

Lektion 7

mathematische Methoden (Kosinus)

Switch-Statement

Arrays

Minsort

Methode zur Berechnung des Cosinus

- Der Cosinus lässt sich durch folgende Reihe berechnen:

$$\cos: \mathbb{R} \rightarrow [-1,1]$$

$$\cos(x) := \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k}}{(2k)!}$$

- Welche **Parameter** und welchen **Rückgabewert** hat die Methode?

```
public static double cos(double x)
```

- Wie sehen die ersten fünf Glieder aus?

Methode zur Berechnung des Cosinus

- Die ersten 5 Reihenglieder sehen wie folgt aus:

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} \mp \dots$$

- Wie kann ein Algorithmus (eine Rechenvorschrift) aussehen, die den Cosinus berechnet?
- Wir schauen zunächst, wie sich Zähler und Nenner in jedem Schritt verändern.

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} \pm \dots$$

Methode zur Berechnung des Cosinus



$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} \pm \dots$$

Diagram illustrating the factorial terms in the denominator of the cosine series expansion:

- $2! = 1 \cdot 2$
- $4! = 3 \cdot 4$
- $6! = 5 \cdot 6$
- $8! = 7 \cdot 8$

Diagram illustrating the powers of x in the numerator of the cosine series expansion:

- $x^2 = x \cdot x$
- $x^4 = x \cdot x \cdot x \cdot x$
- $x^6 = x \cdot x \cdot x \cdot x \cdot x \cdot x$
- $x^8 = x \cdot x \cdot x \cdot x \cdot x \cdot x \cdot x \cdot x$

```
public static double cos(double x)
{
```

```
    double zaehler = 1.0;
```

```
    double nenner = 1.0;
```

```
    double summe = 1.0;
```

Wir belegen den
ersten Summanden vor!

Methode zur Berechnung des Cosinus



$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} \pm \dots$$

Diagram illustrating the factorial terms in the denominator of the cosine series expansion:

- $2! = 1 \cdot 2$
- $4! = 3 \cdot 4$
- $6! = 5 \cdot 6$
- $8! = 7 \cdot 8$

Each term is also associated with a pair of $x \cdot x$ above the fraction line.

Wir belegen den
ersten Summanden vor!

Starten die Schleife
beim 2. Summanden

```
public static double cos(double x)
{
    double zaehler = 1.0;
    double nenner = 1.0;
    double summe = 1.0;
```

```
    for (int i = 2; ; )
    {
```

Methode zur Berechnung des Cosinus



$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} \pm \dots$$

The diagram illustrates the iterative calculation of the cosine series. It shows the first five terms of the series: 1, $-\frac{x^2}{2!}$, $+\frac{x^4}{4!}$, $-\frac{x^6}{6!}$, and $+\frac{x^8}{8!}$. Above each term, a bracket indicates the multiplication of x by itself (e.g., $x \cdot x$ for x^2). Below each factorial denominator, a bracket indicates the multiplication of consecutive integers (e.g., $1 \cdot 2$ for $2!$, $3 \cdot 4$ for $4!$, etc.).

Der Zähler berechnet sich, indem man den alten Zähler mit x^2 multipliziert (und das Vorzeichen ändert).

```
public static double cos(double x)
{
    double zaehler = 1.0;
    double nenner = 1.0;
    double summe = 1.0;

    for (int i = 2; ; )
    {
        zaehler = zaehler * x * x * (-1);
```

Methode zur Berechnung des Cosinus



$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} \pm \dots$$

Diagram illustrating the factorial calculation for the denominator of the cosine series. The terms are grouped by arcs above and below the factorial symbols:

- $2!$ is calculated as $1 \cdot 2$ (arc above: $x \cdot x$)
- $4!$ is calculated as $3 \cdot 4$ (arc above: $x \cdot x$)
- $6!$ is calculated as $5 \cdot 6$ (arc above: $x \cdot x$)
- $8!$ is calculated as $7 \cdot 8$ (arc above: $x \cdot x$)

