
Hello World


Milan


Felix


4!
Freitagsrunde

Was machen wir hier?

- Grundlagen von Java lernen
- Starthilfe für MPEG2

1

Was habt ihr davon?

- Javakenntnisse
- 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 ;-)
- ... und bekommen auch ein wenig Geld

Vortragsräume

1. und 2. Tag: **H 1058** (hier)

3. und 4. Tag: **MA 004**


Franklingebäude

Übungen

MA241 (Unixpool), **FR2516** (Notebookraum),
FR2517, **FR5083**

FR5087 (Notebookraum), **FR5535**, **FR5538**,
FR5539, **FR6514**

Übungsaufgaben:
`http://freitagsrunde.org/Javakurs`

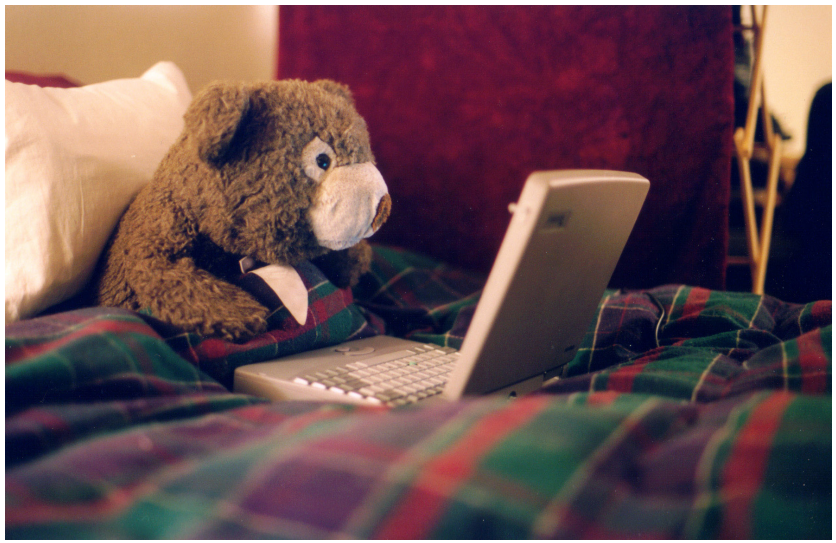
Bei Fragen und Problemen:

FR5535


Dan

Tagesablauf

09:00 - 10:00 erster Vortrag
10:00 - 12:00 erste Übung
12:00 - 13:00 Pause, Mensa
13:00 - 14:00 zweiter Vortrag
14.00 - 16:00 zweite Übung



5

* die Zeiten sind alle c.t.

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`

Los geht's...

Dann geht's jetzt (endlich) los...

8

Notizen machen nicht vergessen.

