

```
double tilgung, zinsen;
int jahre = 0;
double restschuld = auszahlung;

while (restschuld > 0) {
    zinsen = restschuld * zinsSatz / 100.;
    tilgung = rate - zinsen;
    restschuld = restschuld - zinsen;
    jahre = jahre + 1;
}
```

Was passiert hier ?

Obfuscation

```
while ((x = console.readInt()) != -1) {
    summe += x;
}
```

$x = e$ ist formal ein **Ausdruck** des Typs `int`. Sein Wert ist der zugewiesene Integer. Als **Seiteneffekt** erfolgt die Zuweisung dieses Werts an `x`. **Normalerweise** verzichtet man auf den Wert und verwendet die Zuweisung als **Ausdrucksbefehl** (**expression statement**).

Jeder beliebige Ausdruck kann auch als Statement verwendet werden. Der Wert des Ausdrucks wird dann verworfen.

Ich empfehle, die Zuweisung nur in Form eines Statements zu verwenden, rate also von obigem Code ab. Sie werden ihn aber zu sehen bekommen, falls Sie mal irgendwo als Programmierer jobben.

Wir wollen über die Tastatur Zahlen einlesen, solange bis `-1` eingegeben wird, und dann die Gesamtsumme ausgeben:

```
boolean fertig = false;
int x, summe = 0;

while (! fertig) {
    x = console.readInt();
    fertig = (x == -1);
    if (! fertig)
        summe = summe + x;
}
System.out.println(summe);
```

For-Schleife

Oft muss ein Statement eine bestimmte, feste Zahl von Malen durchlaufen werden.

```
int summe = 0;
for(int i = 0; i <= 100; i = i + 1) {
    summe = summe + i;
}
```

Jetzt ist `summe` gleich 5050.

```
for( init ; bedingung ; schritt )
    statement;
```

Hier sind *init* und *schritt* Wertzuweisungen oder Methodenaufrufe, die der Initialisierung und der Aktualisierung dienen.

bedingung ist ein Boole'scher Ausdruck.

Semantisch ist das **for**-Statement äquivalent zu

```
init;
while ( bedingung ) {
    statement;
    schritt;
}
```

```
String t = "";
for (int i = 0 ; i < s.length() ; i++) {
    t = s.charAt(i) + t;
```

Wir können den String auch vom Ende her durchgehen:

```
String t = "";
for (int i = s.length()-1 ; i >=0 ; i--) {
    t = t + s.charAt(i) ;
```

Merke: Im Rumpf einer While oder For-Schleife ist die Bedingung **immer** erfüllt.

Unmittelbar nach einer While oder For-Schleife ist die Bedingung immer falsch.

In der Klasse **String** gibt es die Methode **length** und **charAt**.

Man verwendet sie so:

"Matthias".length() ist 8

"Matthias".charAt(2) ist 't'

Die Methode **charAt** liefert ein Ergebnis vom Typ **char**. Man kann **chars** mit **+** an strings anhängen.

Wir wollen jetzt einen beliebigen String **s** umdrehen, also aus **Matthias** soll **saihttaM** werden.

Dazu müssen wir **s** der Reihe nach durchgehen und die einzelnen Zeichen in umgekehrter Reihenfolge aneinanderhängen.

Ein String enthalte einen Double, einen Integer und einen String, durch Leerzeichen getrennt. Wie kann man diese Daten herauslesen ?

Beispiel: "5 7.0E12 Johanna" besteht aus drei Tokens: "5", "7.0E12" und "Johanna".

Die Klasse **StringTokenizer** gestattet es, einen String in seine "Tokens" zu zerlegen. Ein Token ist ein Teilstring, der kein Leerzeichen enthält.

```
import java.util.StringTokenizer;

/* s der zu zerlegende String */
StringTokenizer tokenizer = new StringTokenizer(s);

while (tokenizer.hasMoreTokens()) {
    String token = tokenizer.nextToken();
    /* hier kann man token verarbeiten */
}
```

Wir wollen alle eingegebenen Wörter der Reihe nach ausgeben, jedes in einer extra Zeile.

Dazu lesen wir die Wörter zeilenweise ein und zerlegen jede Zeile in Tokens, die wir jeweils ausgeben.

