TU Ilmenau, Fakultät IA
FG Telematik/Rechnernetze
Prof. Dr.-Ing. G. Schäfer
https://www.tu-ilmenau.de/telematik/pua

TECHNISCHE UNIVERSITÄT
ILMENAU

Programmierung und Algorithmen – WS 26/27

Übungsblatt 2

Die Lösungen der Aufgaben sind bis zum 01.11.26, 23:59 Uhr abzugeben.

Die Besprechung der Aufgaben erfolgt in KW 45.

Aufgabe 1 (Java Einmaleins) **7 Punkte**

Das untenstehende Java-Programm gibt sieben Zeilen auf der Konsole aus. Geben Sie den Inhalt jeder Zeile an und begründen Sie Ihre Antwort kurz. Ohne Begründung kann Ihre Antwort nicht gewertet werden!

```
1  public static void main(String[] args) {
2      int i = 8;
3      double d = --i;
4      System.out.println(i + ", " + d);
5      switch(i) {
6          case 7: i = 4;
7          default: i = 0;
8      }
9      System.out.println(i);
10      short s = 32767;
11      System.out.println(++s);
12      d = d + --d;
13      System.out.println(d);
14      System.out.println(d > 0 ? "+" : "-");
15      d = 28 / 10;
16      System.out.println(d);
17      int[] a = {21, 22, 23};
18      int[] b = a;
19      b[0] = 0;
20      System.out.println(a[0]);
21  }
```

Aufgabe 2 (Datentypen in Java) **5 Punkte**

Entscheiden Sie für die folgenden Werte (!), ob Java *primitive Datentypen* für ihre Repräsentation zur Verfügung stellt. Falls ja, geben Sie *alle* passenden an.

(a) `true`
(b) -32768
(c) $\{1, 4, 2\}$
(d) `"Hello Java"`
(e) $\sqrt{2}$
(f) $\frac{3}{2}$
(g) ‚Z‘
(h) 9223372036854775807

Bitte wenden!

Aufgabe 3 (Schleifenkonstrukte) **6 Punkte**

Die Schleifenkonstrukte `for`, `while` und `do … while` besitzen dieselbe Ausdruckskraft. Implementieren Sie in Java *drei unterschiedliche Varianten* zur Berechnung der Summe aller Elemente eines gegebenen Arrays `int[] a`. Verwenden Sie dabei jeweils ein anderes Schleifenkonstrukt (zum Beispiel implementiert in drei Methoden).

Hinweis: Die Summe eines leeren Arrays sei als `0` definiert.

Aufgabe 4 (Java) **3 + 3 Punkte**

Ein schnelles Schätzverfahren zur Berechnung der Quadratwurzel von $x \in \mathbb{N}, x \geq 1$ funktioniert wie folgt:

1. Suche die größte Zahl $y \in \mathbb{N}$, für die $x = y^2 + c$ mit $c \in \mathbb{Z}, c \geq 0$ gilt.
2. Berechne $\sqrt{x} \approx y + \frac{c}{2y}$

(a) Implementieren Sie das beschriebene Verfahren in Java in einer Methode mit folgender Signatur:

```
public static double approxSqrt(int x)
```

(b) Implementieren Sie zudem eine „Testroutine“, die ermittelt für welche natürliche Zahl zwischen 1 und 1000 (einschließlich) das Verfahren die *schlechteste* Approximation liefert. Diese Zahl soll auf der Konsole ausgegeben werden. Verwenden Sie als Vergleichswert `Math.sqrt(x)`. Die Testroutine kann zum Beispiel folgende Signatur aufweisen:

```
public static void testSqrt()
```

Hinweis: Da die Signatur der Methode aus Teilaufgabe (a) bekannt ist, können Sie Teilaufgabe (b) auch dann bearbeiten, wenn Sie (a) nicht gelöst haben.

Aufgabe 5 (Anagrammprüfung) **5 Punkte**

Schreiben Sie ein Java-Programm, welches testet, ob zwei Wörter ein Anagramm[1] bilden. Groß- und Kleinschreibung soll dabei nicht berücksichtigt werden. Sie dürfen ferner davon ausgehen, dass die Wörter nur aus lateinischen Buchstaben bestehen (regulärer Ausdruck: `(a+...+z+A+...+Z)`*). Die Eingabe muss über die Kommandozeilenparameter erfolgen (`args[0], args[1]`). Beispiele für Anagramme sind:

- Palindrom – Aprilmond
- Bundestag – Angstbude
- Ingenieur – Eigenruin

Hinweis: Sie dürfen die Methoden der `String`-Klasse[2] zur Hilfe nehmen. Methoden anderer Klassen dürfen jedoch nicht verwendet werden!

[1] Anagramm: Durch Umstellung von Buchstaben innerhalb eines Wortes entstandenes neues sinnvolles Wort.
[2] https://docs.oracle.com/en/java/javase/25/docs/api/java.base/java/lang/String.html