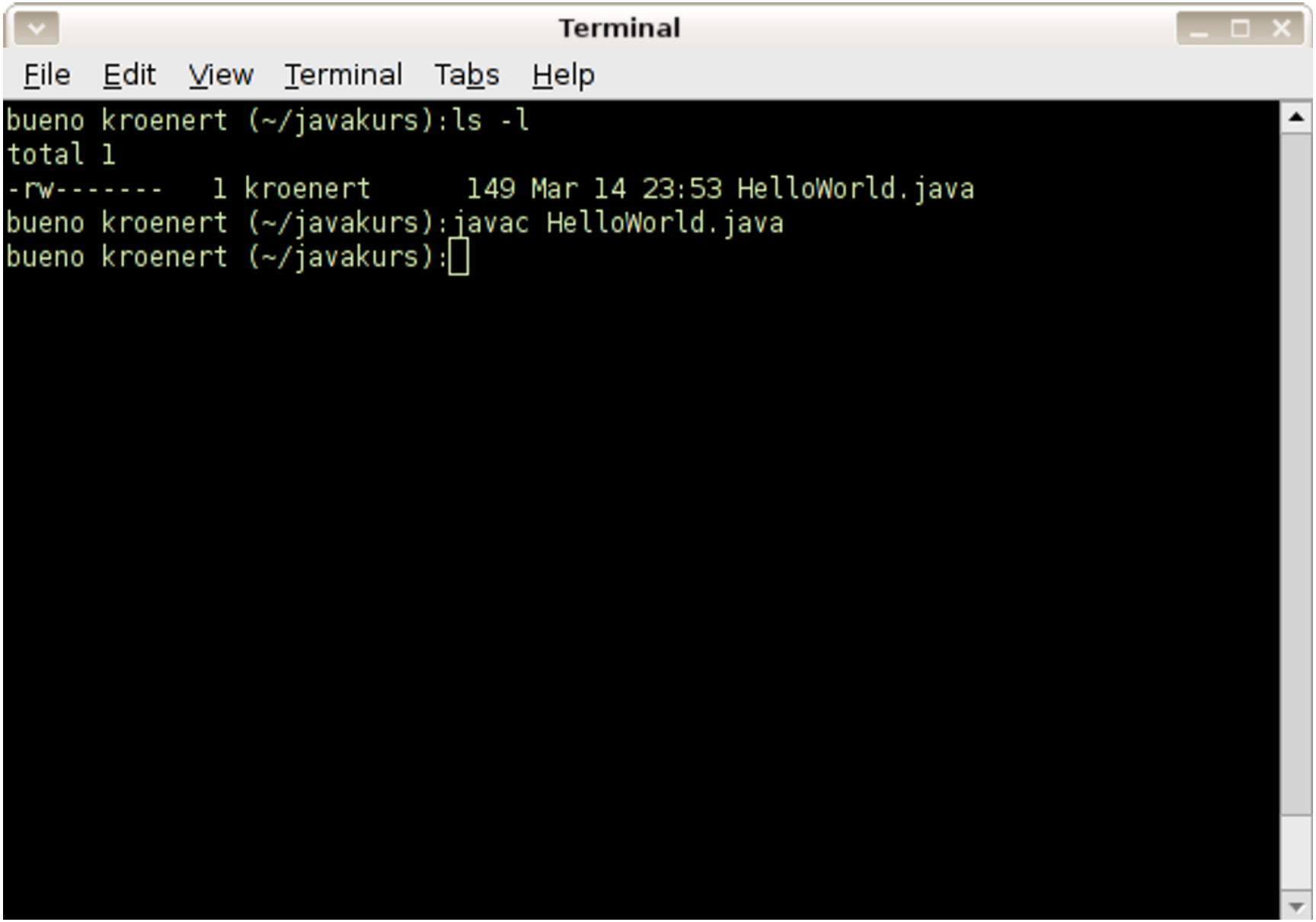
Ausführen

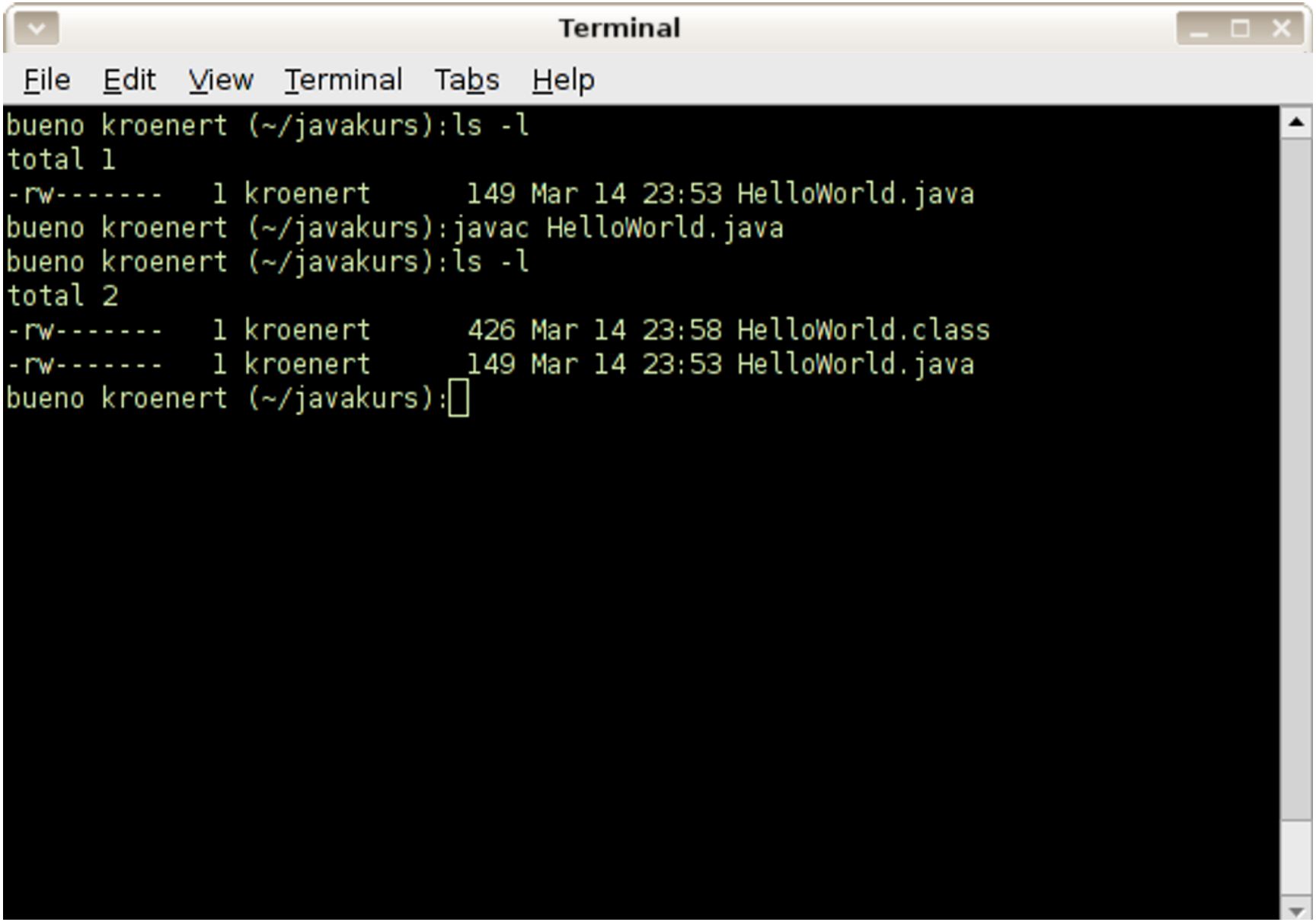
Shell

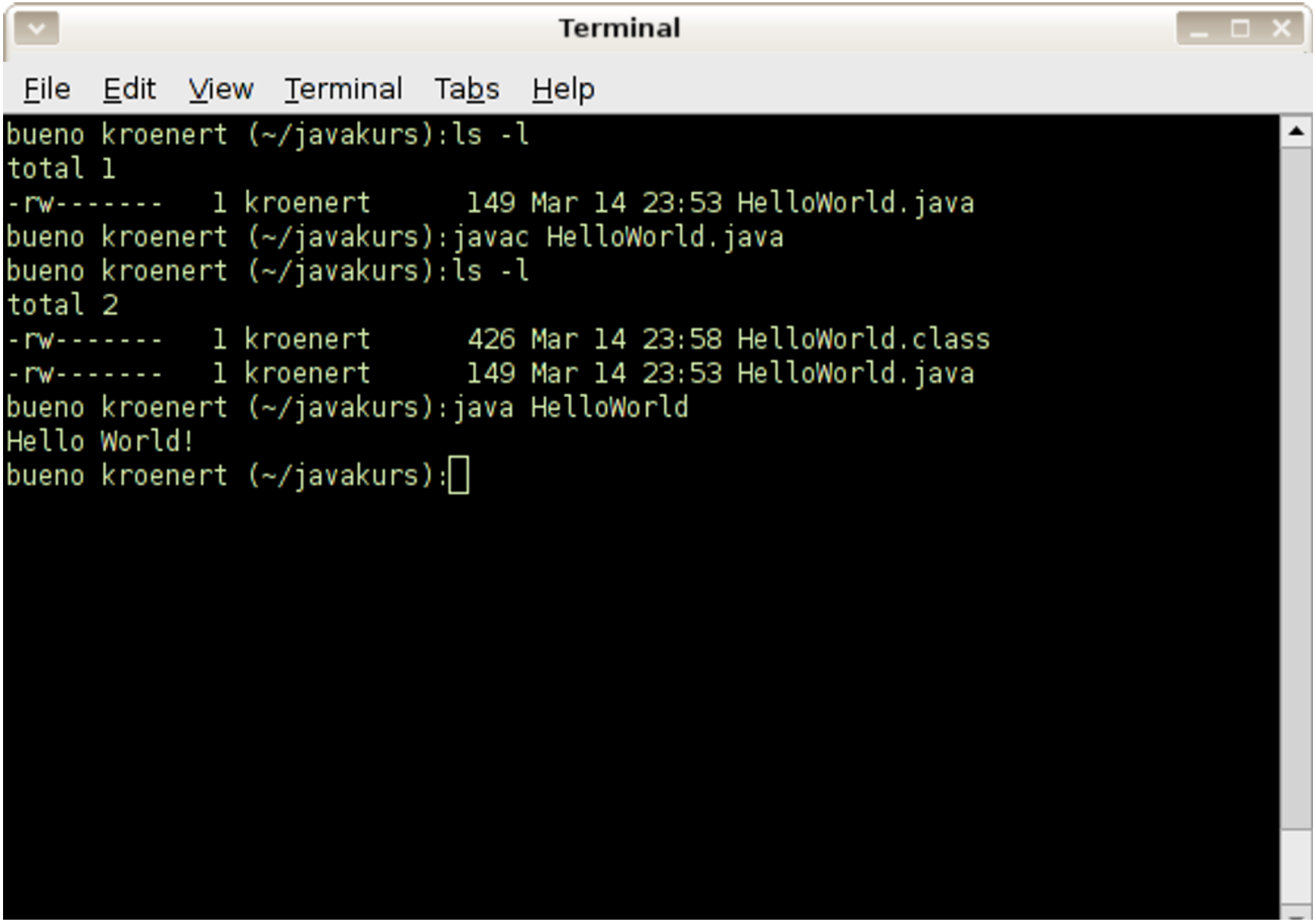
```
bueno kroenert: java HelloWorld
Hello World!
bueno kroenert:
```

- **java** ist die Java Virtual Maschine
- als Parameter wird der Klassenname übergeben
- die Ausgabe ist auf der Console zu sehen


Terminal
File Edit View Terminal Tabs Help
bueno kroenert (~/javakurs):ls -l
total 1
-rw------- 1 kroenert 149 Mar 14 23:53 HelloWorld.java
bueno kroenert (~/javakurs):


Terminal
File Edit View Terminal Tabs Help
bueno kroenert (~/javakurs):ls -l
total 1
-rw------- 1 kroenert 149 Mar 14 23:53 HelloWorld.java
bueno kroenert (~/javakurs):javac HelloWorld.java
bueno kroenert (~/javakurs):

