

# Methoden

Javakurs 2012, 3. Vorlesung

Magdalena Rätz  
maggyrz@freitagsrunde.org

5. März 2013

- 1 Wiederholung
- 2 Wozu Methoden?
- 3 Methoden Schreiben
  - Aufbau einer Methode
  - Kopf
  - Rumpf
    - Variablengültigkeit
  - Überladen Methoden
  - Verschachteln
- 4 Methoden Benutzen
  - Aufruf aus der Klasse
  - Aufruf in einer anderen Klasse
- 5 Rekursion
- 6 ?!

## Tag 1 in Mega-Kurzfassung

- Datentypen: z.B. `int`, `boolean`, `String`
- Variablen deklarieren und initialisieren:  
z.B. `int a = 1;`
- Fallunterscheidungen: `if (Bedingung){} else {}`
- Schleifen: `for (int i = 0; i < ende; i+2){}` und  
`while (Bedingung){}`
- Arrays: `char[] wort = {'w', 'o', 'r', 't'};`

## Wozu brauchen wir Methoden?

- Wiederkehrende Aufgaben  
z.B. eine Zufallszahl berechnen `Math.random()`;
- Strukturierung des Quellcodes

```
1 class Steuerung{
2     public static void main(String[] args) {
3         // ...
4         systemOK = checkSystem();
5         starteMotor(geschwindigkeit);
6         // ...
7     }
8 }
```

## Aufbau einer Methode

Methoden bestehen aus

- Kopf
- Rumpf

Der grundsätzliche Aufbau sieht dann so aus: <Kopf> {<Rumpf>}

## Wohin kommen Methoden?

```
1 class MeineKlasseMitMethoden{
2     // Methoden koennen hierhin
3     public static void main(String[] args) {
4         // ...
5     }
6     // oder und hierhin, die Reihenfolge ist egal
7 }
```

## Methodenkopf

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

- **public**: Festlegung der Sichtbarkeit
- **static**: unterscheidet zwischen Objektmethoden und nicht Objektmethoden
- **void**: Typ des Rückgabewerts, **void** wenn es keine Rückgabe gibt
- **main**: Name der Methode  
Konvention: beginnt mit einem Kleinbuchstaben und nutzt camelCase
- **(String[] args)**: Parameterliste  
kann auch leer sein **int** meineMethode()

## Parameterliste

für jeden Parameter muss angegeben werden:

- der Datentyp
- der lokale Variablenname

Parameter werden durch , getrennt

Beispiel: Berechnung des Flächeninhalts eines Rechtecks:

```
1 static int berechneFlaeche(int a, int b) {  
2     int flaeche = a*b;  
3     return flaeche;  
4 }
```

oder

```
1 static int berechneFlaeche(int a, int b) {  
2     return a*b;  
3 }
```



## Variablengültigkeit

- ① Welche Variablen kennt meine Methode?
  - "Klassenvariablen": in der Klasse definierte Variablen
  - die übergebenen Parameter
- ② Was wenn ich zusätzliche Variablen brauche?  
→ dann definiere ich welche in der Methode, die nur dort gültig sind: "lokale Variablen"

## Beispiel 1:

```
1 class Geometrie {
2     static int berechneFlaeche(int a, int b) {
3         return a*b;
4     }
5     public static void main(String[] args) {
6         int a = 3;
7         int b = 2;
8         int umfang = 2*a+2*b;
9         int c = berechneFlaeche(a,b);
10        System.out.println("Rechteck " + a + " ,
        " + b + ": Flaeche = " + c + " ,
        Umfang = " + umfang);
11    }
12 }
```

Was wird ausgegeben, wenn berechneFlaeche so definiert wird?

Fall 1:

```
1 static int berechneFlaeche(int a, int b) {  
2     int umfang = a+1;  
3     return a*b;  
4 }
```

Fall 2:

```
1 static int berechneFlaeche(int a, int b) {  
2     a = a+1;  
3     return a*b;  
4 }
```

Fall 3:

```
1 static int berechneFlaeche(int b) {  
2     a = a+1;  
3     return a*b;  
4 }
```

## Verschattung

Variablen in Parameterlisten und in der Methode lokal definierte Variablen können genauso heißen wie Klassenvariablen. Die lokalen Variablen *verschatten* dann in der Methode die Klassenvariable.

- kein Zugriff auf den Wert der Klassenvariable
- keine (versehentliche) Änderung des Werts der Klassenvariable
- Klassenvariablen können durch Methoden geändert werden, wenn sie nicht verschattet sind

## Überladen von Methoden

Jetzt neu: Geometrie auch mit Kreisen!

Wir wollen jetzt auch den Flächeninhalt eines Kreises berechnen.

```
1 static int berechneFlaeche(int radius, int  
    pi) {  
2     return pi*radius*radius;  
3 }
```

Die Variablennamen der Parameterliste können nicht zur Unterscheidung von Methoden verwendet werden.

## Überladen

Wenn es mehrere Methoden mit dem gleichen Namen in einer Klasse gibt, heißt das *überladen*.

## Eindeutigkeit von Methoden

Methoden gleichen Namens sind eindeutig definiert mit:

- unterschiedlicher Parameteranzahl
- unterschiedlichen Parametertypen

Auch ein anderer Rückgabetyt ist nicht ausreichend!

## Verschachteln

Kann man Methoden in anderen Methoden deklarieren?

→ Nein, aber man kann in Methoden Methoden aufrufen.

## Aufruf innerhalb der Klasse

- Parameter als Variablen ohne Datentypangabe oder direkte Übergabe von Werten
- ohne Rückgabewert

```
1 public static void main(String[] args) {  
2     int a = 3;  
3     meineMethode(a);  
4 }
```

- mit Rückgabewert

```
1 public static void main(String[] args) {  
2     int b = meineMethode(3);  
3 }
```



## Aufruf in einer anderen Klasse

ohne Rückgabewert:

```
1 public static void main(String[] args) {
2     int meineVar = 3;
3     MeineKlasse.meineMethode(meineVar);
4 }
```

mit Rückgabewert:

```
1 public static void main(String[] args) {
2     int meineVarA = 3;
3     int meineVarB =
        MeineKlasse.meineMethode(meineVarA);
4 }
```

Testen

Testen nicht vergessen!

## Rekursion

Rekursion nennt man die Definition einer Funktion durch sich selbst, z.B. bei der rekursiven Definition von Folgen

$$\begin{aligned} (a_n) &= 2 \cdot a_{n-1} \\ (a_0) &= 1 \end{aligned}$$

Diese Technik wird auch in der Informatik verwendet. Dabei ist die Abbruchbedingung bzw. die Abbruchbedingungen besonders wichtig.

Beispiel: Berechnung der Fakultät

Bildungsvorschrift:  $n! = n \cdot (n - 1)!$

Abbruchbedingung:  $0! = 1$

```
1 class Fakultaet{
2     static int fakultaet(int n) {
3         if (n == 0) {
4             return 1;
5         }
6         return n*fakultaet(n-1);
7     }
8 }
```



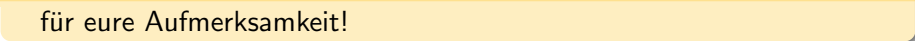
Fragen



???



Vielen Dank



für eure Aufmerksamkeit!