Der Nenner berechnet sich, indem man den alten Nenner mit dem Laufindex i und $i-1$ multipliziert.

```
public static double cos(double x)
{
    double zaehler = 1.0;
    double nenner = 1.0;
    double summe = 1.0;

    for (int i = 2; ; )
    {
        zaehler = zaehler * x * x * (-1);
        nenner = nenner * i * (i-1);
```

Methode zur Berechnung des Cosinus



$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} \pm \dots$$

Diagram illustrating the factorial calculation for the denominator of the cosine series terms:

- For $2!$: $1 \cdot 2$
- For $4!$: $3 \cdot 4$
- For $6!$: $5 \cdot 6$
- For $8!$: $7 \cdot 8$

Each term's numerator is shown as $x \cdot x$ repeated for the power of x .

Aufaddieren!

```
public static double cos(double x)
{
    double zaehler = 1.0;
    double nenner = 1.0;
    double summe = 1.0;
    double summand;
    for (int i = 2; ; )
    {
        zaehler = zaehler * x * x * (-1);
        nenner = nenner * i * (i-1);
        summand = zaehler/nenner;
        summe = summe + summand;
    }
}
```


Methode zur Berechnung des Cosinus



$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} \pm \dots$$

Diagram illustrating the factorial denominators in the cosine series expansion. Arcs connect the terms to their respective factorial denominators, which are shown as products of consecutive integers:

- $2! = 1 \cdot 2$
- $4! = 3 \cdot 4$
- $6! = 5 \cdot 6$
- $8! = 7 \cdot 8$

Um wie viel muss i erhöht werden?

```
public static double cos(double x)
{
    double zaehler = 1.0;
    double nenner = 1.0;
    double summe = 1.0;
    double summand;
    for (int i = 2;          ; i=i+2)
    {
        zaehler = zaehler * x * x * (-1);
        nenner = nenner * i * (i-1);
        summand = zaehler/nenner;
        summe = summe + summand;
    }
}
```

Methode zur Berechnung des Cosinus



$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} \pm \dots$$

Diagram illustrating the factorial terms in the cosine series expansion:

- For $2!$: $1 \cdot 2$
- For $4!$: $3 \cdot 4$
- For $6!$: $5 \cdot 6$
- For $8!$: $7 \cdot 8$

Each factorial term is also shown as a product of two $x \cdot x$ terms, with arcs indicating the grouping of the x terms.

Wie wählen wir die
Laufbedingung?

```
public static double cos(double x)
{
    double zaehler = 1.0;
    double nenner = 1.0;
    double summe = 1.0;
    double summand;
    for (int i = 2; ; i=i+2)
    {
        zaehler = zaehler * x * x * (-1);
        nenner = nenner * i * (i-1);
        summand = zaehler/nenner;
        summe = summe + summand;
    }
}
```

Methode zur Berechnung des Cosinus



$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} \pm \dots$$

Diagram illustrating the factorial denominators in the cosine series expansion:

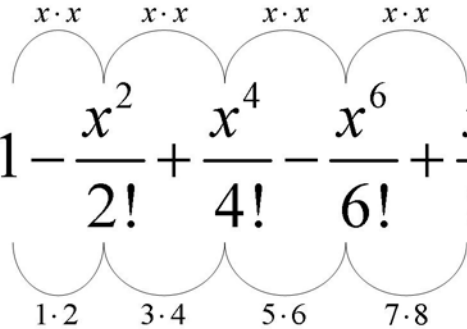
- For $2!$: $1 \cdot 2$
- For $4!$: $3 \cdot 4$
- For $6!$: $5 \cdot 6$
- For $8!$: $7 \cdot 8$

The numerators are shown as $x \cdot x$ for each even power term.

