

## Test 3 in Programmkonstruktion – 1. Phase

24.5 / 30 Punkte

### 1. Multiple-Choice-Aufgaben

15.5 / 18 Punkte

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

#### Aufgabe 1.1.

2 / 3 Punkte

Wählen Sie jene Ausdrücke aus, die in einem korrekten Java-Programm immer `true` ergeben, wobei `x` ein Interface ist, `a` durch `x a`; deklariert wurde und `a != null` gilt:

☐ `int.class instanceof Class`

☒ `!a.equals(null)`

☐ `null instanceof X`

☒ `a instanceof X`

☒ `a.toString().equals(a + "")`

☒ `a.getClass().equals(X.class)`

#### Aufgabe 1.2.

3 / 3 Punkte

Angenommen, der Ausdruck `x.equals(y)` liefert `true`. Wählen Sie jene Ausdrücke aus, die an derselben Programmstelle in einem korrekten Java-Programm ebenfalls immer `true` liefern:

☒ `x.equals(x)`

☒ `x.hashCode() == y.hashCode()`

☒ `y.equals(x)`

☐ `x.toString() == y.toString()`

☐ `x == y`

☒ `x != null && y != null`

### Aufgabe 1.3.

2.5 / 3 Punkte

Wählen Sie jene Arten von Variablen und Methoden aus, die in einem Java-Interface vorkommen dürfen:

☒ als `final static` deklarierte Variablen

☐ als `abstract` deklarierte Variablen

☐ als `private` deklarierte Variablen

☐ als `private` deklarierte Methoden

☐ als `abstract` deklarierte Methoden

☒ als `public` deklarierte Methoden

### Aufgabe 1.4.

2.5 / 3 Punkte

Wählen Sie jene Arten von Zusicherungen aus, für deren Einhaltung entsprechend Design-by-Contract der *Client* sorgen muss:

☐ Methodendeklaration

☐ `assert`-Anweisung

☒ Vorbedingung

☐ Invariante

☐ Nachbedingung

☒ Exception

### Aufgabe 1.5.

3 / 3 Punkte

Wählen Sie jene Modifier aus, die bewirken, dass Aufrufe entsprechend deklarerter Java-Methoden immer statisch gebunden werden:

☐ `abstract`

☒ `final`

☐ `void`

☒ `static`

☐ `public`

☒ `private`

### Aufgabe 1.6.

2.5 / 3 Punkte

Eine Klasse `C` implementiert ein Interface `X` in einem Java-Programm. Wählen Sie jene Aussagen aus, die in diesem Fall zutreffen:

☐ `X` ist Oberklasse von `C`.

☒ `X` ist Obertyp von `C`.

☒ `C` ist Unterklasse von `X`.

☒ Variablen vom deklarierten Typ `X` können alle Objekte vom Typ `C` enthalten.

☐ Variablen vom deklarierten Typ `C` können alle Objekte vom Typ `X` enthalten.

☒ `C` ist Untertyp von `X`.

## 2. Auswahlaufgaben

9 / 12 Punkte

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

### Aufgabe 2.1.

0 / 3 Punkte

An welcher Stelle im Programm findet man üblicherweise die *Invarianten* eines Objekts?

☐ am Beginn von Blöcken

☒ bei Variablendeklarationen

☐ zwischen zwei Anweisungen

☐ am Beginn von Schleifen

☐ bei Methodenköpfen

☒ am Beginn von Konstruktoren

### Aufgabe 2.2.

3 / 3 Punkte

Welche der folgenden Exceptions muss in einer `throws`-Klausel stehen, falls Sie in einer Java-Methode auftreten kann und nicht abgefangen wird?

☐ `NullPointerException`

☐ `ArrayIndexOutOfBoundsException`

☐ `ArithmeticException`

☐ `StackOverflowError`

☒ `IOException`

☐ `AssertionError`

### Aufgabe 2.3.

3 / 3 Punkte

An welcher Stelle einer Schleife braucht eine Schleifeninvariante *nicht* gelten?

- ☐ am Ende des Schleifenrumpfs
- ☐ direkt vor der Schleife
- ☐ direkt nach der Schleife
- ☐ zu Beginn des Schleifenrumpfs
- ☒ zwischen zwei Anweisungen im Schleifenrumpf

### Aufgabe 2.4.

3 / 3 Punkte

Was bewirkt `final` als Modifier einer Java-Klasse?

- ☐ Variablen in der Klasse sind nach der Initialisierung unveränderbar.
- ☒ Die Klasse hat keine Unterklassen.
- ☐ Die Klasse enthält keine statischen Methoden.
- ☐ Der Compiler meldet einen Syntaxfehler.
- ☐ Vor dem Weiterleiten einer Exception wird aufgeräumt.
- ☐ Die Klasse hat keine Instanzen.