Administratives

- **Nach der Vorlesung am 2.11.** erhalten die 125 Personen, die sich auf der Liste während der Vorlesung am 26.10. eingetragen haben, noch nachträglich das Javabuch gegen 40 Euro. **In unserem Sekretariat gibt es keine Bücher mehr.**

- Es kursiert anscheinend eine **Raubkopie** des Buchs. O'Reilly hat uns deshalb bereits angerufen(!). Buchkopieren ist **strafbar**. Kaufen Sie das Buch stattdessen.

- **Physik-Bachelor-**Studenten müssen die beiden Schein-Teilklausuren als Bachelorklausur mitschreiben. Ein Nachschreibetermin wird zur gleichen Zeit wie die Vordiplomsklausur angeboten. Für **andere Bachelorstudiengänge** sammeln wir noch die Informationen; geben Sie sicherheitshalber Ihre Übungsblätter auf jeden Fall ab!

- Die **Anzahl abgegebener Übungsblätter** beeinflußt direkt die **Hiwimittel**, die wir von der RWTH für Ihre Tutoren erhalten. Wenn Sie nicht abgeben, kann Ihre Tutorengruppe eingestellt werden.

- Bis Fr. 27.10. 17h können Sie sich noch nachträglich für eine **Übungsgruppe** online **anmelden.** Wenn Sie die **Übungsgruppe tauschen** müssen, wenden Sie sich bitte an Ihren Tutor.

1995

Objektorientiert

Write Once
Run Anywhere

Vom Quellcode zum Programm

Quellcode-Dokument in der Sprache Java, um ein Monster zu schaffen!

Java-Compiler

Monster

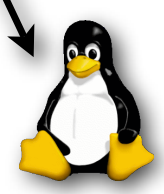
Java-Laufzeitumgebungen (JVM)...

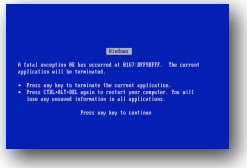
Das versteht nur der Computer!

Java-Bytecode:
Method Monster ()
0 aload_0
1 invokespecial...

...für Macs

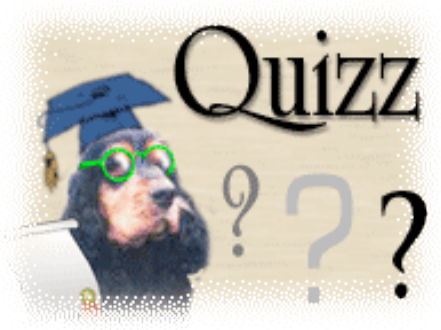
... für Handy und Java-Ring





... für Linux, Windows & andere Systeme

Anweisungen in Java: Eine Kostprobe




```
int alter;
alter = 29;
```

```
String vorname;
vorname = "Kevin";
```

```
int größe = 175;
```

```
System.out.println("Hallo!")
```

```
if (alter > 20) {
    System.out.println ("Alter Sack!");
}
```

Java-Programme: Das Kochrezept

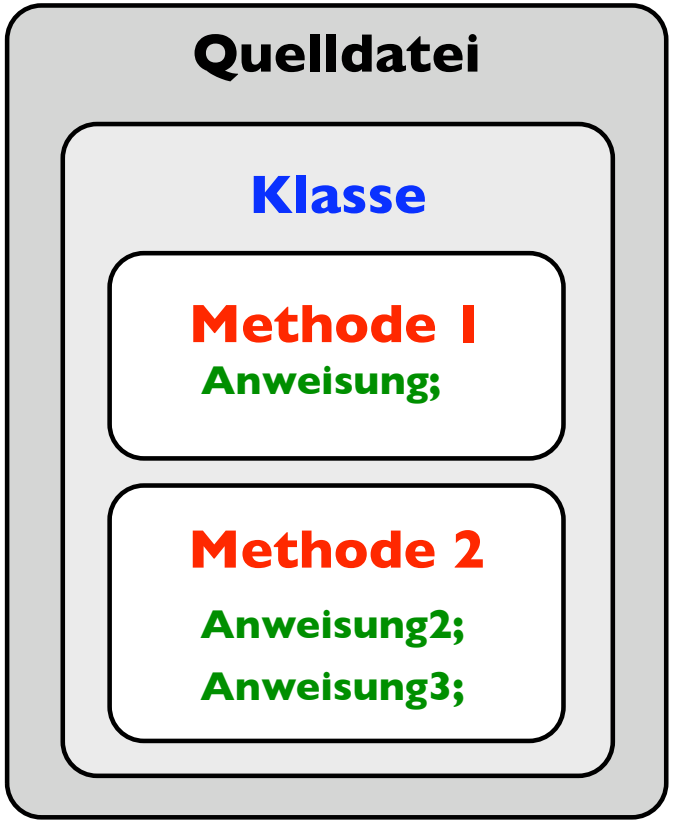
Sie brauchen eine **Quelldatei**...