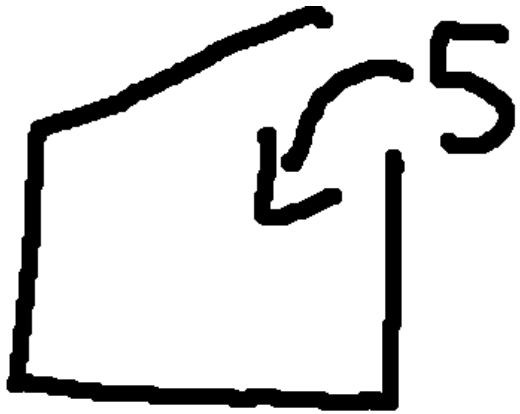

Terminal
File Edit View Terminal Tabs Help
bueno kroenert (~/javakurs):ls -l
total 1
-rw------- 1 kroenert 149 Mar 14 23:53 HelloWorld.java
bueno kroenert (~/javakurs):javac HelloWorld.java
bueno kroenert (~/javakurs):ls -l
total 2
-rw------- 1 kroenert 426 Mar 14 23:58 HelloWorld.class
-rw------- 1 kroenert 149 Mar 14 23:53 HelloWorld.java
bueno kroenert (~/javakurs):


Terminal
File Edit View Terminal Tabs Help
bueno kroenert (~/javakurs):ls -l
total 1
-rw-------   1 kroenert   149 Mar 14 23:53 HelloWorld.java
bueno kroenert (~/javakurs):javac HelloWorld.java
bueno kroenert (~/javakurs):ls -l
total 2
-rw-------   1 kroenert   426 Mar 14 23:58 HelloWorld.class
-rw-------   1 kroenert   149 Mar 14 23:53 HelloWorld.java
bueno kroenert (~/javakurs):java HelloWorld
Hello World!
bueno kroenert (~/javakurs):

