Test 2 in Programmkonstruktion – 1. Phase

18 / 30 Punkte

1. Multiple-Choice-Aufgaben

12 / 21 Punkte

Bitte wählen Sie *alle* zutreffenden Antwortmöglichkeiten aus. Es können beliebig viele Antwortmöglichkeiten zutreffen, auch alle oder keine.

Aufgabe 1.1.

2.5 / 3 Punkte

Angenommen, Variable `x` ist vom Typ `long[]` und `y` vom Typ `long[][]`. Wählen Sie jene Zuweisungen aus, die der Java-Compiler als fehlerhaft erkennt:

- [x] `y[0] = "String"`
- [ ] `x[4][4] = y[9]`
- [ ] `x[0] = y[0][9]`
- [ ] `x = y[8]`
- [ ] `y[9] = x`
- [x] `x[0] = y[3]`

Aufgabe 1.2.

2.5 / 3 Punkte

Wählen Sie jene Ausdrücke aus, die in Java ein Array erzeugen, welches an mindestens einer Stelle `null` enthält:

- [ ] `new double[9][9]`
- [ ] `new String[1][9]`
- [x] `new char[8][]`
- [x] `new String[1][]`
- [x] `new long[7][][]`
- [ ] `new char[][]{new char[9], new char[9]}`

Aufgabe 1.3.

2 / 3 Punkte

Angenommen, `x` ist eine frisch initialisierte Variable vom Typ `Deque<String>` und `s` eine Variable vom Typ `String`. Wählen Sie jene Anweisungsfolgen aus, die dazu führen, dass nach Ausführung `s.equals("ab")` gilt:

- [ ] `x.offerFirst("a"); x.offerFirst("b"); s = x.poll(); s += x.poll();`
- [ ] `x.offer("a"); x.offer("b"); s = x.poll(); s += x.poll();`
- [ ] `x.offer("a"); x.offerFirst("b"); s = x.poll(); s += x.poll();`
- [ ] `x.offer("a"); x.offer("b"); s = x.poll(); s += x.pollLast();`
- [ ] `x.offerFirst("a"); x.offer("b"); s = x.pollLast(); s += x.poll();`
- [x] `x.offerFirst("a"); x.offerFirst("b"); s = x.pollLast(); s += x.pollLast();`

Aufgabe 1.4.

1.5 / 3 Punkte

Wählen Sie jene Anweisungen bzw. Anweisungsfolgen aus, in denen nach Ausführung `x[0] != x[1]` gilt:

- [ ] `String s = "a"; String[] x = { s, s };`
- [ ] `char[] x = new char[2];`
- [ ] `String[][] x = new String[2][2];`
- [ ] `int[][] x = new int[2][3];`
- [ ] `int[][] x = { new int[]{1}, new int[]{1} };`
- [x] `int[] x = { 1, 2 };`

Aufgabe 1.5.

1.5 / 3 Punkte

Wählen Sie jene Anweisungen aus, die in normalen (nicht `static`) Methoden *nicht* vorkommen dürfen:

- [x] `this = x;`
- [ ] `this.x = y;`
- [ ] `this();`
- [ ] `y = this(x);`
- [ ] `y = this.x;`
- [ ] `this(x);`

Aufgabe 1.6. 2 / 6 Punkte

Wählen Sie jene Definitionen der Java-Methode `f` aus, die für alle Parameterwerte im Wertebereich von -10 bis 10 (ohne Überlauf) terminieren:

- [x] `int f(int x) { return x % 2 == 0 ? 1 : f(x - 2) * 2; }`
- [ ] `int f(int x) { return x >= 0 ? 1 : f(x / 4) * 2; }`
- [ ] `int f(int x) { return x <= 0 ? 1 : f(x % 2 - 1) * 2; }`
- [x] `int f(int x) { return x > 0 ? 1 : f(x * x) * 2; }`
- [ ] `int f(int x) { return x < 0 ? 1 : f(x / 2 - 1) * 2; }`
- [ ] `int f(int x) { return x > 0 ? 1 : f(x + 1) * 2; }`

2. Auswahlaufgaben

6 / 9 Punkte

Jede dieser Aufgaben hat genau eine zutreffende Antwortmöglichkeit. Bitte wählen Sie diese aus.

Aufgabe 2.1. 0 / 3 Punkte

Was versteht man unter der *Wurzel* eines Baums?

- ○ Knoten mit mehreren Vorgängern.
- ● Knoten mit mehreren Nachfolgern.
- ○ Knoten ohne Vorgänger.
- ○ Knoten ohne Nachfolger.
- ○ Knoten mit genau einem Vorgänger.
- ○ Knoten mit genau einem Nachfolger.

Aufgabe 2.2. 3 / 3 Punkte

Warum sollen Fließkommazahlen nicht mittels `==` verglichen werden?

- [ ] Weil Fließkommazahlen generell vermieden werden sollen, daher auch Vergleiche darauf.
- [ ] Aus Effizienzgründen.
- [x] Weil `==` wegen Rundungsfehlern auch bei als gleich anzusehenden Zahlen `false` ergeben kann.
- [ ] Weil `==` wegen Rundungsfehlern auch bei als verschieden anzusehenden Zahlen `true` ergeben kann.
- [ ] Weil die Zahlen selbst verglichen werden sollen, nicht nur Referenzen darauf.
- [ ] Es stimmt nicht. Fließkommazahlen soll man mit `==` vergleichen.

Aufgabe 2.3. 3 / 3 Punkte

Was versteht man unter Datenabstraktion?

- [ ] Daten und Methoden sind klar voneinander getrennt und die Sichtbarkeit ist eingeschränkt.
- [ ] Daten und Methoden sind als Einheit unbeschränkt sichtbar.
- [ ] Interfaces ermöglichen die abstrakte Verwendung von Objekten.
- [ ] Daten und Methoden sind klar voneinander getrennt, aber dennoch überall sichtbar.
- [x] Daten und Methoden bilden eine Einheit und die Sichtbarkeit ist eingeschränkt.
- [ ] Große Methoden werden auf mehrere keinere Methoden aufgespaltet.