...in der Sie eine **Klasse** definieren...

...in der Sie **Methoden** definieren...

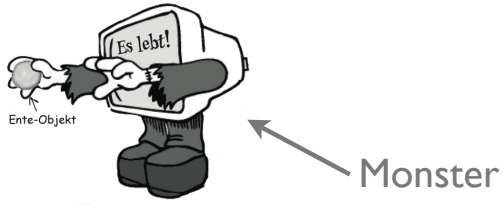
...die **Anweisungen** enthalten, um zu beschreiben, was die Methode macht.

"Puh!"



```
public class Monster {

}
```

Klasse



Methode

```
public class Monster {

  void erwache () {

  }
}
```

Anweisungen

```
public class Monster {

  void erwache () {
     Anweisung1;
     Anweisung2;
  }
}
```

Jede Klasse und jede Methode benutzt geschweifte Klammern **{ }**

Die main-Methode

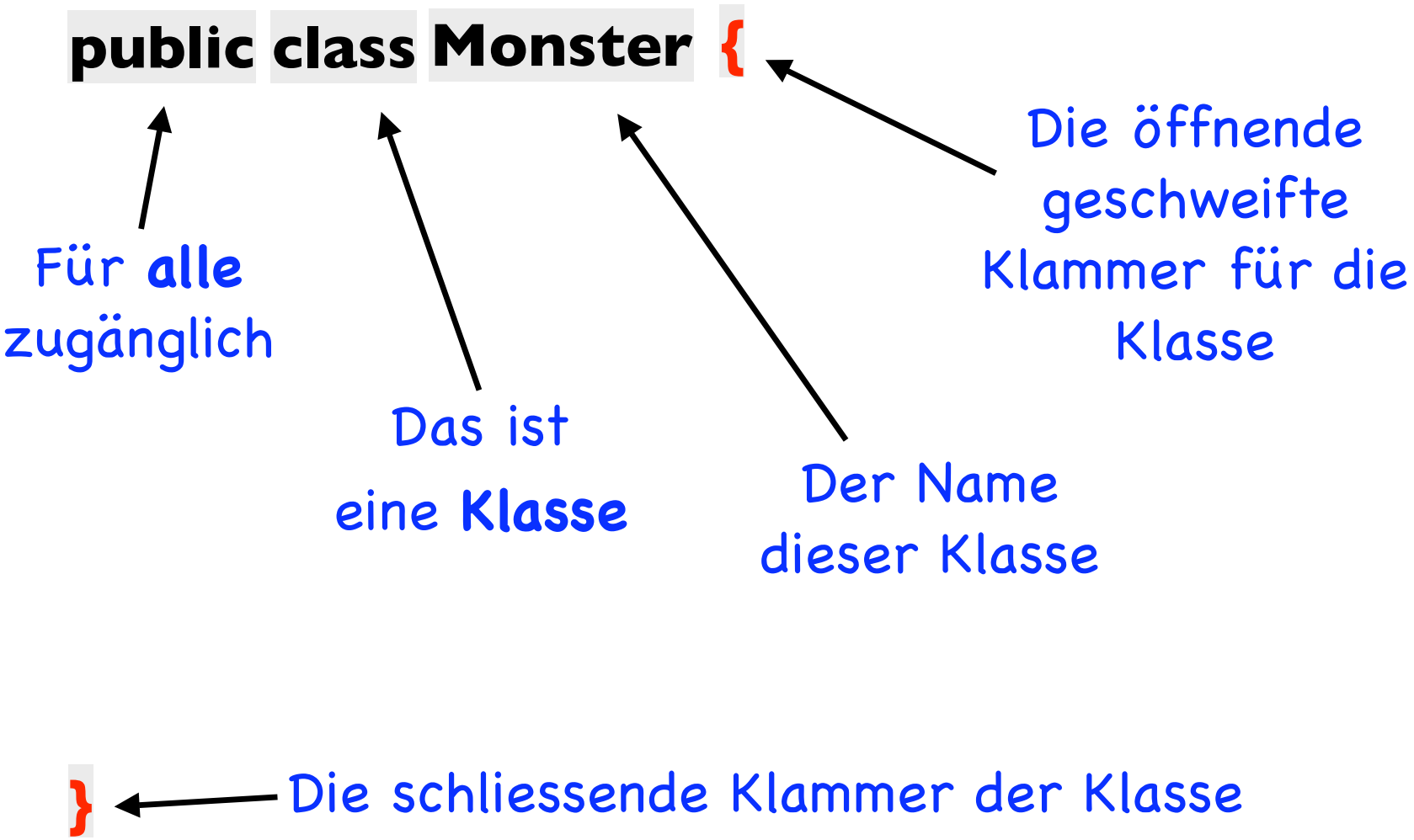
Die JVM fängt immer hier an!