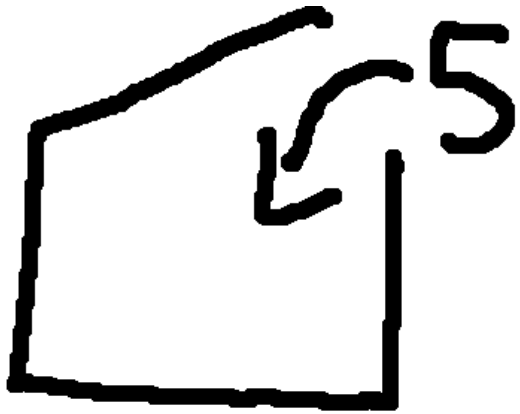
Variablen und Datentypen

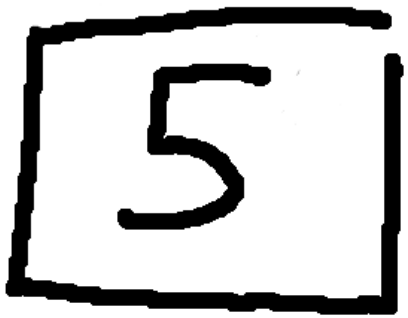
Variablen

Variablen

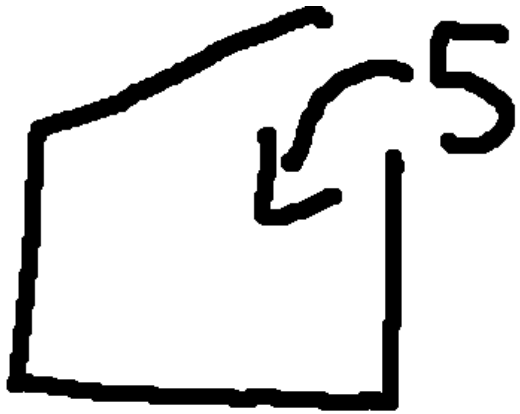


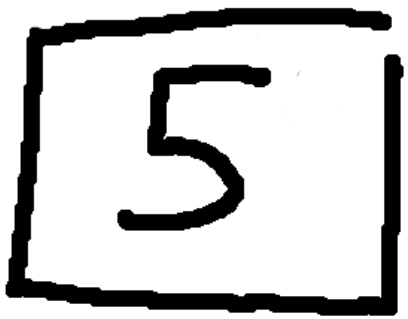
Variablen

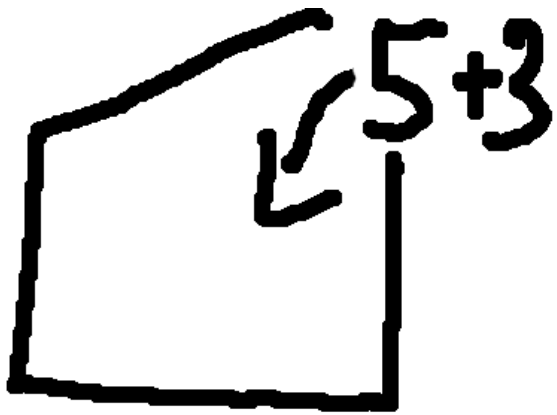




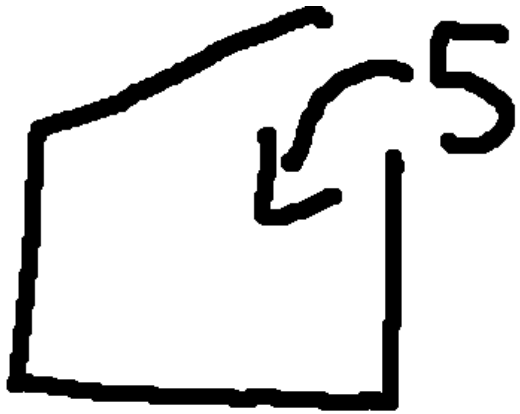
Variablen

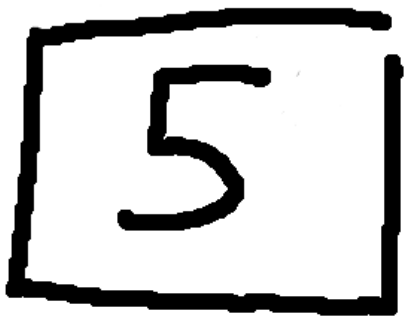


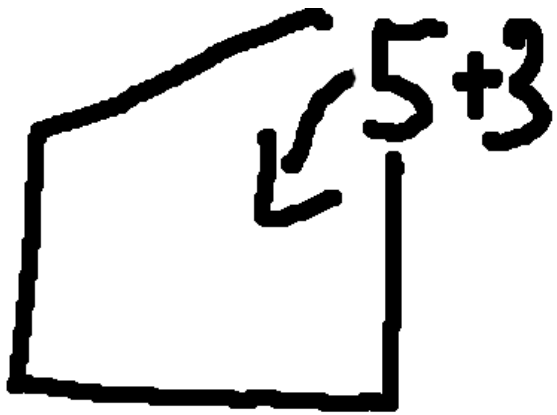


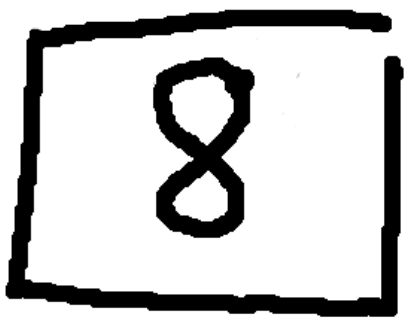


Variablen

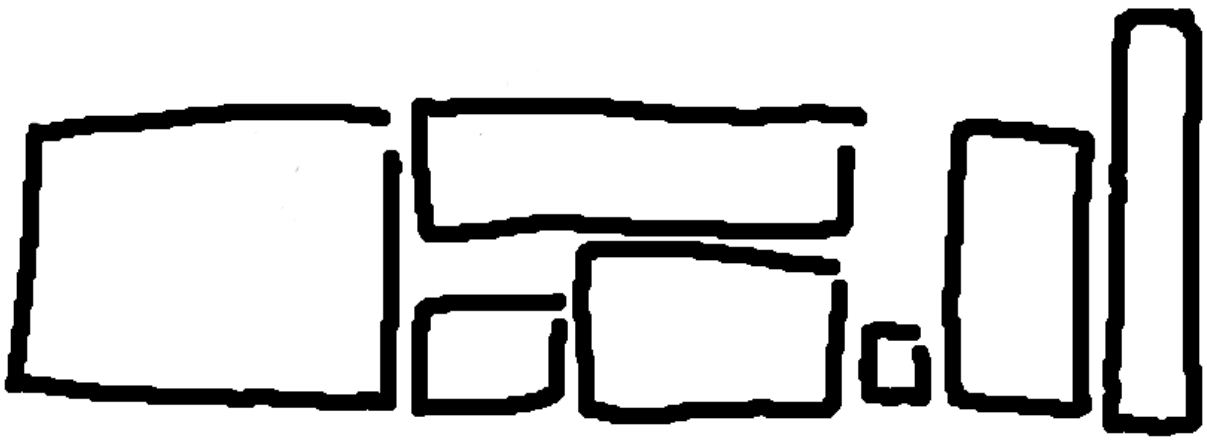








Datentypen



Datentypen – Integer

Variablen.java

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

- **int** steht für **Integer**, eine ganze Zahl
- = weist den rechten Wert der Variablen auf der Linken zu

Datentypen – Integer

Shell

```
bueno kroenert: javac Variablen.java
bueno kroenert: java Variablen
23
```

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

Datentypen – String

Variablen.java

```
1  public class Variablen {
2          public static void main(String[] args) {
3
4                  int age = 20;
5                  int number = 3;
6
7                  age = age + number;
8
9                  String message;
10                 message = ”My age is: ”;
11
12                 System.out.println(message + age);
13         }
14 }
```

- **String** ist eine Zeichenkette
- ” und “ markieren die Enden eines Strings

Datentypen – String

Shell

```
bueno kroenert: javac Variablen.java
bueno kroenert: java Variablen
My age is: 23
```

Datentypen – Double

Variablen.java

```
1  public class Variablen {
2          public static void main(String[] args) {
3
4                  double height = 1.75;
5
6                  String message = ”My height is ”;
7                  System.out.println(message + height);
8
9          }
10 }
```

- **double** ist eine Fließkommazahl

Datentypen – Double

Shell

```
bueno kroenert: javac Variablen.java
bueno kroenert: java Variablen
My height is 1.75
```

Datentypen – Boolean

Bool'sche Werte sind Wahrheitswerte. **true** und **false**

