

Test 2 in Programmkonstruktion – 1. Phase

25.5 / 30 Punkte

1. Multiple-Choice-Aufgaben

19.5 / 21 Punkte

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

Aufgabe 1.1.

3 / 3 Punkte

Wählen Sie jene Anweisungen aus, die zu Beginn eines Konstruktors vorkommen dürfen, sonst aber nirgends:

`this();`

`this = x;`

`this.x = y;`

`this(x);`

`y = this.x;`

`y = this(x);`

Aufgabe 1.2.

3 / 3 Punkte

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

`new String[]{"a", "b", "c"}`

`new int[9][6]`

`new boolean[1][]`

`new String[8][9]`

`"A000"`

`new double[5][1][9]`

Aufgabe 1.3.

3 / 3 Punkte

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

`y[0][8] = x[9]`

`x[9] = y`

`x[8][6] = y[0][3]`

`y = x[0]`

`x[9] = y[0]`

`x[3][0] = y[0]`

Aufgabe 1.4.

3 / 3 Punkte

Angenommen, `x` ist eine Variable vom Typ `Deque<Integer>` und `y` eine Variable vom Typ `int[2]`, beide Variablen mit neuen Objekten initialisiert. Wählen Sie jene Anweisungsfolgen aus, die dazu führen, dass nach Ausführung `y[0] == 0 && y[1] == 1` gilt:

`x.offerFirst(0); x.offerFirst(1); y[0] = x.pollLast(); y[1] = x.pollLast();`

`x.offerFirst(0); y[0] = x.poll(); x.offerFirst(1); y[1] = x.poll();`

`x.offer(0); x.offer(1); y[0] = x.poll(); y[1] = x.poll();`

`x.offerFirst(0); x.offer(1); y[0] = x.poll(); y[1] = x.poll();`

`x.offer(0); x.offer(1); y[0] = x.pollLast(); y[1] = x.poll();`

`x.offer(0); x.offerFirst(1); y[0] = x.poll(); y[1] = x.poll();`

Aufgabe 1.5.

2.5 / 3 Punkte

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

`int[][] x = new int[2][];`

`String[] x = new String[2];`

`int[] x = new int[2];`

`int[] x = { 1, (int)1.7 };`

`int[][] x = { new int[] {}, new int[] {} };`

`String s = "a"; String[] x = { s, s };`

Aufgabe 1.6.

5 / 6 Punkte

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

`int f(int x) { return x < 0 ? 0 : f(x / 2 - 1) + 1; }`

`int f(int x) { return x == 0 ? 0 : f((x % 2) * 2) + 1; }`

`int f(int x) { return x % 2 == 0 ? 0 : f(x - 3) + 1; }`

`int f(int x) { return x > 0 ? 0 : f(x + 2) + 1; }`

`int f(int x) { return x < 0 ? 0 : f(x * -x) + 1; }`

`int f(int x) { return x < 0 ? 0 : f(x / 2) + 1; }`

2. Auswahlaufgaben

6 / 9 Punkte

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

Aufgabe 2.1.

3 / 3 Punkte

Unter welcher Bedingung ist ein Knoten im Baum ein *Blatt*?

- Der Knoten ist keine Wurzel eines Teilbaums.
- Unter dem Knoten hängt kein Teilbaum.
- Der Knoten gehört zu einem Baum ohne Wurzel.
- Unter dem Knoten hängen zwei Teilbäume.
- Unter dem Knoten hängt höchstens ein Teilbaum.
- Der Knoten gehört zu einem Baum mit mehreren Wurzeln.

Aufgabe 2.2.

0 / 3 Punkte

Was unterscheidet eine `static`-Methode von einer nicht-`static` Methode?

- Die `static`-Methode erlaubt rekursive Aufrufe.
- Die `static`-Methode braucht mindestens einen Parameter.
- Die `static`-Methode kann einen Konstruktor enthalten.
- Die `static`-Methode ist nach außen sichtbar.
- Die `static`-Methode hat Zugriff auf `this`.
- Die `static`-Methode gibt ein Ergebnis zurück.

Aufgabe 2.3.

3 / 3 Punkte

Was versteht man unter Datenkapselung?

- Das Verstecken von Daten vor unberechtigten Zugriffen.
- Das Zusammenfügen von Daten und Methoden zu einer Einheit.
- Die klare Trennung zwischen Daten und Methoden.
- Die Abstraktion von Objekten durch Verwendung von Interfaces.
- Das Ermöglichen von Zugriffen auf Daten von außen.
- Die Abstraktion durch Aufspalten großer Methoden auf mehrere kleinere.