Inhalte & Ziele

Inhalte

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

Hello World

Beim Start wird die “main” Methode ausgeführt

HelloWorld.java

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

Klassennamen und Dateiname (ohne .java) müssen übereinstimmen

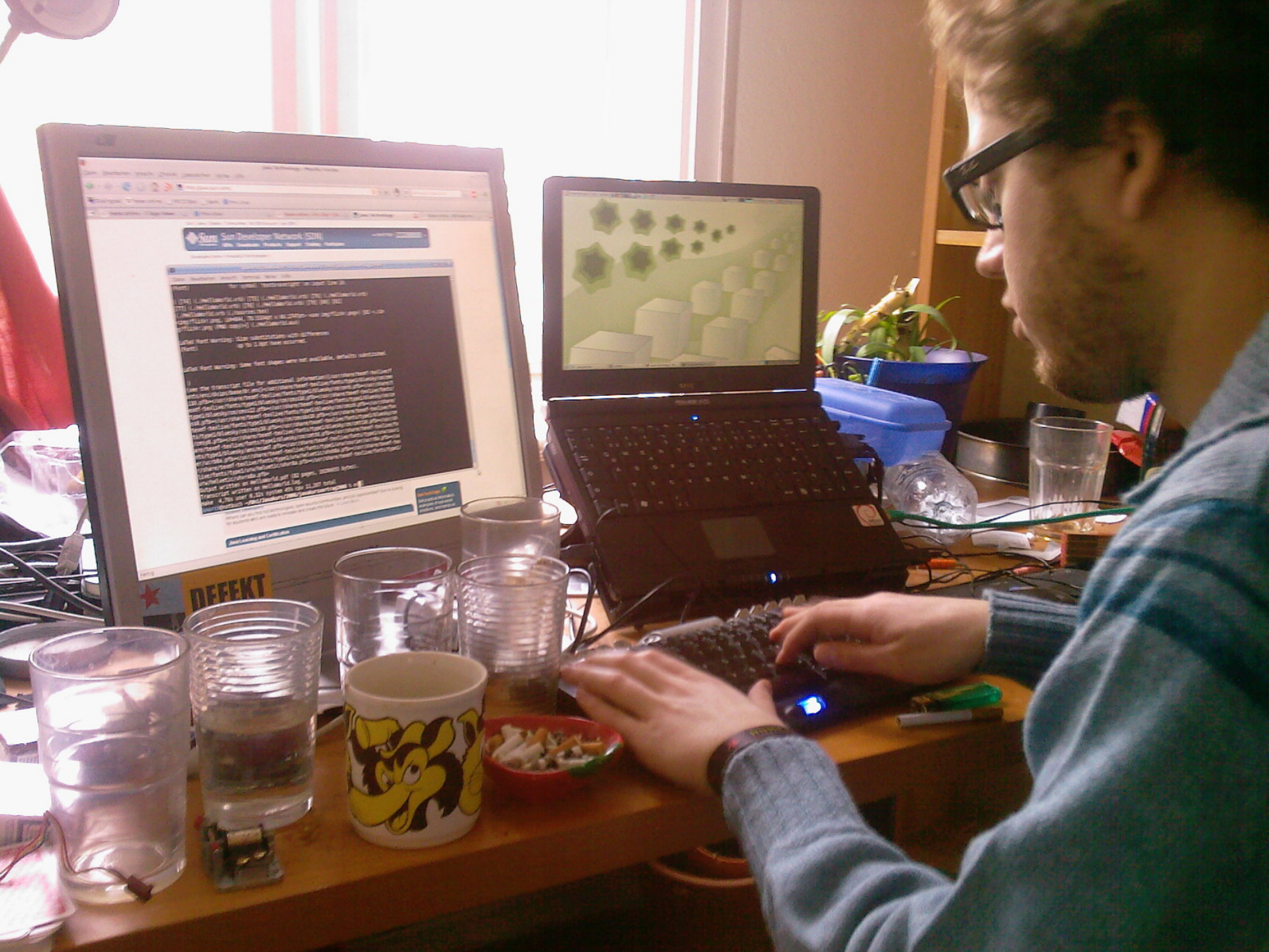
Abarbeiten von Befehlen

HelloWorld.java

```
1  public class HelloWorld {
2          public static void main(String[] args) {
3
4                  System.out.println(”foo”);
5                  System.out.println(”bar”);
6                  System.out.println(”Yeah :-)”);
7
8          }
9  }
```

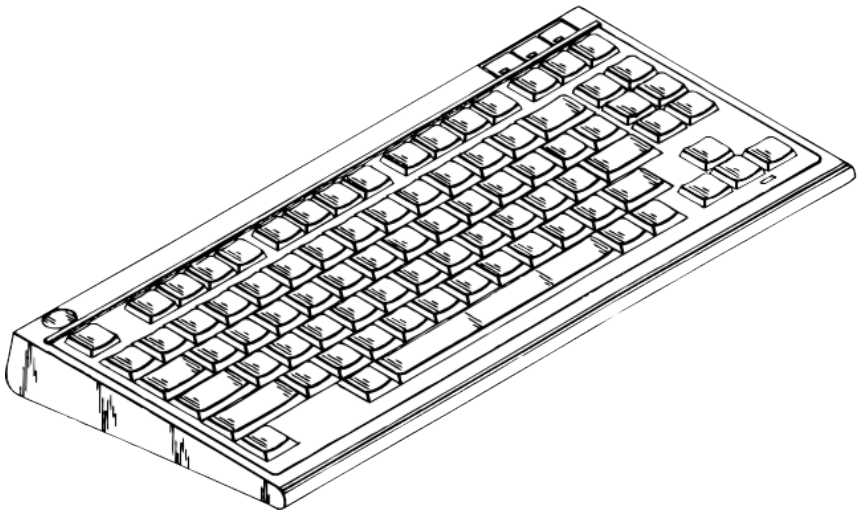
Befehle werden der Reihe nach abgearbeitet