Lösung

```
public class Woerter {
    public static void main(String[] args) {
        ConsoleReader console = new ConsoleReader(System.in);
        boolean fertig = false;
        String line;
        while ( !fertig ) {
            line = console.readLine();
            fertig = (line == null);
            if (!fertig) {
                StringTokenizer tokenizer = new StringTokenizer(line);
                while (tokenizer.hasMoreTokens()) {
                    System.out.println(tokenizer.nextToken());
                }
            }
        }
    }
}
```

Was ist null?

Jeder Objektausdruck, Variable oder zusammengesetzt, kann auch den Wert **null** haben. Das bedeutet, dass er auf kein Objekt verweist.

Die Methode `readLine` liefert **null** zurück, wenn keine Zeile zu lesen mehr vorhanden ist.

Etwa nach Drücken von Strg-D. (“Dateiende”).

Umlenkung von Ein-/Ausgabe

Man kann die Ausgabe eines Programms in eine Datei umlenken und die Eingabe von einer Datei nehmen:

```
java Woerter < eingabe.in > ausgabe.out
```

liest statt von der Tastatur aus der Datei `eingabe.in` und schreibt statt auf den Bildschirm auf `ausgabe.out`.

```
java Woerter < eingabe.in
```

geht auch und ebenso

```
java Woerter > ausgabe.out
```

ja sogar

```
java Woerter < eingabe.in | sort | java Unique > ausgabe.out
```

wenn etwa `java Unique` **aufeinanderfolgende** Dubletten entfernt.

Mit `|` (pipe) lenkt man die Ausgabe eines Programms direkt in ein anderes Programm als Eingabe um.

Einführung in die Informatik: Programmierung und Softwareentwicklung

Lösung

```
public class Powers{
    public static void main(String[] args){
        final int COLUMN_WIDTH = 10;
        for (int x = 1; x <= 10; x++) {
            for (int y = 1; y <= 8; y++) {
                int p = Math.round(Math.pow(x,y));
                String pstr = "" + p;
                while(pstr.length() < COLUMN_WIDTH)
                    pstr = " " + pstr;
                System.out.print(pstr);

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

Einführung in die Informatik: Programmierung und Softwareentwicklung

Geschachtelte Schleifen

Im Rumpf einer Schleife kann wieder eine solche stehen.

Anwendungsbeispiel: Ausgabe einer Tabelle der Potenzen x^y für $x = 1..10$, $y = 1..8$.

1	1	1	1	1	1	1	
2	4	8	16	32	64	128	
3	9	27	81	243	729	2187	
4	16	64	256	1024	4096	16384	6
5	25	125	625	3125	15625	78125	39
6	36	216	1296	7776	46656	279936	167
7	49	343	2401	16807	117649	823543	576
8	64	512	4096	32768	262144	2097152	1677
9	81	729	6561	59049	531441	4782969	4304
10	100	1000	10000	100000	1000000	10000000	10000

Einführung in die Informatik: Programmierung und Softwareentwicklung

Computersimulation

Auf einem Papier befinden sich parallele Linien im Abstand 2cm.

Eine Nadel der Länge 1cm wird zufällig auf das Papier geworfen.

Wie wahrscheinlich ist es, dass eine Linie getroffen wird?

Das untere Ende der Nadel liege auf Höhe $0 \leq y_{\text{low}} \leq 2$ (bezogen auf die Linie unmittelbar unterhalb der Nadel)

Der Winkel der Nadel betrage $0 \leq \alpha \leq 180$.

Beide Größen (y_{low} und α) seien gleichverteilt.

Das obere Ende der Nadel liegt dann auf Höhe $y_{\text{high}} = y_{\text{low}} + \sin(\alpha)$.

Ein Treffer liegt vor, wenn $y_{\text{high}} \geq 2$.

Näherungsweise Bestimmung der Trefferrate durch Simulation:

Einführung in die Informatik: Programmierung und Softwareentwicklung

Zufallsgenerator

Die Klasse `Random` stellt einen Zufallsgenerator bereit. Die Methode `nextDouble()` liefert einen “zufälligen” `Double`-Wert im Bereich $[0, 1]$

```
import java.util.Random;
public class RandomTest{
    public static void main(String[] args){
        Random gen = new Random();
        System.out.println(""+ gen.nextDouble() + " "
                           + gen.nextDouble());
    }
}
```

Druckt zwei “Zufallszahlen” aus, z.B.:

0.3119991282517587 0.2614453715060384

Der Aufruf `gen.nextInt( $n$ )` liefert einen “zufälligen” Integer im Bereich $0 \dots n - 1$.

Einführung in die Informatik: Programmierung und Softwareentwicklung

Ergebnis

...dauert ein paar Minuten und ist

Tries / hits:3.1414311574995777

Einführung in die Informatik: Programmierung und Softwareentwicklung

Buffon - Simulation