Wie wählen wir die
Laufbedingung?

```
public static double cos(double x)
{
    double zaehler = 1.0;
    double nenner = 1.0;
    double summe = 1.0;
    double summand = 1;
    for (int i = 2; summand > 1E-15 ||
                summand < -1E-15; i=i+2)
    {
        zaehler = zaehler * x * x * (-1);
        nenner = nenner * i * (i-1);
        summand = zaehler/nenner;
        summe = summe + summand;
    }
}
```

Methode zur Berechnung des Cosinus

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} \pm \dots$$


Rückgabewert ergänzen!

```
public static double cos(double x)
{
    double zaehler = 1.0;
    double nenner = 1.0;
    double summe = 1.0;
    double summand = 1;
    for (int i = 2; summand > 1E-15 ||
        summand < -1E-15; i=i+2)
    {
        zaehler = zaehler * x * x * (-1);
        nenner = nenner * i * (i-1);
        summand = zaehler/nenner;
        summe = summe + summand;
    }
    return summe;
}
```

Methodendokumentation

```
/**
 * Berechnet den Kosinus von der übergebenen Zahl x.
 * @param x reelle Zahl (in RAD), von der der Kosinus berechnet werden soll.
 * @return Ergebnis des Kosinus; reelle Zahl zwischen -1 und 1 (beide inklusive)
 */
public static double cos(double x)
{
    double zaehler = 1.0;
    double nenner = 1.0;
    double summe = 1.0;
    double summand = 1;
    for (int i = 2; summand > 1E-15 || summand < -1E-15; i = i + 2)
    {
        zaehler = zaehler * x * x * (-1);
        nenner = nenner * i * (i - 1);
        summand = zaehler / nenner;
        summe = summe + summand;
    }
    return summe;
}
```

Wenn andere die Methode
verwenden sollen, sollten
wir sie entsprechend
dokumentieren!

Anderen Methoden zur Verfügung stellen

```
package de.fhws;  
public class MyMath {  
    /**  
     * Berechnet den Kosinus von der übergebenen Zahl x.  
     * @param x reelle Zahl (in RAD), von der der Kosinus berechnet werden soll.  
     * @return Ergebnis des Kosinus; reelle Zahl zwischen -1 und 1 (beide inklusive)  
     */  
    public static double cos(double x) {  
        double zaehler = 1.0;  
        double nenner = 1.0;  
        double summe = 1.0;  
        double summand = 1;  
        for (int i = 2; summand > 1E-15 || summand < -1E-15; i = i + 2) {  
            zaehler = zaehler * x * x * (-1);  
            nenner = nenner * i * (i - 1);  
            summand = zaehler / nenner;  
            summe = summe + summand;  
        }  
        return summe;  
    }  
}
```

Methoden einbinden

```
package pkg06;
```

```
import de.fhws.MyMath;
```

```
public class KosinusTest
```

```
{
```

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

```
    {
```

```
        System.out.println(MyMath.cos(0.5));
```

```
    }
```

```
}
```

double de.fhws.MyMath.cos(double x)
Berechnet den Kosinus von der übergebenen Zahl x.

Parameters:
x reelle Zahl (in RAD), von der der Kosinus gezogen werden soll.

Returns:
Ergebnis des Kosinus; reelle Zahl zwischen -1 und 1 (beide inklusive)

Press 'F2' for focus

Um die Methode aufrufen zu können,
muss die Klasse MyMath auf dem
Klassenpfad liegen.

Mathematische Funktionen

- Die Klasse `java.lang.Math` stellt eine Menge an geläufigen mathematischen Konstanten und Funktionen zur Verfügung:

Mathematische Funktion oder Konstante	Methode oder Konstante in Java
e	double Math.E
π	double Math.PI
$ x $	double Math.abs (double x)
$\cos(x)$	double Math.cos (double x)
$\sin(x)$	double Math.sin (double x)
\sqrt{x}	double Math.sqrt (double x)
x^y	double Math.pow (double x, double y)
gibt Zufallszahl x mit $x \in [0,1)$ zurück	double Math.random ()

switch-Statement

- Ein switch-Statement ist eine Kontrollstruktur, die i.d.R. alternativ zu einer if-Anweisung mit mehreren else-ifs eingesetzt wird.
- Die switch-Anweisung nimmt
 - eine ganze Zahl (byte, char, short, int, aber **kein** long),
 - einen String
 - oder ein Enum (später dazu mehr)entgegen.


```
int zahl = (int) (Math.random() * 6 + 1);  
switch (zahl) {  
  
  
  
  
  
  
  
  
  
}
```



switch-Statement

- Die switch-Anweisung beinhaltet verschiedene **case**-Labels.

```
int zahl = (int) (Math.random() * 6 + 1);
switch (zahl) {
    case 1:
        System.out.println("Es wurde eine 1 gewürfelt.");
        break;
    case 3:
        System.out.println("Es wurde eine 3 gewürfelt.");
    case 4:
        System.out.println("Es wurde eine 3 oder 4 gewürfelt.");
        break;
}
```



