

# Schleifen und Arrays

## Javakurs

Sebastian Dyrhoff  
Robert Buchholz

[freitagrunde.org/Javakurs](http://freitagrunde.org/Javakurs)  
24. März 2009

# Was ist das Problem?

```
System.out.println(5);  
System.out.println(4);  
System.out.println(3);  
System.out.println(2);  
System.out.println(1);  
System.out.println("Los!");
```

Aufgabe:  
Gib einen Countdown von  
5 bis 1 auf der Konsole aus.



Ausgabe:

5

4

3

2

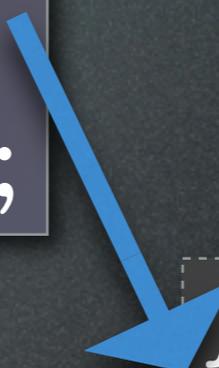
1

Los!

# Schönere Lösung als Schleife

```
int zahl = 5;
while (zahl >= 1) {
    System.out.println(zahl);
    zahl = zahl - 1;
}
System.out.println("Los!");
```

- Keine Zeilen gespart
- Kein doppelter Code



Ausgabe:

5

4

3

2

1

Los!

# Ein einfaches Beispiel



120

fakultaet

Aufgabe:

Berechne die Fakultät von 5 ( $5!$ ).

$$n! = 1 * 2 * 3 * \dots * n$$

# Ein einfaches Beispiel

```
int zahl = 1;
int fakultaet = 1;
while (zahl <= 5) {
    fakultaet = fakultaet * zahl;
    zahl = zahl + 1;
}
System.out.println("5! = " + fakultaet);
```

Aufgabe:

Berechne die Fakultät von 5 (5!).

$n! = 1 * 2 * 3 * \dots * n$



Ausgabe:  
5! = 120

```
int grenze = 5;
```

```
int zahl = 1;
```

```
while (zahl < grenze) {
```

```
    if (zahl % 2 == 1) {
```

```
        // zahl ist ungerade
```

```
        System.out.println(zahl);
```

```
    }
```

```
    zahl = zahl + 1;
```

```
}
```

Initialisierung

nicht vor  
Bedingung

Inkrement

Aufgabe:

Gib alle ungeraden, positiven  
Zahlen kleiner als grenze aus.

# Schleifen mit „for“

```
int grenze = 5;
for (int zahl = 1; zahl < grenze; zahl = zahl + 1) {
    if (zahl % 2 == 1) {
        // zahl ist ungerade
        System.out.println(zahl);
    }
}
```

Aufgabe:  
Vereinfache die Schleife  
von der vorherigen Folie.

Trennung von  
Zähler und Rumpf.

# for vs. while

```
for («Init»; «Bed»; «Inkr») {  
    «Rumpf»  
}
```

Yeah!

...

Geht beides!

```
«Init»  
while («Bed») {  
    «Rumpf»  
    «Inkr»  
}
```

# Ausgabe:

```
double  
while (  
    System  
    x = x  
}
```

```
// geh
```

```
for (d  
    Syst  
}
```

```
0.0  
0.1  
0.2  
0.3000000000000000  
0.4  
0.5  
0.6  
0.7  
0.7999999999999999  
0.8999999999999999  
0.9999999999999999  
1.0999999999999999  
1.2  
1.3  
1.4000000000000001
```

Rundungsfehler bei reellen Zahlen.

```
1) {
```

Aufgabe:  
Zähle x in



```
double x = 0.0;
while (x <= 1.0) {
    System.out.println(x);
    x = x + 0.1;
}
```

Aufgabe:

Zähle x in 0.1er-Schritten bis 1 hoch.