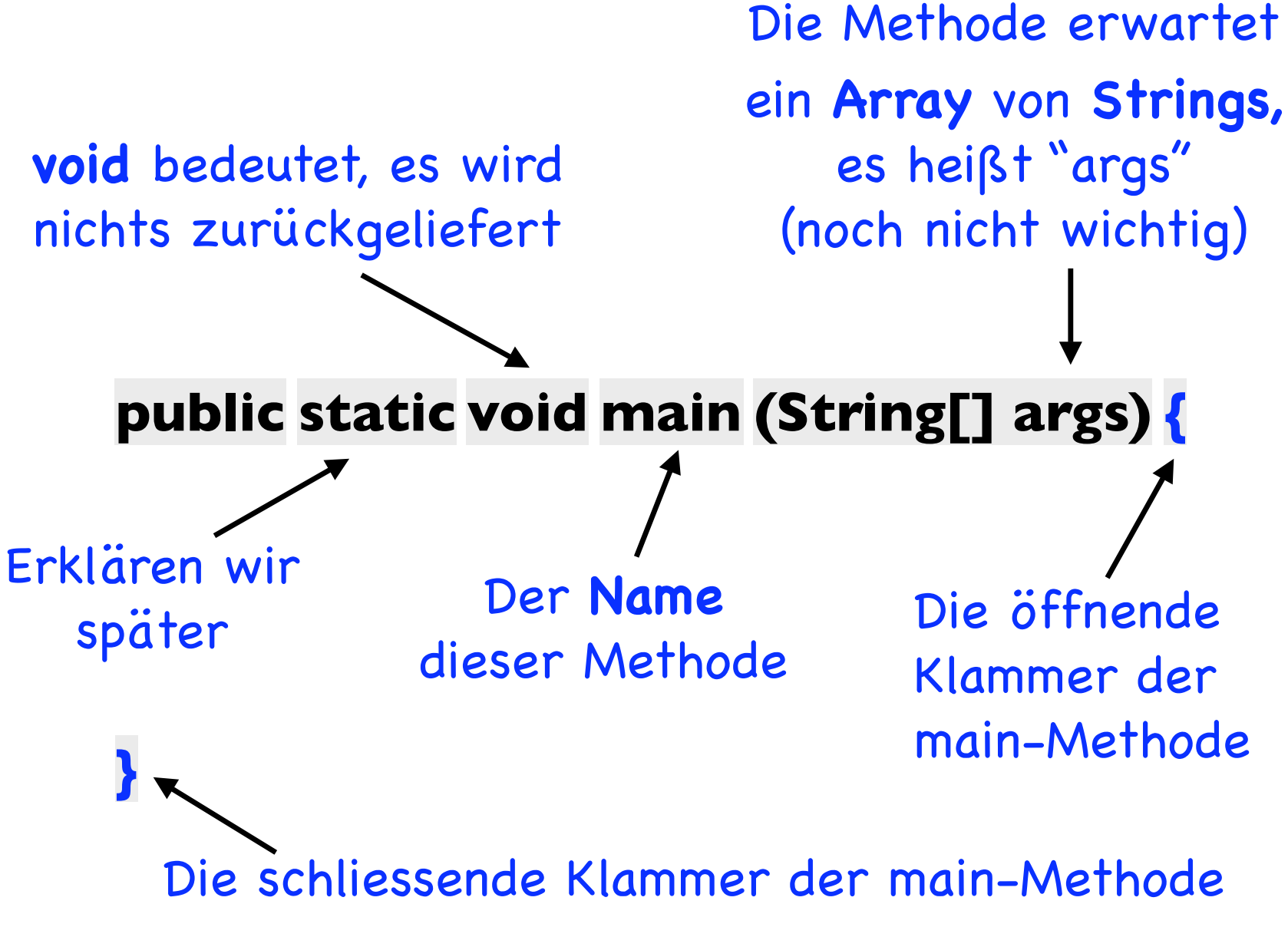
(z.B. um Ihr Monster zu erschaffen)

```
public static void main (String[] args) {

    // hier kommt Ihr Code rein

}
```