Kompilieren und Ausführen


DEFEKT

Kompilieren und Ausführen

1. Einloggen



9

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

Kompilieren und Ausführen

1. Einloggen
2. eine Shell[1] öffnen

10

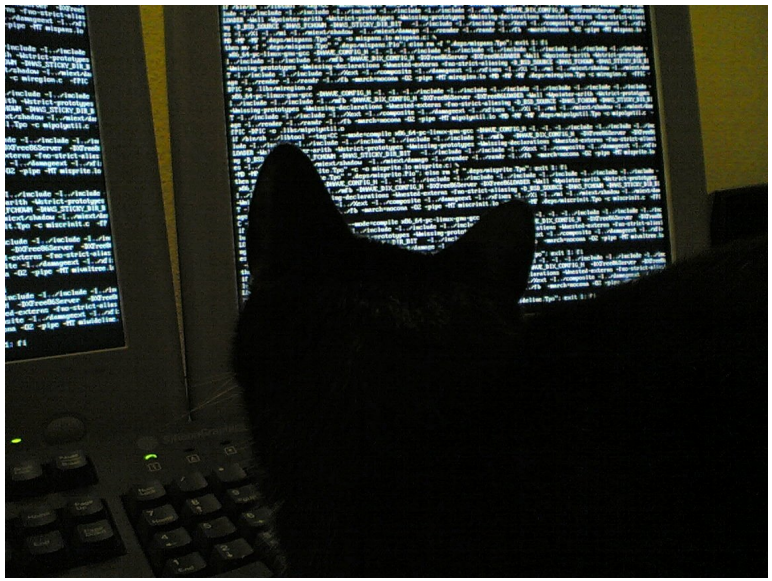
z.b. **gnome-terminal** (Gnome) oder **konsole** (KDE)

[1] 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 fiesta felixf: javac HelloWorld.java
2 fiesta felixf:
```

Bytecode

Shell

