Unterprogramme / Methoden

Methoden oder allgemein Unterprogramme kann man mit Makros in einer Textverarbeitung oder Funktionen in der Mathematik vergleichen. Mit ihnen kann man größere Programme besser aufbauen:

1. Ein Unterprogramm kann an mehreren Stellen eines Programms aufgerufen werden (*Ökonomie Vorteil*).
2. Ein Programm wird durch Unterprogramme gegliedert und damit übersichtlicher und leichter verstehbar (*Gliederungs-Vorteil*).
3. Ein Unterprogramm kann als Baustein in mehreren Programmen eingebunden werden. Die Bausteine werden in einer „Programm-Bibliothek“ gesammelt (*Bausteine-Vorteil*).

Eigentlich gehören in Java Methoden zum objektorientierten Programmieren, was wir aber erst später behandeln werden. Deshalb arbeiten wir im Moment nur mit speziellen Methoden, sogenannten Klassenmethoden oder statischen Methoden.
Diese werden stets innerhalb einer Klasse definiert und haben folgenden Aufbau:

```
public static <Rückgabetyp> <Methodenname>(){
  // Code, der in der Methode ausgeführt wird
}
```

oder wenn eine Methode Parameter erwartet:

```
public static <Rückgabetyp> <Methodenname>(<Parameterliste>){
       // Code, der in der Methode ausgeführt wird
}
```

- Unter `<Rückgabetyp>` einer Methode versteht man den Datentyp, den die Methode zurückliefern soll. In der Mathematik liefert z. B. die Funktion sin(x) ein Ergebnis vom Typ double zurück. Wenn man eine Methode keinen Wert zurückliefern soll, schreibt man als Typ „`void`“.
- Der `<Methodenname>` ist ein Bezeichner (Regeln: siehe früher), unter dem die Methode von Java erkannt werden soll.
- Die `<Parameterliste>` ist eine Menge von Werten, die der Methode übergeben werden. Mit diesen Werten arbeitet die Methode dann. Die Deklaration eines Parameters entspricht der einer Variablen, abgesehen von Initialisierungswerten. Mehrere Parameter werden durch Kommata getrennt und zu jedem Parameternamen eine Typbezeichnung angegeben werden.

Beispiel 1:

```
1  import Prog1Tools.IOTools;
2
3  public class MethodenDemo{
4
5     public static void Methode1(){
6       System.out.println("In Methode 1");
7       for(int i=0;i<=10;i++)
8         System.out.println(i+"  "+(i*i));
9     }
10
11    public static void Methode2(int endzahl){
12      System.out.println("In Methode 2");
13      for(int i=0;i<=endzahl;i++)
14        System.out.println(i+"  "+(i*i));
15      // hier kommt später etwas hin!
16    }
```



```
17
18    public static void main(String args[]){
19      int anzahl;
20
21      System.out.println("Hauptprogramm");
22      Methode1();
23
24      anzahl=IOTools.readInteger("Quadratzahl ausgeben bis?");
25      Methode2(anzahl);
26    }
27 }
```

Das Beispielprogramm MethodenDemo enthält 2 Methoden, die wir jetzt näher untersuchen:

- In den Zeilen 18-26 befindet sich das sogenannte Hauptprogramm, in Java ist es die sogenannte main-Methode. Auf die Parameter werden wie später zu sprechen kommen. Die Java Virtual Machine sucht bei jedem Java-Programm diese Methode und führt sie als erstes aus.
- `Methode1` ist eine Methode ohne Parameter und ohne Rückgabewert. Sie wird in den Zeilen 5-9 definiert. Eine Methode ohne Parameter dient dazu, ein bestimmtes Unterprogramm, das mehrmals in genau der gleichen Weise ablaufen soll, nur einmal programmieren bzw. schreiben zu müssen.
  Eine Methode ohne Parameter ruft man auf, indem man ihren Namen gefolgt von `();` notiert (siehe Zeile 22). Das aufrufende Programm wird an der entsprechenden Stelle angehalten, die aufgerufene Methode ausgeführt und anschließend das Programm an der Unterbrechungsstelle fortgesetzt.
- `Methode2` ist eine Methode mit einem Parameter und keinem Rückgabewert. Sie wird in den Zeilen 11-16 definiert. In solche Methoden werden Werte verarbeitet. Im Beispiel wird im Hauptprogramm die Anzahl der Wiederholungen eingelesen und in der Zeile 25 an die Methode2 übergeben.

Aufgabe 1:

Ergänze Beispiel 1 um folgendes:

- Ersetze den Kommentar in Zeile 15 durch: `endzahl++;`
- Füge vor } in der Zeile 26 die folgende Zeile ein: `System.out.println(anzahl);`

Was wird als letztes vom Programm ausgegeben? Bevor du das Programm startest, notiere deine Vermutung! Vergleiche die tatsächliche Ausgabe mit deiner Vermutung. War sie richtig?
Auf die Hintergründe werden wir auf der nächsten Seite eingehen.