Sieht **immer** so aus und darf nie fehlen!

```
public class Monster {

    public static void main (String[] args) {

        System.out.println ("Buuuuhh!") ;

    }

}
```


public class Monster {
Für alle zugänglich
Das ist eine Klasse
Der Name dieser Klasse
Die öffnende geschweifte Klammer für die Klasse
}
Die schliessende Klammer der Klasse


Die Methode erwartet
ein **Array** von **Strings,**
es heißt "args"
(noch nicht wichtig)
void bedeutet, es wird
nichts zurückgeliefert
public static void main (String[] args) {
Erklären wir
später
Der **Name**
dieser Methode
Die öffnende
Klammer der
main-Methode
}
Die schliessende Klammer der main-Methode

Diese Anweisung gibt etwas auf dem Monitor aus

Jede Anweisung **MUSS** mit einem Semikolon enden!!

```
public static void main (String[] args) {

    System.out.println ("Buuuuhh!") ;

}
```

Der String, der ausgegeben wird

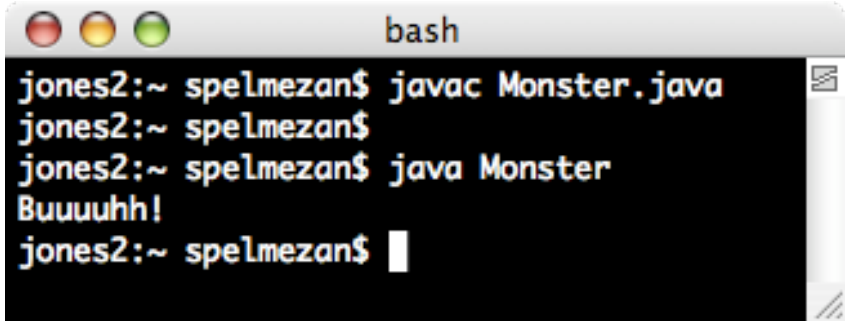
Demo

Speichern Sie den Code in der Datei **Monster.java**

Kompilieren Sie den Code mit **javac Monster.java**
Der Compiler erstellt die Bytecode-Datei Monster.class.

Starten Sie die JVM, um den Code auszuführen:
java Monster

Beim Starten wird der Dateiname
ohne die Endung .class geschrieben!



Syntax: Wie schreibe ich Java-Anweisungen?

// Jede Anweisung endet mit einem Semikolon.

```
x = x + 1;
```

// Mehrere Anweisungen auf einer einzigen Zeile und
// mit mehreren Leerzeichen sind OK.

```
x    =    3;    y =    21445;
```

Syntax: Wie schreibe ich Java-Anweisungen?

```
// Das ist ein Kommentar. Kommentare beginnen mit //

/* Kommentare können über mehrere Zeilen gehen.
Der Compiler ignoriert alles zwischen dem
Schrägstrich und dem Sternchen */

/** Kommentare können in Java auch Dokumentationstext
 *  generieren, der in der Java-API erscheint, um Klassen
 *  und Methoden zu beschreiben.
 */
```

Syntax

// Eine **Variable** wird mit einem **Typ** und einem **Namen** **deklariert**.

```
int gewicht;    String name;
```

// Deklariert eine Integer-Variable und **initialisiert** sie mit Wert 175.

```
int größe = 175;
```

Syntax

// Deklariert und initialisiert eine String-Variable.

String gegner1 = “Godzilla”;

doppelte Anführungszeichen

// Das geht aber nicht!