Es wird der case betreten, der mit `zahl` übereinstimmt.
Von dort werden **alle** Anweisungen bis zum nächsten
`break` oder dem Ende des `switch`-Blocks ausgeführt.

Im case muss ein konstanter
Ausdruck vom Typ der Variablen
im switch stehen

switch-Statement

- Die switch-Anweisung beinhaltet verschiedene **case**-Labels.

```
int zahl = (int) (Math.random() * 6 + 1);
switch (zahl) {
    case 1:
        System.out.println("Es wurde eine 1 gewürfelt.");
        break;
    case 3:
    case 4:
        System.out.println("Es wurde eine 3 oder 4 gewürfelt.");
        break;
    default:
        System.out.println("Es wurde keine 1, 3 oder 4 gewürfelt.");
}
```

- Trifft kein case zu, wird die **default**-Anweisung ausgeführt.

```
public static int tageImMonat(int monat) {  
    int tage;  
    switch(monat) {  
        case 1:  
        case 3:  
        case 5:  
        case 7:  
        case 8:  
        case 10:  
        case 12:  
            tage = 31;  
            break;  
        case 2:  
            tage = 28;  
            break;  
        case 4:  
        case 6:  
        case 9:  
        case 11:  
            tage = 30;  
            break;  
    }  
    return tage;  
}
```



Methode zur Zuordnung der Anzahl Tage in einem Monat

The local variable **tage** may not have
been initialized

```
public static int tageImMonat(int monat) {  
    int tage = -1;  
    switch(monat) {  
        case 1:  
        case 3:  
        case 5:  
        case 7:  
        case 8:  
        case 10:  
        case 12:  
            tage = 31;  
            break;  
        case 2:  
            tage = 28;  
            break;  
        case 4:  
        case 6:  
        case 9:  
        case 11:  
            tage = 30;  
            break;  
    }  
    return tage;  
}
```

Methode zur Zuordnung der Anzahl Tage in einem Monat

```
public static int tageImMonat(int monat) {  
    int tage;  
    switch(monat) {  
        case 1:  
        case 3:  
        case 5:  
        case 7:  
        case 8:  
        case 10:  
        case 12:  
            tage = 31;  
            break;  
        case 2:  
            tage = 28;  
            break;  
        case 4:  
        case 6:  
        case 9:  
        case 11:  
            tage = 30;  
            break;  
        default:  
            tage = -1;  
    }  
    return tage;  
}
```

Methode zur Zuordnung der Anzahl Tage in einem Monat

```
public static int tageImMonat(int monat) {  
    int tage;  
  
    switch(monat) {  
        case 2:  
            tage = 28;  
            break;  
        case 4:  
        case 6:  
        case 9:  
        case 11:  
            tage = 30;  
            break;  
        default:  
            tage = 31;  
    }  
    return tage;  
}
```

Methode zur Zuordnung der Anzahl Tage in einem Monat

Was passiert bei `monat < 1` oder `monat > 12` ?

```
public static int tageImMonat(int monat) {  
    int tage;  
    if (monat > 12 || monat < 1) return -1;  
    switch(monat) {  
        case 2:  
            tage = 28;  
            break;  
        case 4:  
        case 6:  
        case 9:  
        case 11:  
            tage = 30;  
            break;  
        default:  
            tage = 31;  
    }  
    return tage;  
}
```

Methode zur Zuordnung der Anzahl Tage in einem Monat

Was passiert bei `monat < 1` oder `monat > 12` ?


```
public static int tageImMonat(String monat)
{
    int tage;
    switch(monat)
    {
        case "Februar":
            tage = 28;
            break;
        case "April":
        case "Juni":
        case "September":
        case "November":
            tage = 30;
            break;
        default:
            tage = 31;
    }
    return tage;
}
```