```
  fiesta felixf: ls -alh
-rw------- 1 felixf 426 Mar 10 13:51 HelloWorld.class
-rw------- 1 felixf 106 Mar 10 13:51 HelloWorld.java
```

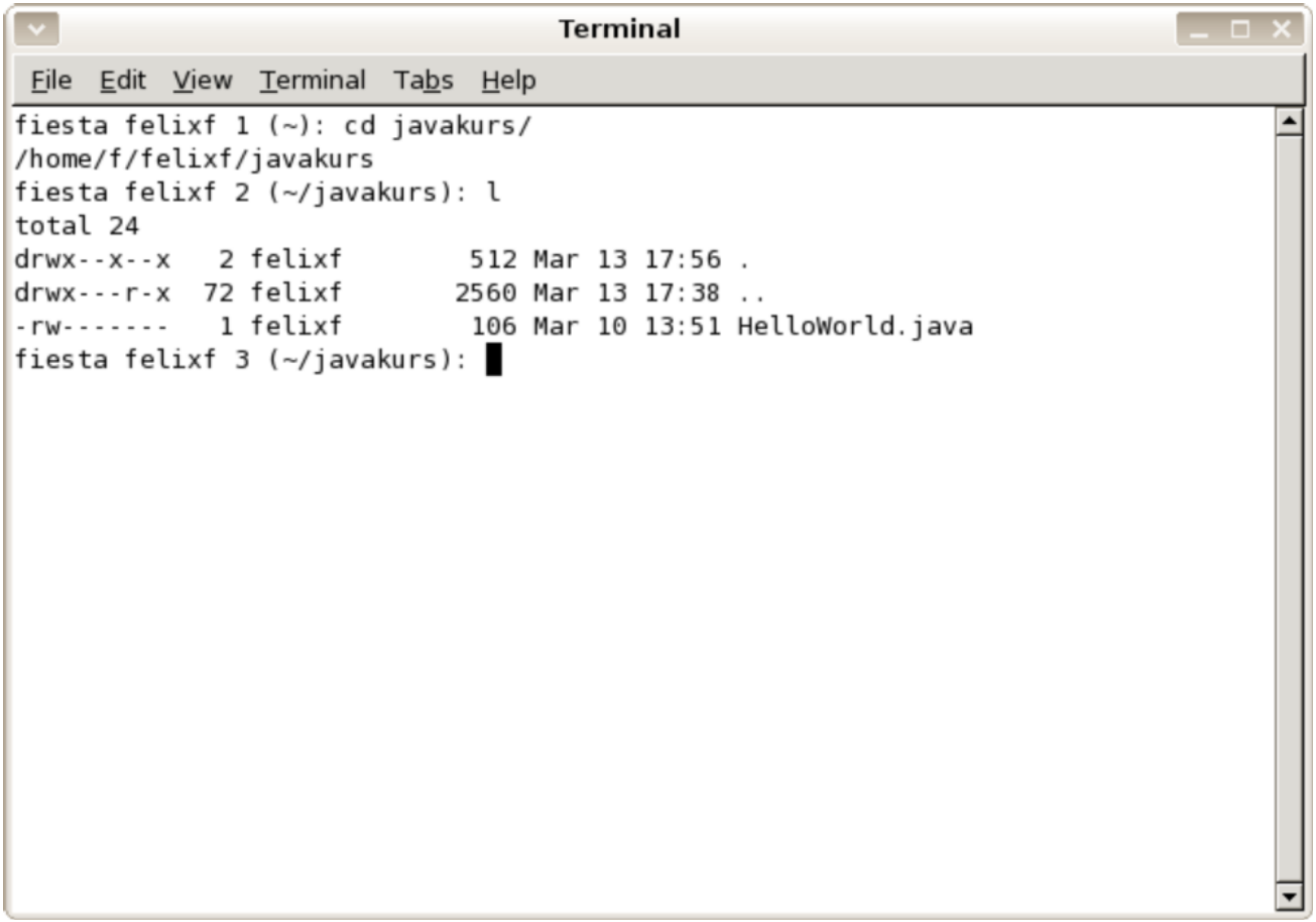
- Compilieren erzeugt .class Dateien, sog. Bytecode
- Bytecode kann mit einer **Java Virtual Machine** ausgeführt werden
- Bytecode ist maschinen**un**abhängig

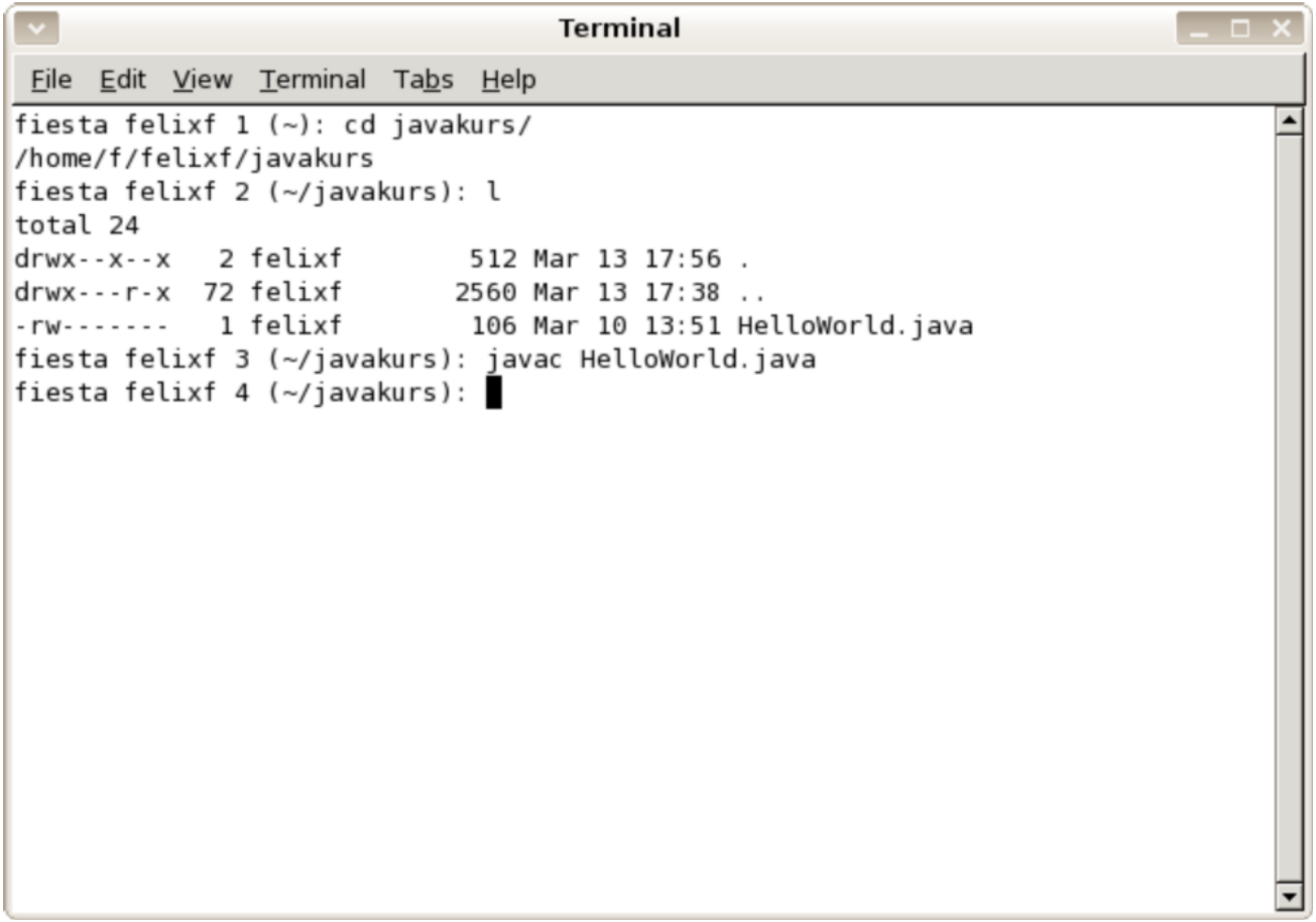
Ausführen

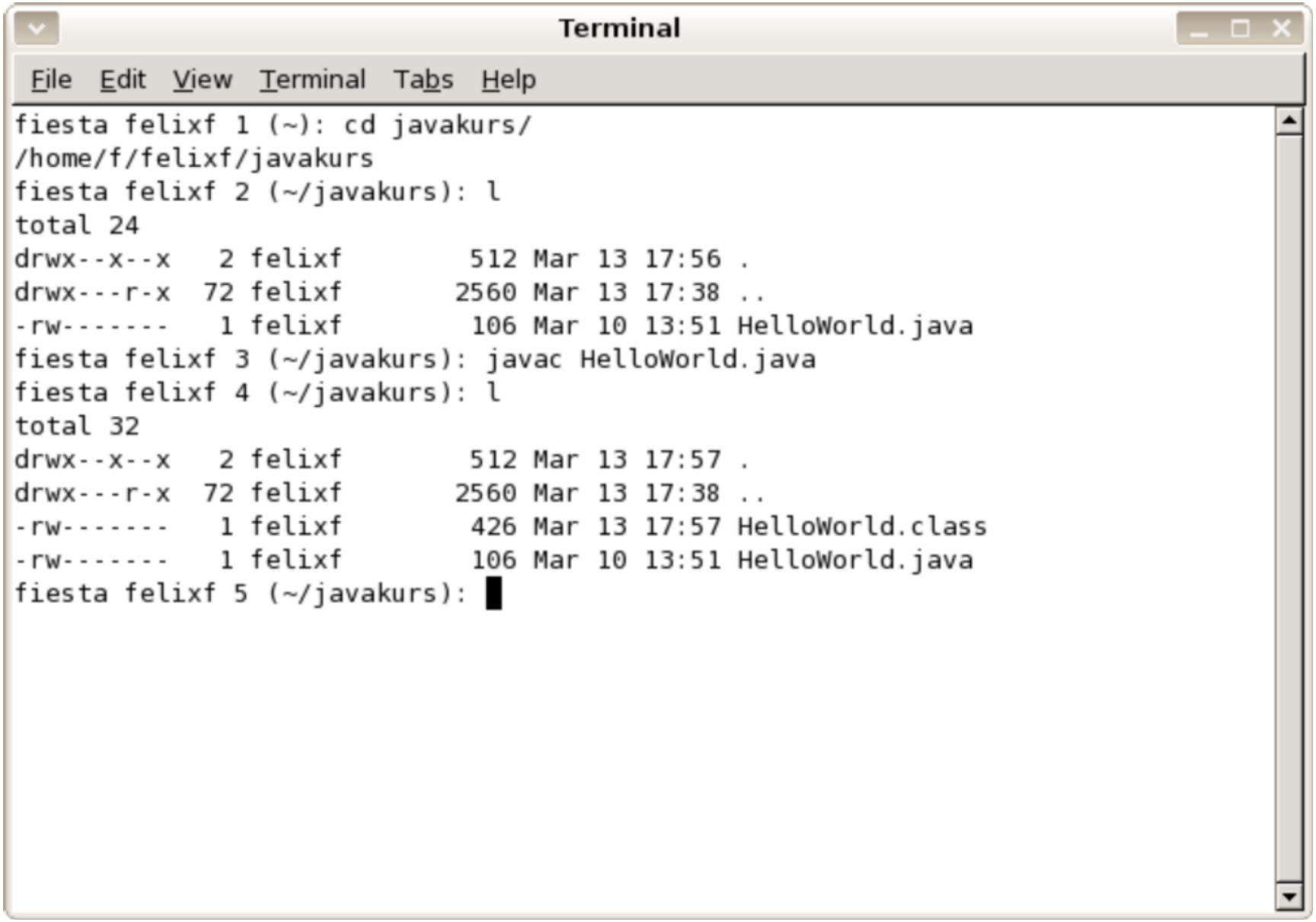
Shell

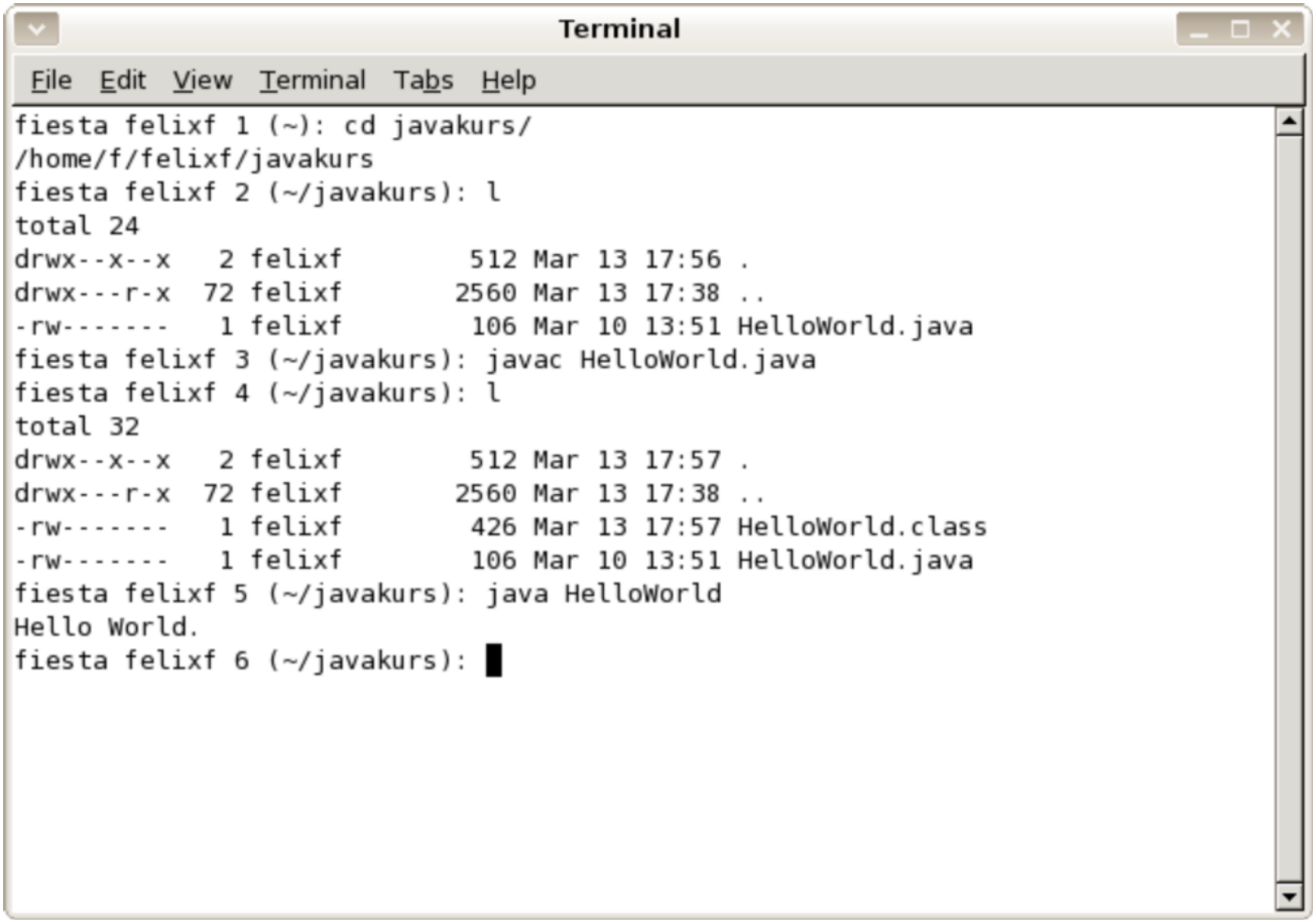
```
fiesta felixf: java HelloWorld
Hello World.
fiesta felixf:
```

- **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
fiesta felixf 1 (~): cd javakurs/