Variablen.java

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

Datentypen – Boolean

Shell

```
1 bueno kroenert: javac Variablen.java
2 bueno kroenert: java Variablen
3 Am I a smart Java hacker? false
```

:-(

Konventionen

- Variablennamen werden im so genannten camelCase geschrieben
- Der erste Buchstabe ist aber klein.

12

Zum Beispiel: **amIAJavaHacker**

Empfehlung:

- kurze und aussagekräftige Namen verwenden

Datentypen im Überblick

Typ	Wertebereich
`boolean result = false;`	{true;false}
`int age = 20;`	{-2147483648 ... 2147483647}
`double height = 1.75;`	$\{\pm 4,9 \cdot 10^{-324} \ldots \pm 1,7977 \cdot 10^{+308}\}$
`String message = "My age is";`	{ ""... endlich }

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

Operatoren

Operatoren

Operatoren.java

```
1   public class Operatoren {
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          }
12  }
```

- es gelten die üblichen Rechenregeln

Operatoren

Logische Operatoren

&&	und
||	oder
!	Negation

Arithmetische Operatoren

+	Addition
-	Subtraktion
/	Division
*	Multiplikation
%	Modulo

Fallunterscheidungen

Fallunterscheidungen

Fallunterscheidung.java

```
1   boolean condition = true;
2
3   if ( condition ) {
4           System.out.println(“wahr”);
5   }
6
7   if ( !condition ) {
8           System.out.println(“falsch”);
9   }
10
```

Fallunterscheidungen

Fallunterscheidung.java

```
1   boolean condition = true;
2
3   if ( condition ) {
4           System.out.println(“wahr”);
5   }
6
7   else {
8           System.out.println(“falsch”);
9   }
10
```

Fallunterscheidungen - Klausur

Klausur.java

```
1  int punkte = 74;
2
3  if ( punkte > 50 ) {
4          System.out.println("Du hast bestanden.");
5  }
6  else {
7          System.out.println("Du bist durchgefallen.");
8  }
```

Fallunterscheidungen - Klausur

Klausur.java

```
1  int punkte = 74;
2
3  if ( punkte > 50 ) {
4          System.out.println("Du hast bestanden.");
5  }
6  else {
7          System.out.println("Du bist durchgefallen.");
8  }
```