String gegner2 = “Space Godzilla”;

auf zwei Zeilen verteilt

Syntax

```
// Weist der Variable x den Wert von größe - 5 zu.

x = größe - 5;


// Der Zuweisungsoperator ist ein Gleichheitszeichen =
// Der Gleichheitsoperator besteht aus zwei Gleichheitszeichen ==

x = 15;
if (x == 15) { x = x * 2; }
```

Syntax

System.out.**print** ("Den Wert der Variablen x ausgeben: " + x);
System.out.**println** ("Einen Zeilenumbruch am Ende ausgeben.");

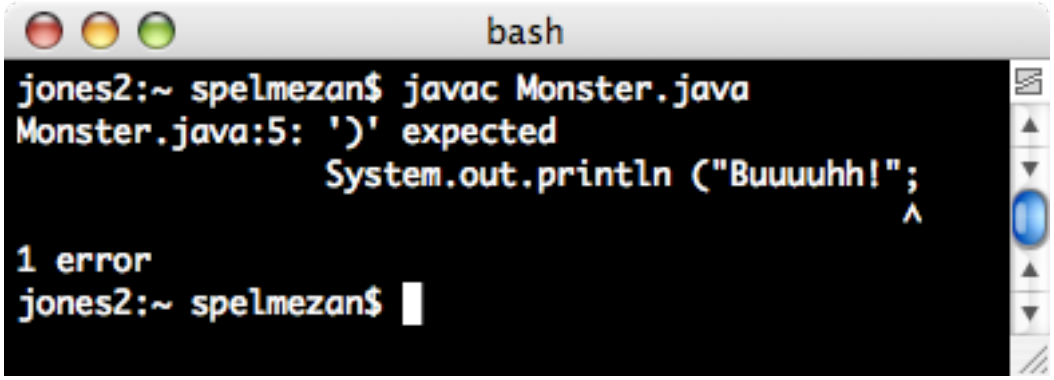
/* Klassen, Methoden und Codeblöcke werden mit einem
Paar geschweifter Klammern deklariert. */

```
public void starteRakete () {
        // Anweisungen, um die Rakete zu starten...
}
```

Der Java-Compiler

Erkennt nur syntaktische Fehler.

Fehlerbeschreibung mit Ausgabe der Datei, der Zeile und der Fehlerstelle ^



Und jetzt Sie!

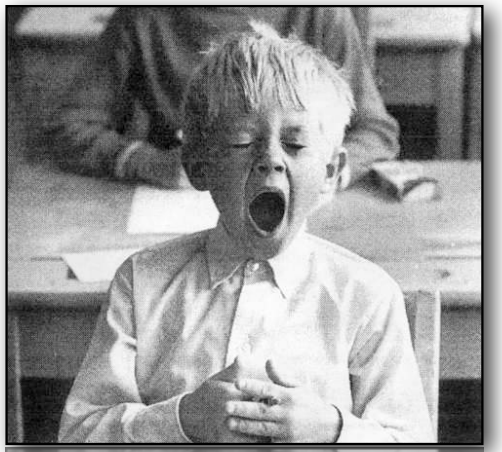

```
int variable1 = 200;
int variable2 = 300;
```

Ihre Aufgabe:
Vertauschen Sie den Wert der zwei int-Variablen.

Es geht so ähnlich wie mit Biergläsern...

Wollen Sie etwas mehrmals wiederholen, z.B. immer wieder den Wert von x ausgeben, können Sie das so machen:

```
System.out.println ("x ist " + x);

System.out.println ("x ist " + x);

... (immer wieder)

System.out.println ("x ist " + x);
```



Sie können aber auch Schleifen verwenden...

Solange eine Bedingung **wahr** ist, wiederhole.

while (Schleifenbedingung) { Anweisungsblock }

Die Schleifenbedingung wird zu einem Booleschen Wert ausgewertet:
true oder **false**

Boolesche Tests prüfen das Ergebnis eines Vergleichs, z.B.
<, >, ==, >=, <=

```
int summe = 0;  int zahl = 1;
int n = 100;

// wird nur ausgeführt, solange Bedingung wahr ist:
while (zahl <= n) {
      summe = summe + zahl;
      zahl ++;    // entspricht zahl = zahl + 1
}

System.out.println ("Die Summe ist " + summe);
```