/home/f/felixf/javakurs
fiesta felixf 2 (~/javakurs): l
total 24
drwx--x--x   2 felixf   512 Mar 13 17:56 .
drwx---r-x 72 felixf   2560 Mar 13 17:38 ..
-rw-------   1 felixf   106 Mar 10 13:51 HelloWorld.java
fiesta felixf 3 (~/javakurs):


Terminal
File Edit View Terminal Tabs Help
fiesta felixf 1 (~): cd javakurs/
/home/f/felixf/javakurs
fiesta felixf 2 (~/javakurs): l
total 24
drwx--x--x   2 felixf   512 Mar 13 17:56 .
drwx---r-x 72 felixf   2560 Mar 13 17:38 ..
-rw-------   1 felixf   106 Mar 10 13:51 HelloWorld.java
fiesta felixf 3 (~/javakurs): javac HelloWorld.java
fiesta felixf 4 (~/javakurs):


Terminal
File Edit View Terminal Tabs Help
fiesta felixf 1 (~): cd javakurs/
/home/f/felixf/javakurs
fiesta felixf 2 (~/javakurs): l
total 24
drwx--x--x   2 felixf   512 Mar 13 17:56 .
drwx---r-x 72 felixf   2560 Mar 13 17:38 ..
-rw-------   1 felixf   106 Mar 10 13:51 HelloWorld.java
fiesta felixf 3 (~/javakurs): javac HelloWorld.java
fiesta felixf 4 (~/javakurs): l
total 32
drwx--x--x   2 felixf   512 Mar 13 17:57 .
drwx---r-x 72 felixf   2560 Mar 13 17:38 ..
-rw-------   1 felixf   426 Mar 13 17:57 HelloWorld.class
-rw-------   1 felixf   106 Mar 10 13:51 HelloWorld.java
fiesta felixf 5 (~/javakurs):


Terminal
File Edit View Terminal Tabs Help
fiesta felixf 1 (~): cd javakurs/
/home/f/felixf/javakurs
fiesta felixf 2 (~/javakurs): l
total 24
drwx--x--x   2 felixf   512 Mar 13 17:56 .
drwx---r-x 72 felixf   2560 Mar 13 17:38 ..
-rw-------   1 felixf   106 Mar 10 13:51 HelloWorld.java
fiesta felixf 3 (~/javakurs): javac HelloWorld.java
fiesta felixf 4 (~/javakurs): l
total 32
drwx--x--x   2 felixf   512 Mar 13 17:57 .
drwx---r-x 72 felixf   2560 Mar 13 17:38 ..
-rw-------   1 felixf   426 Mar 13 17:57 HelloWorld.class
-rw-------   1 felixf   106 Mar 10 13:51 HelloWorld.java
fiesta felixf 5 (~/javakurs): java HelloWorld
Hello World.
fiesta felixf 6 (~/javakurs):

Variablen und 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

