Übung 4, Lösungsvorschlag

TECHNISCHE UNIVERSITÄT DARMSTADT

Aufgabe 1 (Wissensfrage)

Beantworten Sie die folgenden Fragen:

a) Was bedeutet der Begriff „Modularität“? Welche Strukturen kennen Sie in Java, um Modularität zu gewährleisten?

b) Aus welchen Elementen setzt sich eine Methode in Java zusammen?

c) Ist der Datentyp `int` dem Datentyp `double` in Java „übergeordnet“ oder nicht? Welche Rolle spielt diese „Ordnung“ beim Casting von `int` nach `double` in Java?

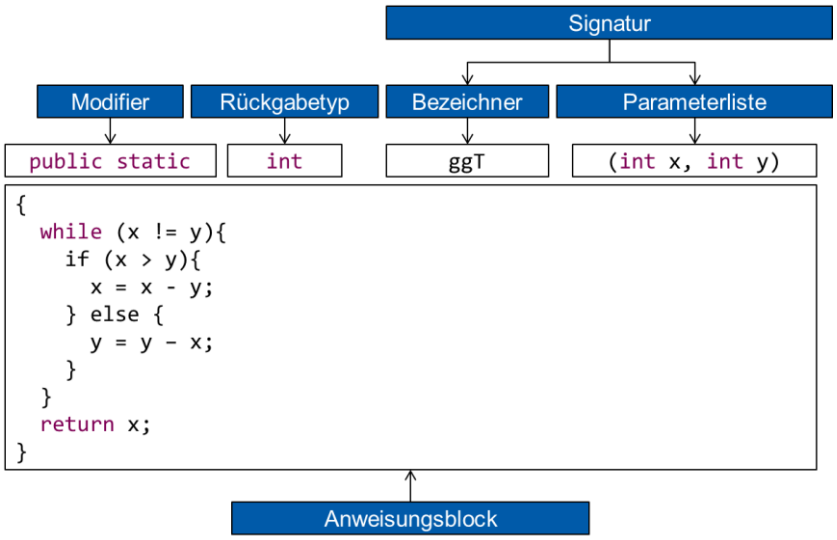
d) Welchen Wert enthält die Variable `y` nach der Abarbeitung dieser Anweisung:
```
double y = 8 / 3;
```
Begründen Sie Ihre Antwort.

e) Welche vier Arten von einfachen Datentypen existieren in Java?

Lösung Aufgabe 1

a) Reale Programme sind in der Regel sehr umfangreich. Anstatt den gesamten Code in einem großen Block „monolithisch“ zu programmieren wird das Gesamtproblem in viele kleine Probleme zerlegt, die dann unabhängig voneinander gelöst werden können. In Java werden Programme in verschiedene Methoden, Klassen, Pakete und Bibliotheken zerlegt, um diese modular zu gestalten.

b) Eine Methode besteht aus fünf Elementen: Modifier, Rückgabetyp, Bezeichner, Parameterliste sowie Anweisungsblock. Außerdem (das wurde hier nicht gefragt) ergeben der Bezeichner sowie die Parameterliste einer Methode deren Signatur.



c) Nein, der Datentyp `double` ist dem Datentyp `int` übergeordnet. Implizites Casting ist weiterhin nur dann möglich, wenn der umzuwandelnde Datentyp dem geforderten Datentyp untergeordnet ist. Aus diesem Grund ist kein implizites Casting vom Datentyp `double` in den Datentyp `int` möglich. Hier müsste ein explizites Casting angewendet werden. Hierbei muss jedoch einem Informationsverlust (abgeschnittene Nachkommastellen) gerechnet werden. Eine mathematische Rundung findet nicht statt!

d) Zunächst wird der Ausdruck `8 / 3` ausgewertet. Da es sich bei beiden Operanden um Ganzzahlen handelt, ist das Ergebnis des Ausdrucks ebenso vom Datentyp `int`. Obwohl die Division hier nicht restfrei möglich ist, können keine Nachkommastellen gespeichert werden! Damit werden alle Nachkommastellen abgeschnitten. Der Wert des Ausdrucks ist somit 2. Anschließend wird dieser Wert der `double`-Variable zugeordnet. Dabei erfolgt eine implizite Typumwandlung vom Datentyp `int` zum Datentyp `double`. In der Variable `y` wird somit der Wert `2.0` abgelegt.