Geht das?


```
int anzahl = 100;

while (anzahl) {
    // nächstes Zeichen aus der Datei lesen
    ...
    anzahl --;
}
```

Wiederhole, solange eine Bedingung **wahr** ist.

do { Anweisungsblock } while (Schleifenbedingung)


```
int summe = 0;  int zahl = 1;
int n = 100;

// wird mindestens einmal ausgeführt
do {
      summe = summe + zahl;
      zahl ++;
} while (zahl <= 100);

System.out.println ("Die Summe ist " + summe);
```

Verzweige, wenn Bedingung **wahr** ist.

```
if (Bedingung) {
    Anweisungsblock
}
else {
    Anweisungsblock
}
```

```
// Zuweisungsoperator
int anzahlStudenten = 510;

// Gleichheitsoperator
if (anzahlStudenten == 580) {
      System.out.println ("Paßt genau.");
} else if (anzahlStudenten < 580) {
      System.out.println ("Plätze frei.");
} else {
      System.out.println ("Aufstocken!");
}
```

int zahl1; int zahl2;

Ihre Aufgabe:
Schreiben Sie ein Programm, das den Wert der größeren int-Variablen bestimmt.

Spielen Sie Compiler! Gibt es Syntaxfehler?

```
int ergebnis = 1;
int anzahl = 5;

do {
    ergebnis = ergebnis * 2;
    if (ergebnis > 10) {
        ergebnis = ergebnis - anzahl;
        anzahl --;
    }
} while (anzahl > 0)

System.out.println ("Das Ergebnis ist " + ergebnis);
```

Spielen Sie Compiler! Gibt es Syntaxfehler?

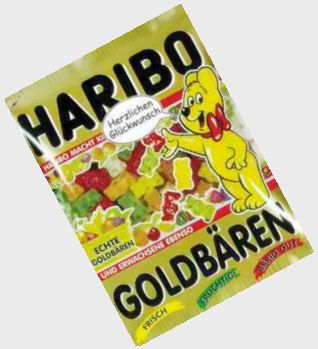
```
int ergebnis = 1;
int anzahl = 5;

do {
    ergebnis = ergebnis * 2;
    if (ergebnis > 10) {
        ergebnis = ergebnis - anzahl;
        anzahl --;
    }
} while (anzahl > 0);

System.out.println ("Das Ergebnis ist " + ergebnis);
```

Semikolon nicht vergessen!

Spielen Sie JVM!
Was ist das Ergebnis?

```
int ergebnis = 1;
int anzahl = 5;

do {
    ergebnis = ergebnis * 2;
    if (ergebnis > 10) {
        ergebnis = ergebnis – anzahl;
        anzahl = anzahl – 1;
    }
} while (anzahl > 0);

System.out.println ("Das Ergebnis ist " + ergebnis);
```

SMS mit Ergebnis an:

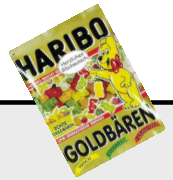


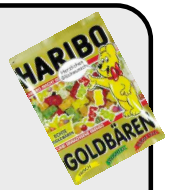
Variablenwerte in der while-Schleife:

```
 int ergebnis = 1;
int anzahl = 5;

do {
   ergebnis = ergebnis * 2;
   if (ergebnis > 10) {
      ergebnis = ergebnis – anzahl;
      anzahl = anzahl – 1;
   }
} while (anzahl > 0)

System.out.println ("Das Ergebnis
 ist " + ergebnis);
```

Ergebnis	Anzahl
2	5
4	5
8	5
11	4
18	3
33	2
64	1
127	0

Im Detail:
Berechnung von “ergebnis”





2
4
8
11 (16 - 5)
18 (22 - 4)
33 (36 - 3)
64 (66 - 2)
127 (128 - 1)

Schreiben Sie *lesbaren* Code

Kommentare

Einrückungen

aussagekräftige Variablennamen

Klassennamen fangen mit einem großen Buchstaben an, Variablen mit einem kleinen Buchstaben.

```
public class Chef {
        ...
      // Firmenboss bestimmen
      if (meinGehalt > deinGehalt) {
         boss = ich;
      }
      else {
         boss = du;
      }
      ...
}
```

Jetzt sind Sie wieder an der Reihe!

Schnappen Sie sich das Buch und lesen Sie die **Einführung** und **Kapitel 1**