# Schleifenbedingungen

```
int zahl = 11;  
  
while (zahl != 9) {  
    zahl = zahl + 1;  
}
```

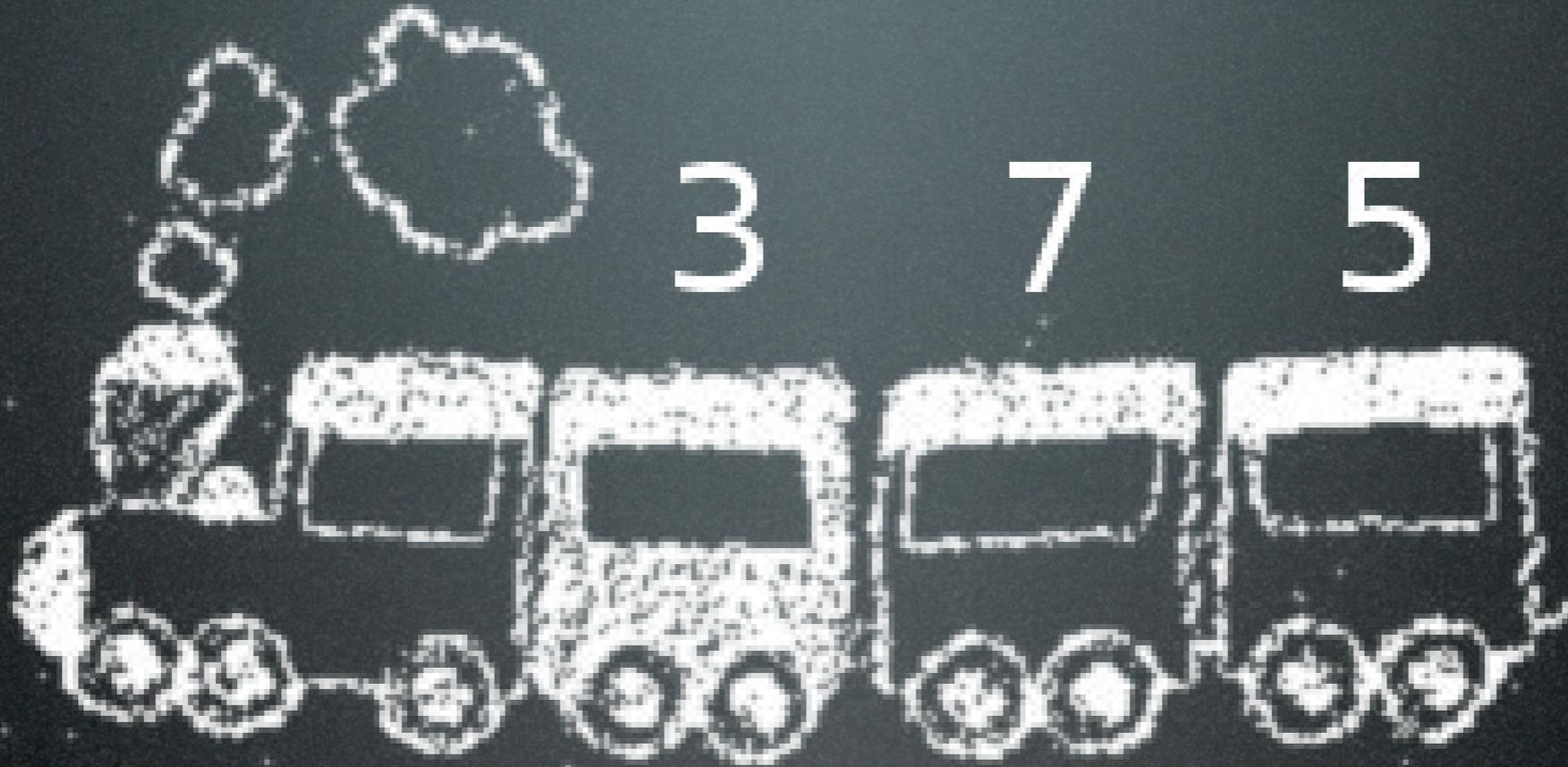
```
int zahl = 11;  
  
while (zahl < 9) {  
    zahl = zahl + 1;  
}
```

Der Rumpf wird oft nicht  
nie ausgeführt. oft wird  
die Schleife  
schlaufen?

Aufgabe:

Zähle zahl in Einerschritten bis 9 hoch.

# Nächstes Thema: Arrays



Aufgabe:

Denke an einen Zug.

Er hat Waggon, darin sind Fahrgäste.

# Zug mit Variablen

```
// Wir modellieren einen Zug  
int waggon1 = 3;  
int waggon2 = 7;  
int waggon3 = 5;
```

Das geht nur für  
genau drei Waggonns.

:-/

Frage:

Wie viele Fahrgäste sitzen in

```
int gaeste = waggon1 + waggon2 + waggon3;  
System.out.println(gaeste);
```