e)

Wahrheitswerte	→	`boolean`
Einzelne Zeichen	→	`char`
ganze Zahlen	→	`byte, short, int, long`
(Gleit-) Kommazahlen	→	`float, double`

Aufgabe 2 (Wetterstation)

Eine Wetterstation hat für 14 Tage folgende Temperaturwerte aufgenommen:

Tag	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Temperatur	12	14	9	12	15	16	15	15	11	8	13	13	15	12

a) Erstellen Sie folgendes Java Programm: Legen Sie zuerst eine Klasse mit dem Namen „Wetterstation“ an. Legen sie in dieser Klasse zusätzlich noch eine main-Methode an. Deklarieren Sie innerhalb der main-Methode ein Array mit dem Bezeichner `temperaturen`, das ganzzahlige Werte speichert. Nutzen Sie hierzu die statische Erzeugung mit den Werten aus obiger Tabelle.

b) Schreiben Sie in diese Klasse nun unterhalb der main-Methode (also außerhalb des Anweisungsblocks) eine neue Methode mit folgendem Modifier, Rückgabetyp sowie Signatur:
`public static double berechneDurchschnittstemperatur(int[] temps).`
Die Methode bekommt beim Aufruf ein Integer-Array über den Parameter `temps` übergeben. Im Folgenden sollen Sie im Anweisungsblock der Methode die Durchschnittstemperatur für die im Array enthaltenen Temperaturen berechnen. Das Ergebnis soll vom Datentyp `double` sein und schlussendlich an den Aufrufer zurückgegeben werden. Nutzen Sie bei der Berechnung eine for-Schleife, um das Array zu durchlaufen und dabei das Ergebnis Schritt für Schritt zu berechnen.
Um die Methode zu testen, schreiben Sie nun innerhalb der main-Methode eine entsprechende Anweisung, die die Methode mit dem Array `temperaturen` aufruft. Geben Sie das Ergebnis (also die Durchschnittstemperatur) auf der Konsole aus.

c) Schreiben Sie nun eine zweite Methode (analog zu Aufgabenteil b)
`public static int berechneExtremtemperatur(int[] temps),`
die die maximale im übergebenen Array enthaltene Temperatur als Ganzzahl zurückgibt. Nutzen Sie hierbei eine while-Schleife, um das Array zu durchlaufen. Rufen Sie die Methode dann aus der Main-Methode auf und geben Sie wiederum aus der main-Methode den zurückgegebenen Wert auf der Konsole aus.

d) Schreiben Sie nun eine dritte Methode (analog zu Aufgabenteil b)
`public static int berechneUmschwung(int[] temps),`
die basierend auf dem übergebenen Array den Tag ermittelt und zurückgibt, an dem es den größten Temperaturumschwung (d.h. die größte Temperaturdifferenz zum Vortag) gab. Verwenden Sie hier eine do-while Schleife. Rufen Sie die Methode dann aus der Main-Methode auf und geben Sie den zurückgegebenen Wert wie zuvor auf der Konsole aus.

Lösung Aufgabe 2

```java
public class Wetterstation {
  public static void main(String[] args) {
    int[] temperaturen = { 12, 14, 9, 12, 15, 16, 15, 15, 11, 8, 13, 13,
        15, 12 };
    // Die folgenden Ausgaben beziehen sich auf die Aufgabenteile b) bis d)
    System.out.println(berechneDurchschnittstemperatur(temperaturen));
    System.out.println(berechneExtremtemperatur(temperaturen));
    System.out.println(berechneUmschwung(temperaturen));

  }

  // b)
  public static double berechneDurchschnittstemperatur(int[] temps) {
    double average = 0;
    for (int i = 0; i < temps.length; i++) {
      average = average + temps[i];
    }
    average = average / temps.length;
    return average;
  }

  // c)
  public static int berechneExtremtemperatur(int[] temps) {
    int max = temps[0];
    int i = 1;
    while (i < temps.length) {
      if (temps[i] > max) {
        max = temps[i];
      }
      i++;
    }
    return max;
  }

  // d)
  public static int berechneUmschwung(int[] temps) {
    int maxDiff = 0;
    int umschwung = 0;
    int tag = 0;
    int i = 1;
    do  {
      umschwung = temps[i] - temps[i - 1];
      if (umschwung < 0) {
        umschwung = umschwung * -1;
      }
      if (umschwung > maxDiff) {
        maxDiff = umschwung;
        tag = i + 1;
      }
      i++;
    } while (i < temps.length);
    return tag;
  }
}
```