Methode zur Zuordnung der Anzahl Tage in einem Monat

Nehmen wir an, wir wollen 10000
Würfelwürfe simulieren!

```
int einser = 0, zweier = 0, dreier = 0,  
vierer = 0, fuenfer = 0, sechser = 0;
```

```
for (int i = 0; i < 10000; i++) {  
    int wurf = (int)(Math.random()*6+1);  
    switch(wurf) {  
        case 1:  
            einser++; break;  
        case 2:  
            zweier++; break;  
        case 3:  
            dreier++; break;  
        case 4:  
            vierer++; break;  
        case 5:  
            fuenfer++; break;  
        case 6:  
            sechser++; break;  
    }  
}  
System.out.println("1er: " + einser);  
System.out.println("2er: " + zweier);  
System.out.println("3er: " + dreier);
```

...

```
int einser = 0, zweier = 0, dreier = 0,  
vierer = 0, fuenfer = 0, sechser = 0;
```

```
for (int i = 0; i < 10000; i++) {  
    int wurf = (int)(Math.random()*6+1);  
    switch(wurf) {
```

```
        case 1:  
            einser++; break;
```

```
        case 2:  
            zweier++; break;
```

```
        case 3:  
            dreier++; break;
```

```
        case 4:  
            vierer++; break;
```

```
        case 5:  
            fuenfer++; break;
```

```
        case 6:  
            sechser++; break;
```

```
    }  
}  
System.out.println("1er: " + einser);  
System.out.println("2er: " + zweier);  
System.out.println("3er: " + dreier);  
...
```

gleichartige Informationen:
Zähle bei einer 1 die Einser hoch
Zähle bei einer 2 die Zweier hoch
etc.

```
int einser = 0, zweier = 0, dreier = 0,  
vierer = 0, fuenfer = 0, sechser = 0;
```

```
for (int i = 0; i < 10000; i++) {  
    int wurf = (int)(Math.random()*6+1);  
    switch(wurf) {
```

```
        case 1:  
            einser++; break;
```

```
        case 2:  
            zweier++; break;
```

```
        case 3:  
            dreier++; break;
```

```
        case 4:  
            vierer++; break;
```

```
        case 5:  
            fuenfer++; break;
```

```
        case 6:  
            sechser++; break;
```

```
    }  
}  
System.out.println("1er: " + einser);  
System.out.println("2er: " + zweier);  
System.out.println("3er: " + dreier);  
...
```

gleichartige Informationen:
Zähle bei einer 1 die Einser hoch
Zähle bei einer 2 die Zweier hoch
etc.

Kann man die verschiedenen
Variablen kurz auf gleiche Weise
behandeln?

Arrays

- Arrays (auch Felder genannt) werden verwendet, um mehrere gleichartige Datentypen zu gruppieren.

```
/*Deklariere eine Variable, die eine Referenz auf ein Feld für Integervariablen speichert*/
```

```
int[] zahlen;
```

```
/*Lege Speicher für ein Feld von 6 Integervariablen an
```

```
Der new-Operator gibt eine Referenz auf den angelegten Speicherbereich zurück. */  
zahlen = new int[6];
```

```
/* Beides in einer Zeile:
```

```
Deklariere (ein Feld mit) 6 Integervariablen*/  
int[] zahlen = new int[6];
```

Arrays

- Nehmen wir an, wir wollen die Lottozahlen speichern:

```
/*Deklariere (ein Feld mit) 6 Integervariablen*/  
int[] zahlen = new int[6];
```

```
//In den folgenden Zeilen werden diese mit Werten belegt:  
zahlen[0] = 5;  
zahlen[1] = 7;  
zahlen[2] = 23;  
zahlen[3] = 34;  
zahlen[4] = 39;  
zahlen[5] = 41;
```

