Test 3 in Programmkonstruktion – 1. Phase

24 / 30 Punkte

1. Multiple-Choice-Aufgaben

15 / 18 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

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

- [x] `C` ist Unterklasse von `X`.
- [x] `X` ist Obertyp von `C`.
- [x] Jedes Objekt vom Typ `C` ist auch Objekt vom Typ `X`.
- [ ] `X` ist Oberklasse von `C`.
- [ ] Jedes Objekt vom Typ `X` ist auch Objekt vom Typ `C`.
- [x] `C` ist Untertyp von `X`.

Aufgabe 1.2.

1.5 / 3 Punkte

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

- [x] `x.equals(y)`
- [ ] `x == null || y == null`
- [x] `x.toString().equals(y.toString())`
- [x] `y.equals(x)`
- [ ] `x.equals("false")`
- [ ] `x == y`

Aufgabe 1.3. 2.5 / 3 Punkte

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

- [x] `a.equals(null)`
- [x] `null instanceof X`
- [ ] `int.class instanceof Object`
- [ ] `a instanceof X`
- [ ] `a.toString().equals("" + a)`
- [ ] `X.class.equals(a.getClass())`

Aufgabe 1.4. 3 / 3 Punkte

Wählen Sie jene Arten von Zusicherungen aus, bei denen der *Client* entsprechend Design-by-Contract darauf vertrauen darf, dass sie eingehalten sind:

- [ ] Exception
- [ ] Variablendeklaration
- [x] Invariante
- [x] Nachbedingung
- [ ] Vorbedingung
- [ ] Konstruktor

Aufgabe 1.5. 3 / 3 Punkte

Wählen Sie jene Deklarationen aus, die in einem Java-Interface vorkommen dürfen:

- [x] `abstract boolean bar();`
- [ ] `private boolean empty();`
- [x] `public void foo();`
- [ ] `boolean done;`
- [x] `public final static double V = 0.09;`
- [ ] `static void baz();`

Aufgabe 1.6. 2.5 / 3 Punkte

Wählen Sie jene Modifier aus, die bewirken, dass damit deklarierte Java-Methoden in Unterklassen nicht vorhanden, nicht sichtbar oder nicht überschreibbar sind:

- [ ] `abstract`
- [ ] `public`
- [x] `private`
- [ ] `void`
- [x] `final`
- [ ] `static`

2. Auswahlaufgaben 9 / 12 Punkte

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

Aufgabe 2.1. 0 / 3 Punkte

Wo in einer Klasse braucht eine Invariante auf einem Objekt *nicht* gelten?

- ○ am Ende eines Konstruktors
- ○ zwischen zwei einfachen Anweisungen
- ● nach einem rekursiven Methodenaufruf
- ○ am Anfang einer Methode
- ○ vor einem rekursiven Methodenaufruf
- ○ am Ende einer Methode

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?

- ○ `RuntimeException`
- ○ `NullPointerException`
- ○ `ArrayIndexOutOfBoundsException`
- ○ `ArithmeticException`
- ● `Exception`
- ○ `StackOverflowError`

Aufgabe 2.3. 3 / 3 Punkte

An welcher Stelle im Programm findet man üblicherweise die *Vorbedingungen* einer Methode?

- ○ bei einer Schleife
- ○ beim Kopf der Klasse
- ○ bei Deklarationen lokaler Variablen
- ○ zwischen zwei Anweisungen in der Methode
- ● beim Kopf der Methode
- ○ bei einer Programmverzweigung

Aufgabe 2.4. 3 / 3 Punkte

Was bewirkt `abstract` als Modifier einer Java-Methode?

- ○ Die Methode wird von einer Oberklasse geerbt.
- ○ Der Compiler meldet einen Syntaxfehler.
- ○ Die Methode ist in Unterklassen nicht überschreibbar.
- ○ Die Methode kommt in Unterklassen nicht vor.
- ○ Die Methode kommt nur in einem Interface vor.
- ● Die Methode ist nicht implementiert.