Aufgabe 3 (Mehrdimensionale Arrays)

Implementieren Sie das folgende Programm in Java. Erstellen Sie hierzu eine Klasse mit dem Namen „Matrixuebung“ sowie eine main-Methode. Alle Aufgabenteile können Sie innerhalb des Anweisungsblocks der main-Methode implementieren.

a) Deklarieren und initialisieren Sie eine 3x2-Matrix mit den folgenden ganzzahligen Inhalten:

$$\begin{pmatrix} 2 & 3 \\ 21 & 42 \\ 0 & 1 \end{pmatrix}$$

Wählen Sie für die Variable den Bezeichner `matrix`.

b) Geben Sie nun die Werte der Matrix auf der Konsole aus. Verwenden Sie hierzu zwei Schleifen und gehen Sie die Arrays schrittweise durch (statt die Werte einzeln ausgeben zu lassen).

Die Ausgabe soll wie folgt aussehen:

```
Erste Matrix:
2     3
21    42
0     1
```

Hinweis: *Um aufeinanderfolgende Ausgaben auf der Konsole nebeneinander zu schreiben, können Sie für den ersten Aufruf die Methode* `System.out.print();` *nutzen. Die folgende Ausgabe wird in derselben Zeile dargestellt. Testen Sie zuerst die aufeinanderfolgenden vier Ausgaben, und machen Sie sich den Unterschied klar:*

```
System.out.print(1 + "   ");
System.out.println(2);
System.out.print(3 + "   ");
System.out.println(4);
```

c) Berechnen Sie nun die Summe der ersten drei Elemente der Matrix (gezählt wird von links nach rechts und von oben nach unten). Weisen Sie das Ergebnis dem vierten Element der Matrix zu. Die Summe soll in Ihrem Programm zur Laufzeit errechnet werden (und nicht bei der Initialisierung bereits gesetzt werden).

d) Ziehen Sie vom Wert des dritten Elements den Wert `5` ab und speichern Sie das Ergebnis erneut im dritten Element der Matrix.

e) Lassen Sie sich die Matrix erneut analog zu Teilaufgabe b) ausgeben. Die zweite Ausgabe soll nun wie folgt aussehen:

```
Zweite Matrix:
2     3
16    26
0     1
```

Lösung Aufgabe 3

```
01 public class Matrixuebung {
02   public static void main(String[] args) {
03
04     // Teilaufgabe (a)
05     int[][] matrix = { {2, 3}, {21, 42}, {0, 1} };
06
07     // Teilaufgabe (b)
08     System.out.println("Erste Matrix:");
09
10     for (int i = 0; i < matrix.length; i++) {
11       for (int j = 0; j < matrix[i].length; j++) {
12         System.out.print(matrix[i][j] + "  ");
13       }
14       System.out.println();
15     }
16
17     // Teilaufgabe (c)
18     matrix[1][1] = matrix[0][0] + matrix[0][1] + matrix[1][0];
19
20     // Teilaufgabe (d)
21     matrix[1][0] = matrix[1][0] - 5;
22
23     // Teilaufgabe (e)
24     System.out.println("Zweite Matrix:");
25
26     for (int i = 0; i < matrix.length; i++) {
27       for (int j = 0; j < matrix[i].length; j++) {
28         System.out.print(matrix[i][j] + "  ");
29       }
30       System.out.println();
31     }
32   }
33 }
```