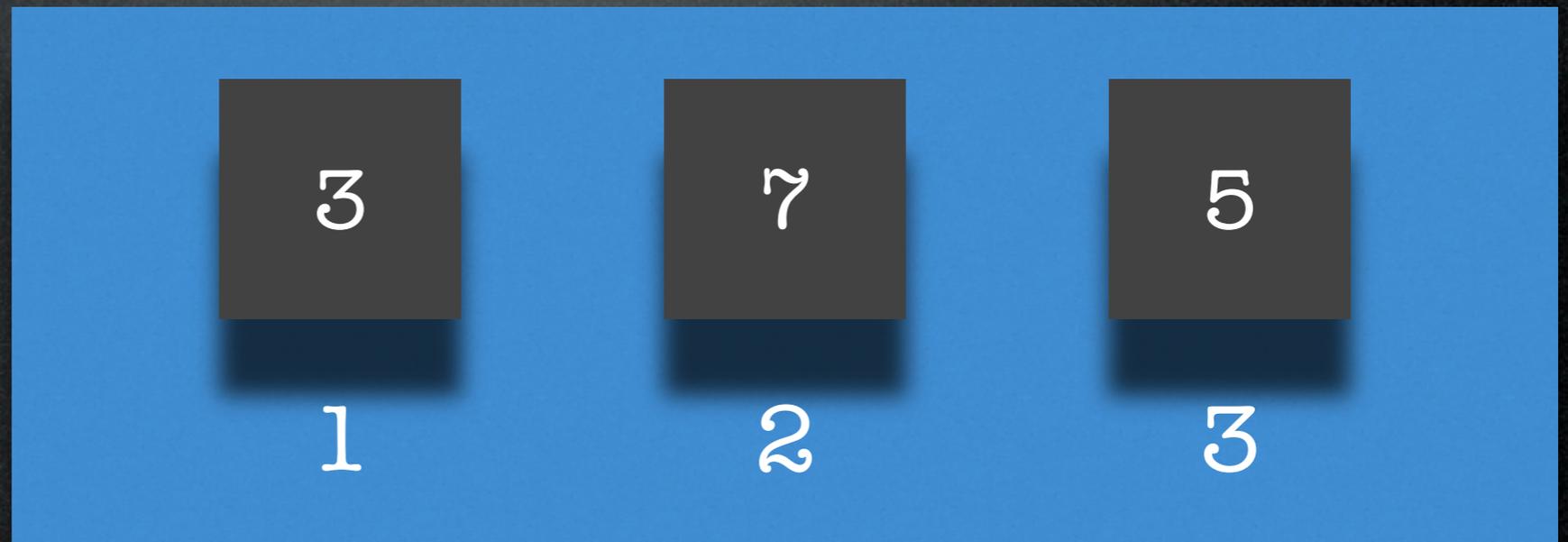
# Einzelne Variablen vs. Datenstruktur

3

7

5

waggon1 waggon2 waggon3



# Zug mit Array

```
// Wir modellieren einen Zug  
// zweiter Versuch  
zug[0] = 3;  
zug[1] = 7;  
zug[2] = 5;
```

Der Typ fehlt...

Frage:

Wie viele Leute

```
int gaeste = 0;  
for (int i = 0; i < zug.length; i = i + 1) {  
    gaeste = gaeste + zug[i];  
}  
System.out.println(gaeste);
```

# Zug, jetzt richtig

Die Länge in eckigen Klammern.

```
int [] zug = new int[3];  
zug[0] = 3;  
zug[1] = 7;  
zug[2] = 5;
```

Alle Elemente haben denselben Typ.

Indizierung von 0 bis n-1.

# Das Array verrät seine Länge

```
int [] zug = new int[3];  
zug[0] = 3;  
zug[1] = 7;  
zug[2] = 5;
```

```
System.out.println("Länge: " + zug.length);
```



Ausgabe:  
Länge: 3

# Don't #1

Initialisierung  
vergessen!

```
int zug[];  
zug[0] = 3; // Fehler!
```

```
$ javac ArrayFehler.java  
ArrayFehler.java:6: variable zug  
might not have been initialized
```

```
int zug[] = new int[3];  
zug[0] = 3;
```

1 error

# Don't #2

Über die Grenze.

```
int zug[] = new int[3];  
zug[3] = 10; // Fehler!
```

```
$ javac ArrayFehler.java  
$
```

```
$ java ArrayFehler  
Exception in thread "main"  
java.lang.ArrayIndexOutOfBoundsException: 3  
at ArrayFehler.main(ArrayFehler.java:6)
```

```
int zug[] = new int[4];  
zug[3] = 10;
```

# Telefonkartei



Aufgabe:

Organisiere deine Telefonnummern  
in einem Array.

# Telefonkartei als Array

```
int [] nummern = new int[4];  
nummern[0] = 92211;  
nummern[1] = 110;  
nummern[2] = 2342;  
nummern[3] = 31421386;
```

Aufgabe:

Finde heraus, wie oft eine bestimmte Nummer vorkommt.

```
int gesucht = 112;
int gefundene = 0;
for (int i = 0; i < nummern.length; i = i + 1) {
    if (nummern[i] == gesucht) {
        gefundene = gefundene + 1;
    }
}
System.out.println("Die Nummer kommt "
    + gefundene + " mal vor.");
```



Ausgabe:  
Die Nummer kommt 0 mal vor.

Aufgabe:

Finde heraus, wie oft eine  
bestimmte Nummer vorkommt.

# Was haben wir gelernt?

- Schleifen können Code mehrmals ausführen.
- Arrays enthalten Daten gleichen Typs.

Aufgabe:

Macht die Übungsaufgaben.

Und habt Spaß dabei :-)

This work is licensed under the Creative Commons Attribution-Share Alike 3.0 License.

To view a copy of this license, visit

<http://creativecommons.org/licenses/by-sa/3.0/>

or send a letter to Creative Commons, 543  
Howard Street, 5th Floor, San Francisco,  
California, 94105, USA.



Picture „card index box“ is copyright by  
Melanie Kuipers, malen-zeichnen.de

1-5 Hands „Asl alphabet gallaudet ann.svg“  
by Marnanel