Arrays

```
int[] zahlen = new int[6];
```

//In den folgenden Zeilen werden diese mit Werten belegt

```
zahlen[0] = 5;
```

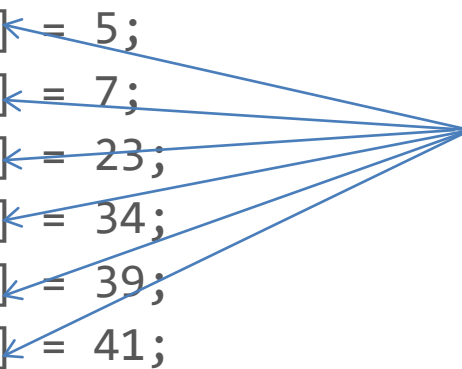
```
zahlen[1] = 7;
```

```
zahlen[2] = 23;
```

```
zahlen[3] = 34;
```

```
zahlen[4] = 39;
```

```
zahlen[5] = 41;
```



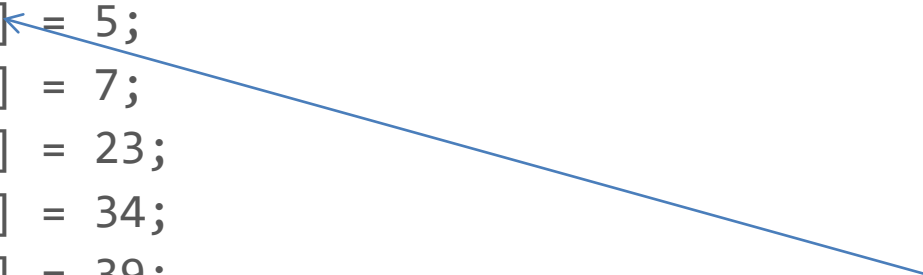
Der Index in den eckigen Klammern gibt an, auf welche Variable des Arrays zugegriffen wird.

Arrays

```
int[] zahlen = new int[6];
```

//In den folgenden Zeilen werden diese mit Werten belegt

```
zahlen[0] = 5;  
zahlen[1] = 7;  
zahlen[2] = 23;  
zahlen[3] = 34;  
zahlen[4] = 39;  
zahlen[5] = 41;
```



Das erste Element des Arrays ist über den Index „0“ zugreifbar,
das zweite Element über den Index „1“, etc.

Initialisierung und Länge von Arrays

- Eine direkte Initialisierung des Arrays kann in der Deklarationszeile erfolgen:

```
int[] zahlen = new int[]{5, 7, 23, 34, 39, 41};
```

oder kürzer

```
int[] zahlen = {5, 7, 23, 34, 39, 41};
```

- Von einem Array kann man über dessen Attribut ***Length*** die Anzahl der Elemente im Array abfragen.
- `zahlen.Length` ergibt in obigem Bsp. den Wert 6.

```
int einser = 0, zweier = 0, dreier = 0,  
    vierer = 0, fuenfer = 0, sechser = 0;
```

```
for (int i = 0; i < 10000; i++) {  
    int wurf = (int)(Math.random()*6+1);  
    switch(wurf) {  
        case 1:  
            einser++; break;  
        case 2:  
            zweier++; break;  
        case 3:  
            dreier++; break;  
        case 4:  
            vierer++; break;  
        case 5:  
            fuenfer++; break;  
        case 6:  
            sechser++; break;  
    }  
}  
System.out.println("1er: " + einser);  
System.out.println("2er: " + zweier);  
System.out.println("3er: " + dreier);
```