```
fiesta felixf: javac Variablen.java
fiesta felixf: 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

```
fiesta felixf: javac Variablen.java
fiesta felixf: 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

```
fiesta felixf: javac Variablen.java
fiesta felixf: 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 Javahacker ? ";
10                 System.out.println(message + result);
11         }
12 }
```

Datentypen – Boolean

Shell

```
fiesta felixf: javac Variablen.java
fiesta felixf: java Variablen
Am I a smart Javahacker? false
```

:-(

Konventionen

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

Zum Beispiel: **javaRocks**

Empfehlung:

- kurze und aussagekräftige Namen verwenden

Datentypen im Überblick

Typ	**Wertebereich**
int	-2.147.483.648 ... 2.147.483.647
double	$\pm 4,9 \cdot 10^{-324}$ ... $\pm 1,7977 \cdot 10^{+308}$
boolean	true; false

12

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 square = 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

If.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 - Party

Party.java

```
1  int age = 15;
2
3  if ( age < 16 ) {
4          System.out.println("Go home at midnight");
5  }
6  else {
7          System.out.println("Party all night!");
8  }
```

Fallunterscheidungen - Party

Party.java

```
1  int age = 15;
2
3  if ( age < 16 ) {
4          System.out.println("Go home at midnight");
5  }
6  else {
7          System.out.println("Party all night!");
8  }
```

Shell

```
1  fiesta mmehner: javac Party.java
2  fiesta mmehner: java Party
3  fiesta mmehner: Go home at midnight
```

Fallunterscheidungen - Wettervorhersage

Weather.java

```
1  boolean itsRaining = true;
2  boolean sunIsShining = false;
3
4  if ( itsRaining && sunIsShining ) {
5          System.out.println("Go out. There's a rainbow!");
6  } else if ( sunIsShining ) {
7          System.out.println("Go out. The sun is shining!");
8  } else if ( itsRaining ) {
9          System.out.println("Don't go out. It's just raining!");
10 }
```

Fallunterscheidungen - Wettervorhersage

Weather.java

```
1  boolean itsRaining = true;
2  boolean sunIsShining = false;
3
4  if ( itsRaining && sunIsShining ) {
5          System.out.println(“Go out. There’s a rainbow!”);
6  } else if ( sunIsShining ) {
7          System.out.println(“Go out. The sun is shining!”);
8  } else if ( itsRaining ) {
9          System.out.println(“Don’t go out. It’s just raining!”);
10 }
```

Shell

```
1  fiesta mmehner: javac Weather.java
2  fiesta mmehner: java Weather
3  fiesta mmehner: Don’t go out. It’s just raining!
```

Operatoren mit boolschem Rückgabewert

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

Kommentare

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)

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  }
```

Wie es nicht geht

NoComment.java

```
1   int number1 = 53;
2   int number2 = 20;
3   int number3 = 10;
4
5   if ( ( number1 - number2 ) > number3 ) {
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  }
```

Blöcke

Blöcke

Den Rumpf der If-Abfrage nennt man Block.

```
1  if ( condition ) {
2
3          // Hier steht der Inhalt des Blocks
4
5
6
7  }
```

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

Blöcke – Ein Beispiel, wie es *nicht* geht

Block.java

```
1  public class Block {
2          public static void main(String[] args) {
3
4
5                  int age = 23;
6
7                  if (age == 23) {
8
9                          String message = “Hey, you’re 23!”;
10                 }
11
12                 System.out.println(message);
13
14         }
15 }
```

Blöcke – Ein Beispiel

Shell

```
1 fiesta felixf: javac Block.java
2 Block.java:12: cannot find symbol
3 symbol  : variable message
4 location: class Block
5                 System.out.println(message);
6                                    ^
7 1 error
```

Blöcke – Ein Beispiel, wie es geht

Block.java

```
1  public class Block {
2          public static void main(String[] args) {
3
4                  int age = 23;
5
6                  // Variable außerhalb des Blockes deklarieren
7                  String message = "";
8
9                  if (age == 23) {
10
11                         message = "Hey, you're 23!";
12                 }
13
14                 System.out.println(message);
15         }
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 felixf: 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 felixf: javac Errors.java
2  Errors.java:4: 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
4                  if (true) {
5                          System.out.println(“Hallo“);
6          }
7  }
```

Shell

```
1  fiesta felixf: 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                  int number = 1 / 0;
5          }
6  }
```

Shell

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

- Compiler meckert nur bei Syntaxfehlern

Fehler

Shell

```
fiesta felixf: 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 Programmieren!

Quellenverzeichnis

Vielen Dank an:

[World at titlepage]; Name: **Nasa**; Source: `http://flickr.com/`
[Dan's Photo]; Name: **Dan Levin**; Source:
[Felix's Photo]; Name: **Felix Friedrich**; Source:
[Milan's 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/`
[Franklingebäude]; Name: **Thomas**; Source: `https://wiki.freitagsrunde.org/Bild:Franklingebaeude.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: **victor_nuno**; Source: `http://www.flickr.com/photos/victornuno/253646322/`