Beispiel 2 a):

```
1  public class Wertetabelle1{
2
3    public static double f(double x){
4      double wert=Math.sqrt(x)+2*x;
5
6      return wert;
7    }
8
9    public static void main(String args[]){
10     double erg;
11
12     System.out.println("x          f(x)");
13     for(double x=0;x<=5;x=x+0.5){
```



```
14          erg=f(x);
15          System.out.println(x+"       "+erg);
16        }
17      }
18  }
```

Dieses Beispiel gibt eine Wertetabelle der Funktion $f(x) = \sqrt{x} + 2 \cdot x$ aus. Die mathematische Funktion wird in Java durch die gleichnamige Methode in den Zeilen 3 – 7 formuliert. Die Methode hat einen Parameter namens x vom Typ double und einen Rückgabewert vom Typ double. In ihr wird der Funktionswert `wert` an der Stelle `x` bestimmt. An das Hauptprogramm wird er durch die Zeile 6 „`return wert`“ zurückgeliefert. Außerdem wird durch das return die Methode verlassen.
Der Funktionswert wird dann durch eine Zuweisung in der Zeile 14 in der Variablen erg gespeichert. Dies passiert genauso wie bei einer vorgegebenen Java-Methode.

Dieses Beispiel läßt sich aber auch knapper formulieren. Die Hilfsvariablen erg und wert wurden nur für das Verständnis verwendet. Das folgende Beispiel arbeitet ohne sie.

Beispiel 2 b):

```
public class Wertetabelle2{

  public static double f(double x){
    return Math.sqrt(x)+2*x;
  }

  public static void main(String args[]){
    System.out.println("x          f(x)");
    for(double x=0;x<=5;x=x+0.5){
      System.out.println(x+"       "+f(x));
    }
  }
}
```

Regeln für Methoden:

1. **Block-Regel:**
   Als *Block* bezeichnet man einen Bereich zwischen { und }.
   Gültigkeitsbereich: Eine Konstante bzw. Variable ist in dem Programmblock gültig, in dem sie vereinbart wurde, sowie in allen untergeordneten Blöcken.
   Globaler Name: Eine Variable, die außerhalb von Methoden definiert wird, nennt man globale Variable (oder später auch statische Variable oder Klassenvariable). Sie hat im im gesamten Programm bzw. der gesamten Klasse Gültigkeit. Global heißt „in allen untergeordneten Methoden und Programmteilen bekannt und gültig“.
   Lokaler Name: Eine Konstante bzw. Variable, die in einer Prozedur vereinbart wurde, hat nur innerhalb dieser Prozedur und evtl. in weiteren von dieser Prozedur aufgerufenen Unterprogrammen Gültigkeit.
2. **Ausblenden-Regel:** Treten in einem Programm globale und lokale Größen mit gleichem Namen auf, so gilt innerhalb einer Prozedur nur der lokale Name. Die globale Größe wird vorübergehend ausgeblendet (ignoriert). Nach Verlassen der Prozedur hat die globale Größe noch ihren vorherigen Wert.
3. Methoden können nicht ineinander geschachtelt werden.

Parameterübergabe bei Methoden

Die Parameter in der Definition der Methoden bezeichnet man auch als *formale Parameter*.

Beim Aufruf einer Methode passiert folgendes:

- Man ruft die Methode mit ihrem Namen auf und übergibt die Werte der aktuellen Variablen oder Konstanten. Diese werden von links nach rechts ausgewertet. Man bezeichnet diese Parameter als *aktuelle Parameter*. Sie müssen mit den Datentypen der formalen Parameter übereinstimmen oder sich in diese umwandeln lassen („zuweisungskompatibel“).
- Das Ergebnis der Auswertung wird einer lokalen Variablen mit dem Namen des formalen Arguments abgelegt.

In der Methode wird also immer mit einer Kopie der übergebenen Daten gearbeitet. Deshalb nennt man die Parameter auch „Werteparameter“ und die Aufrufkonvention „call by value“.

Schnittstelle zwischen Methode und übergeordnetem Programm:

Um eine sauber definierte Schnittstelle zwischen Methode und übergeordnetem Programm zu erhalten, sollten innerhalb einer Methode möglichst keine globalen Variablen verwendet werden. Auf diese Weise können unerwünschte Nebeneffekte d.h. nicht geplante Veränderungen der Variableninhalte vermieden werden.

Regel für Variablen und Konstanten:
So lokal wie möglich - nur so global wie unbedingt nötig.

Aufgaben:

1. Schreibe das Beispielprogramm zur Berechnung der Quersumme so um, dass eine Funktion Quersumme verwendet wird.
2. Schreibe ein Programm, dass mit Hilfe einer Funktion f mit einem Parameter vom Typ double und dem Rückgabewert vom Typ double eine Wertetabelle ausgibt.
   f ist dabei folgendermaßen definiert: $f(x) = \frac{\sin(x)}{\sqrt{x+5}}$
   Hinweis: Der Sinuswert einer Zahl z (im Bogenmaß) wird in Java durch Math.sin(z) berechnet und die Wurzel einer Zahl x durch Math.sqrt(x).
3. Schreibe ein Java-Programm mit einer Methode namens `hoch` für ganze Zahlen, die einen Ausdruck der Form Basis $^{\text{Exponent}}$ berechnet und das Ergebnis zurückliefert. Das Hauptprogramm soll zwei Zahlen von der Tastatur einlesen, die Methode `hoch` aufrufen und das Ergebnis ausgeben.
4. Warum funktioniert das folgende Programm nicht?

```
public class WasIstLos{

  public static void vertausche(int a,int b){
    int hilf=a; a=b; b=hilf;
  }

  public static void main(String args[]){
    int x=2;
    int y=3;
    vertausche(x,y);
    System.out.println(x+"  "+y);
  }
}
```