...

```
int[] gewuerfelteSeiten = new int[6];

for (int i = 0; i < 10000; i++) {
    int wurf = (int)(Math.random()*6+1);
    switch(wurf) {
        case 1:
            einser++; break;
        case 2:
            zweier++; break;
        case 3:
            dreier++; break;
        case 4:
            vierer++; break;
        case 5:
            fuenfer++; break;
        case 6:
            sechser++; break;
    }
}
System.out.println("1er: " + einser);
System.out.println("2er: " + zweier);
System.out.println("3er: " + dreier);
...
```

```
int[] gewuerfelteSeiten = new int[6];

for (int i = 0; i < 10000; i++) {
    int wurf = (int)(Math.random()*6+1);
    switch(wurf) {
        case 1:
            gewuerfelteSeiten[0]++; break;
        case 2:
            gewuerfelteSeiten[1]++; break;
        case 3:
            gewuerfelteSeiten[2]++; break;
        case 4:
            gewuerfelteSeiten[3]++; break;
        case 5:
            gewuerfelteSeiten[4]++; break;
        case 6:
            gewuerfelteSeiten[5]++; break;
    }
}
System.out.println("1er: " + einser);
System.out.println("2er: " + zweier);
System.out.println("3er: " + dreier);
...
```

```
int[] gewuerfelteSeiten = new int[6];

for (int i = 0; i < 10000; i++) {
    int wurf = (int)(Math.random()*6+1);
    switch(wurf) {
        case 1:
            gewuerfelteSeiten[0]++; break;
        case 2:
            gewuerfelteSeiten[1]++; break;
        case 3:
            gewuerfelteSeiten[2]++; break;
        case 4:
            gewuerfelteSeiten[3]++; break;
        case 5:
            gewuerfelteSeiten[4]++; break;
        case 6:
            gewuerfelteSeiten[5]++; break;
    }
}

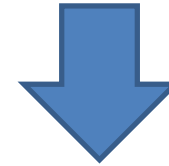
for (int i = 0; i < gewuerfelteSeiten.length; i++)
{
    System.out.println(i+1+"er: " + gewuerfelteSeiten[i]);
}
```

```
int[] gewuerfelteSeiten = new int[6];
```

```
for (int i = 0; i < 10000; i++) {  
    int wurf = (int)(Math.random()*6+1);  
    switch(wurf) {  
        case 1:  
            gewuerfelteSeiten[wurf-1]++; break;  
        case 2:  
            gewuerfelteSeiten[wurf-1]++; break;  
        case 3:  
            gewuerfelteSeiten[wurf-1]++; break;  
        case 4:  
            gewuerfelteSeiten[wurf-1]++; break;  
        case 5:  
            gewuerfelteSeiten[wurf-1]++; break;  
        case 6:  
            gewuerfelteSeiten[wurf-1]++; break;  
    }  
}
```

```
for (int i = 0; i < gewuerfelteSeiten.length; i++)  
{  
    System.out.println(i+1+"er: " + gewuerfelteSeiten[i]);  
}
```

Die Anweisung in jedem
Case ist jetzt gleich.



Switch-Case wird jetzt
nicht mehr benötigt.

```
int[] gewuerfelteSeiten = new int[6];
```

```
for (int i = 0; i < 10000; i++) {  
    int wurf = (int)(Math.random()*6+1);
```

```
    gewuerfelteSeiten[wurf-1]++;
```

```
}  
for (int i = 0; i < gewuerfelteSeiten.length; i++)  
{  
    System.out.println(i+1+"er: " + gewuerfelteSeiten[i]);  
}
```


Wir wollen ein Feld von Zahlen sortieren, so
dass die kleinste zuerst kommt.

Sortieren



68	22	56	34	12	38	9	17
----	----	----	----	----	----	---	----

- Der Inhalt der ersten Feld-Komponente wird als Minimum angesehen (68)!
- In den rechts davon liegenden Feldkomponenten wird ein neues Minimum gesucht:
 - $68 > 22 \rightarrow 22$ neues Minimum
 - $22 < 56 \rightarrow$ kein neues Minimum
 - $22 < 34 \rightarrow$ kein neues Minimum
 - $22 > 12 \rightarrow 12$ neues Minimum
 - $12 < 38 \rightarrow$ kein neues Minimum
 - $12 > 9 \rightarrow 9$ neues Minimum
 - $9 < 17 \rightarrow$ kein neues Minimum

68 wird mit 9 getauscht !

9	22	56	34	12	38	68	17
---	----	----	----	----	----	----	----

Sortieren



9	22	56	34	12	38	68	17
---	----	----	----	----	----	----	----

- Der Inhalt der zweiten Feld-Komponente wird als Minimum angesehen (22)!
- In den rechts davon liegenden Feldkomponenten wird ein neues Minimum gesucht:
 - $22 < 56 \rightarrow$ kein neues Minimum
 - $22 < 34 \rightarrow$ kein neues Minimum
 - $22 > 12 \rightarrow$ 12 neues Minimum
 - $12 < 38 \rightarrow$ kein neues Minimum
 - $12 < 68 \rightarrow$ kein neues Minimum
 - $12 < 17 \rightarrow$ kein neues Minimum

22 wird mit 12 getauscht !