Shell

```
1  fiesta mmehner: javac Klausur.java
2  fiesta mmehner: java Klausur
3  fiesta mmehner: Du hast bestanden.
```

Fallunterscheidungen - Klausur erweitert

KlausurErweitert.java

```
1  boolean hausaufgabenGemacht = false;
2  int punkte = 74;
3
4  if ( punkte > 50 && hausaufgabenGemacht ) {
5          System.out.println("Du hast bestanden");
6  }
7  else {
8          if ( hausaufgabenGemacht) {
9                  System.out.println("Klausur wiederholen.");
10         }
11         else {
12                 System.out.println("Du bist durchgefallen.");
13         }
14 }
```

Fallunterscheidungen - Klausur erweitert

KlausurErweitert.java

```
1   boolean hausaufgabenGemacht = false;
2   int punkte = 74;
3
4   if ( punkte > 50 && hausaufgabenGemacht ) {
5           System.out.println("Du hast bestanden");
6   }
7   else if ( hausaufgabenGemacht) {
8           System.out.println("Klausur wiederholen.");
9   }
10  else {
11          System.out.println("Du bist durchgefallen.");
12  }
```

Fallunterscheidungen - Klausur erweitert

KlausurErweitert.java

```
1  boolean hausaufgabenGemacht = false;
2  int punkte = 74;
3
4  if ( punkte > 50 && hausaufgabenGemacht ) {
5          System.out.println("Du hast bestanden");
6  } else if ( hausaufgabenGemacht) {
7          System.out.println("Klausur wiederholen.");
8  } else {
9          System.out.println("Du bist durchgefallen."); }
```

Shell

```
1  fiesta mmehner: javac KlausurErweitert.java
2  fiesta mmehner: java Klausur
3  fiesta mmehner: Du bist durchgefallen.
```

Operatoren mit boolschem Rückgabewert

$<$	kleiner
$>$	größer
$==$	gleich
$<=$	kleiner gleich
$>=$	größer gleich

Kommentare

Benutzung von Kommentaren in Java

Comments.java

```
1   public class Operatoren {
2           public static void main(String[] args) {
3
4                   // Dies ist ein einzeiliger Kommentar
5
6                   /*
7                     Dieser Kommentar umfasst
8                     mehrere Zeilen
9                    */
10
11
12          }
13  }
```

Kommentare

Warum sind Kommentare sinnvoll?

- Sie helfen anderen den Quelltext zu verstehen
- Sie helfen dir deinen eigenen Quelltext auch ein Jahr später noch zu verstehen
- Du kannst Anmerkungen während des Programmierens festhalten (z.B. TODOs)

Wie es nicht geht

NoComment.java

```
1   int foo = 53;
2   int bar = 20;
3   int blubber = 10;
4
5   if ( ( foo - bar ) > blubber ) {
6           System.out.println(“ja”);
7   }
8   else {
9           System.out.println(“nein”);
10  }
```

Besser

RasenMaehen.java

```
1   /*
2    * Dieses Programm berechnet, ob du deinen Rasen mähen solltest
3    */
4   //alle Maßangaben in cm
5   int jetzigeLaenge = 53;
6   int erwuenschteLaenge = 20;
7   int maxUeberschuss = 10;
8
9   // Ist der Ueberschuss zu groß, sollte geschnitten werden
10  if ( (jetzigeLaenge - erwuenschteLaenge) > maxUeberschuss )
11  {
12          System.out.println("ja");
13  }
14  else {
15          System.out.println("nein");
16  }
```

Fehler

Fehler

Errors.java

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

Shell

```
1  fiesta mmehner: javac Errors.java
2  Errors.java:5: ';' expected
3            }
4            ^
5  1 error
```

Fehler

Errors.java

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

Fehler liegen meistens **vor** der Zeile, die der Compiler angibt.