```
import java.util.Random;
public class Buffon
{ public static void main(String[] args){
    Random generator = new Random();
    int hits = 0;
    final int NTRIES = 100000000;
    for (int i = 1; i <= NTRIES; i++) {
        double ylow = 2 * generator.nextDouble();
        double angle = 180. * generator.nextDouble();
        double yhigh= ylow + Math.sin(Math.toRadians(angle));
        if (yhigh >= 2) hits++;
    }
    System.out.println("Tries / hits:" + (NTRIES * 1.0) / hits);
}}
```

Einführung in die Informatik: Programmierung und Softwareentwicklung

Analytische Lösung

$$\begin{aligned} & \Pr(y_{\text{low}} + \sin(\alpha) \geq 2) \\ &= \frac{1}{2} \int_{y=1}^2 \Pr(\sin(\alpha) \geq 2 - y) \, dy \\ &= \frac{1}{2} \int_{y=1}^2 1 - \Pr(\sin(\alpha) \leq 2 - y) \, dy \\ &= \frac{1}{2} \int_{y=1}^2 1 - 2 \cdot \Pr(\alpha \leq \arcsin(2 - y)) \, dy \\ &= \frac{1}{2} \int_{y=1}^2 1 - 2 \cdot \arcsin(2 - y) / \pi \, dy \end{aligned}$$

Einführung in die Informatik: Programmierung und Softwareentwicklung

Ausrechnen des Integrals

```
mhofmann@linux:~/work/teaching/EinfInfo> maple
|~|      Maple 7 (IBM INTEL LINUX)
_|_|    |/_|. Copyright (c) 2001 by Waterloo Maple Inc.
\  MAPLE / All rights reserved. Maple is a registered trademark o
<____ ____> Waterloo Maple Inc.
|          Type ? for help.
> 1/2 * int(1 - 2 * arcsin(2 - t) / Pi, t=1..2);
      1
     ----
      Pi
```

Einführung in die Informatik: Programmierung und Softwareentwicklung

Test

Was wird hier gedruckt?

```
for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 10; j++)
        System.out.print(i * j % 10);
    System.out.println();}
```

Wie oft werden diese Schleifen ausgeführt?

```
for (i = 1; i <= 10; i++) ...
for (i = -10; i <= 10; i++) ...
for (i= -10; i <= 10; i = i + 3) ...
for (i= -10; i <= 10; i = i - 1) ...
```

Welches sind die Tokens von `Hello, cruel world!` ?

Wie kann man eine Eingabe der Form *NameDerBrücke*
LängeDerBrücke lesen, wenn *NameDerBrücke* ein oder mehrere
Wörter sind und *LängeDerBrücke* ein Double ?

Einführung in die Informatik: Programmierung und Softwareentwicklung

Zusammenfassung: Schleifen

- Schleifen dienen zur wiederholten Ausführung von Statements.
- Es gibt die **while**-Schleife und die **for**-Schleife. Die **for**-Schleife wird benutzt, wenn im Verlauf der Schleife ein numerischer Wert in konstanten Schritten herauf- oder heruntergezählt wird.
- Beim Lesen von Eingaben kann man das Ende der Eingabe entweder durch einen Sonderwert (z.B. -1) oder durch Dateiende feststellen.
- Die Konsolenein- und -ausgabe kann auf Dateien umgelenkt werden.
- Mit Zufallszahlen innerhalb einer Schleife lassen sich Zufallsexperimente simulieren.

Einführung in die Informatik: Programmierung und Softwareentwicklung

Übungsaufgabe: Random Walk

vgl. Horstmann, P6.4, S. 263.

Man simuliere die Wanderung eines “Betrunkenen” in einem “Straßengitter”.

Man zeichne ein Gitter von 10×10 Straßen (=Linien) und repräsentiere den “Betrunkenen” als Kreuz in der Mitte des Gitters.

Eine bestimmte Zahl von Malen, z.B. 100, lasse man den Betrunkenen zufällig eine Richtung (N, O, S, W), bewege ihn einen Block weiter in dieser Richtung und zeichne ihn neu.

Anschließend bestimme man die zurückgelegte Entfernung.

Einführung in die Informatik: Programmierung und Softwareentwicklung