9	12	56	34	22	38	68	17
---	----	----	----	----	----	----	----

Sortieren



9	12	56	34	22	38	68	17
---	----	----	----	----	----	----	----

- Der Inhalt der dritten Feld-Komponente wird als Minimum angesehen (56)!
- In den rechts davon liegenden Feldkomponenten wird ein neues Minimum gesucht:

- $56 > 34 \rightarrow 34$ neues Minimum
- $34 > 22 \rightarrow 22$ neues Minimum
- $22 < 38 \rightarrow$ kein neues Minimum
- $22 < 68 \rightarrow$ kein neues Minimum
- $22 > 17 \rightarrow 17$ neues Minimum

56 wird mit 17 getauscht !

9	12	17	34	22	38	68	56
---	----	----	----	----	----	----	----

usw.

Was ist der Wert von routeNumber nach dem switch-Block?

```
String zipCode = "93705";  
int routeNumber;  
switch (zipCode)  
{  
    case "93705":  
    case "93706":  
        routeNumber = 1;  
        break;  
    case "93710":  
    case "93720":  
        routeNumber = 2;  
        break;  
    default:  
        routeNumber = 0;  
        break;  
}
```

A: 0

B: 1

C: 2

D: nicht definiert



source: <http://www.finchrobot.com/sites/default/files/BirdBrain1010-0005.jpg>

- Lichtsensor
- Temperatursensor
- Sensor für Hindernisse
- Beschleunigungsmesser
- Motoren
- Buzzer
- RGB “Schnabel”-LED
- Stiftbefestigung (zum Malen)
- Strom über USB Kabel

Finch



source: <http://www.finchrobot.com/sites/default/files/finchie.jpg>

Finch einrichten

- von <http://www.finchrobot.com/downloads> Eclipse package herunterladen!
- FinchJava.zip entpacken
- Ordner FinchSoftware muss nachher zu sehen sein.
- In Eclipse:
 - Neues Projekt anlegen
 - Finch Software\SourceFiles\Code nach src kopieren
 - Finch Software\javadoc in das Projekthauptverzeichnis kopieren
 - civil.dll, finch.jar kopieren in das Projekthauptverzeichnis
 - finch.jar dem Klassenpfad hinzufügen (Rechtsklick auf finch.jar -> Build Path -> Add to Build Path)

Javadoc einrichten

- Um die Hilfe zu den Methoden zu sehen,

```
myFinch.setLED(255, 255, 0);  
myFinch.  
// Set L  
myFinch.  
myFinch.  
// Set L
```

● void edu.cmu.ri.createlab.terk.robot.finch.Finch.setLED(int red, int green, int blue)

setLED

```
public void setLED(int red,  
                  int green,  
                  int blue)
```

Sets the color of the LED in the Finch's beak. The LED can be any color that can be created by mixing red, green, and blue; turning on all three colors in equal amounts results in white light. Valid ranges for the red, green, and blue elements are 0 to 255.

@ Javadoc Declaration Search Console

Doclet (Java Application) C:\Program Files\Java\jdk-7.0_71\bin\javac.exe (10/11/2014 12:44:53)

muss das Javadoc eingerichtet werden.

Javadoc einrichten

- Rechte Taste auf das Projekt -> Properties
- Java Build Path -> Libraries -> finch.jar aufklappen
- *Javadoc location* anklicken und Edit auswählen
- Unter JavaDoc URL den Pfad des javadoc Ordners auswählen:
file:/<absoluter Pfad zum Eclipse Workspace>/<projektname>/javadoc/
- z.B.
 - file:/C:/W/WorkspaceProgrammierenI/Finch2/javadoc/

Finch testen

- Finch per USB an den Rechner anschließen
- Dance.java in Eclipse starten