Fehler

Errors.java

```
1  public class Errors {
2          public static void main(String[] args) {
3                  System.out.printl(”Hello World.”);
4          }
5  }
```

Shell

```
1  fiesta mmehner: javac Errors.java
2  Errors.java:3: cannot find symbol
3  symbol  : method printl(java.lang.String)
4  location: class java.io.PrintStream
5                  System.out.printl("Hello World.");
6                                    ^
7  1 error
```

Fehler

Errors.java

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

Fehler

Errors.java

```
1  public class Errors {
2          public static void main(String[] args) {
3
4                  if (true) {
5                          System.out.println(“Hallo“);
6          }
7  }
```

Shell

```
1  fiesta mmehner: javac Errors.java
2  Errors.java:7: ’}’ expected
3  }
4   ^
5  1 error
```

Fehler

Errors.java

```
1  public class Errors {
2          public static void main(String[] args) {
3
4                  if (true) {
5                          System.out.println(“Hallo“);
6                  }
7          }
8  }
```

Fehler

Errors.java

```
1  int age = 23;
2
3  if (age == 23) {
4          String message = “Hey, you’re 23!”;
5  }
6  System.out.println(message);
```

Shell

```
1  fiesta mmehner: javac Errors.java
2  Errors.java:6: cannot find symbol
3  symbol  : variable message
4  location: class Errors
5                System.out.println(message);
6                                   ^
```

Fehler

Errors.java

```
1  public class Errors {
2          public static void main(String[] args) {
3                  int age = 23;
4
5                  // Variable außerhalb des Blockes deklarieren
6                  String message = "";
7
8                  if (age == 23) {
9                          message = "Hey, you're 23!";
10                 }
11                 System.out.println(message);
12         }
13 }
```

Besonderheit bei Blöcken: Innerhalb eines Blockes initialisierte Variablen gelten nur innerhalb des Blockes, dahinter nicht mehr!

Fehler

Errors.java

```
1  public class Errors {
2          public static void main(String[] args) {
3
4                  int number = 1 / 0;
5          }
6  }
```

Shell

```
1  fiesta mmehner: javac Errors.java
2  fiesta mmehner:
```

- Compiler meckert nur bei Syntaxfehlern

Fehler

Shell

```
fiesta mmehner: java Errors
Exception in thread "main"
java.lang.ArithmeticException: / by zero
        at Errors.main(Errors.java:4)
```

- Ein Fehler der beim Ausführen auftritt, nennt sich **Runtime Error** (Laufzeitfehler)
- In Java heißen diese Exceptions (deutsch: Ausnahmen)
- Solche Fehler treten z.B. auch bei Endlosrekursionen auf

Fragen?

Viel Spaß beim Java-Programmieren!

Zur Erinnerung:

10:00 - 11:00 erster Vortrag

11:00 - 13:00 erste Übung

13:00 - 14:00 Pause (Tipp: Mensa)

14:00 - 15:00 zweiter Vortrag

15.00 - 17:00 zweite Übung

* die Zeiten sind alle c.t.

Quellenverzeichnis

Vielen Dank an:

[World at titlepage]; Name: **Nasa**; Source: `http://flickr.com/`
[Milan's Photo]; Name: **private**; Source:
[Thaddäus Photo]; Name: **private**; Source:
[1]; Name: **Sharp**; Source: `http://www.flickr.com/photos/sharples/21815958/`
[2]; Name: **MikeJ1971**; Source: `http://www.flickr.com/photos/mikej1971/154113222/`
[Telefunkengebäude]; Name: **TUB**; Source: `http://www.isti.tu-berlin.de/uploads/pics/telefunken.jpg`
[4]; Name: **debagel**; Source: `http://www.flickr.com/photos/35034360312@N01/316403365`
[5]; Name: **selva**; Source: `http://www.flickr.com/photos/35237096015@N01/24604141`
[6]; Name: **amazeman**; Source: `http://www.flickr.com/photos/49064193@N00/157195124`
[7]; Name: **bloqseven**; Source: `http://www.flickr.com/photos/bloqseven/33854882/`
[8]; Name: **iwouldstay**; Source: `http://www.flickr.com/photos/iwouldstay/85799041/`
[9]; Name: **johnny_automatic**; Source: `http://openclipart.org/media/files/johnny_automatic/2165`
[10]; Name: **miskan**; Source: `http://www.flickr.com/photos/37084659@N00/6786622`
[11]; Name: **markhoekstra**; Source: `http://www.flickr.com/photos/geektechnique/316790513/`
[12]; Name: **xikita**; Source: `http://www.flickr.com/photos/xikita/48647105/`