Begonnen am	Thursday, 25. June 2020, 09:35
Status	Beendet
Beendet am	Thursday, 25. June 2020, 10:05
Verbrauchte Zeit	29 Minuten 50 Sekunden
Bewertung	**271,67** von 300,00 (**91**%)

Information

Bitte beachten Sie:

- 30 Minuten Zeit
- maximal **30 Punkte**. Anmerkung: Die Punkte im Test sind mit **10 skaliert**. Die Bewertungen der Fragen ergeben insgesamt 300 Punkte, die dann durch durch 10 dividiert werden.
- Bitte Fragen genau lesen
- Bei Antworttexten bitte kurze und präzise Antworten geben! Nur die gestellten Fragen beantworten. Antworten, die für die Fragestellung nicht relevant sind, führen zu Punkteabzügen.
- Bei Multiple-Choice-Fragen sind mehrere Antworten möglich. Falsche und fehlende Ankreuzungen führen zu Punkteabzügen.

Frage 1

Richtig

Erreichte Punkte 25,00 von 25,00

Kreuzen Sie an, in welchem Bereich ein mit dem Sichtbarkeitsattribut **protected** markiertes Programmelement sichtbar ist:

Wählen Sie eine oder mehrere Antworten:

- [x] in Unterklassen in fremden Packages ✔
- [x] in der Klasse selbst ✔
- [x] in Unterklassen im gleichen Package ✔
- [ ] in Klassen in fremden Packages
- [x] in Klassen im gleichen Package ✔

Die Antwort ist richtig.

Die richtigen Antworten sind: in der Klasse selbst, in Klassen im gleichen Package , in Unterklassen im gleichen Package , in Unterklassen in fremden Packages

Frage 2

Vollständig

Erreichte Punkte 20,00 von 20,00

Was bedeutet das Attribut final bei Methoden?

unveränderbar, die Methode kann nicht überschrieben werden. Also eine Unterklasse darf die Methode nicht überschreiben (@Override verboten)

Kommentar:

Frage 3

Teilweise richtig

Erreichte Punkte 16,67 von 25,00

Bei einer Methode, die eine Methode aus der Basisklasse überschreibt darf:

Wählen Sie eine oder mehrere Antworten:

- [x] der Rückgabewert spezieller sein ✔
- [ ] der Rückgabewert allgemeiner sein
- [ ] die Methode mehr Exceptions werfen
- [ ] die Sichtbarkeit erweitert werden
- [x] die Methode weniger Exceptions werfen ✔

Die Antwort ist teilweise richtig.

Sie haben 2 richtig ausgewählt.
Die richtigen Antworten sind: die Sichtbarkeit erweitert werden, der Rückgabewert spezieller sein, die Methode weniger Exceptions werfen

Frage 4

Richtig

Erreichte Punkte 20,00 von 20,00

Welche Antwort ist richtig?

Wählen Sie eine oder mehrere Antworten:

- [ ] Der dynamische Typ einer Variablen muss gleich ihrem statischen Typ sein!
- [ ] Der dynamische Typ einer Variablen muss spezieller als ihr statischer Typ sein!
- [ ] Der dynamische Typ einer Variablen muss allgemeiner als ihr statischer Typ sein!
- [x] Der dynamische Typ einer Variablen kann spezieller als ihr statischer Typ sein! ✔

Die Antwort ist richtig.

Die richtige Antwort lautet: Der dynamische Typ einer Variablen kann spezieller als ihr statischer Typ sein!

Frage 5

Vollständig

Erreichte Punkte 30,00 von 30,00

Erklären Sie den Unterschied zwischen

- statischen Member-Klassen
- nicht-statischen Member-Klassen

statische Member Klasse ist statisch gebunden, von ihr können Objekte erzeugt werden dass ein Objekt der äußeren Klasse erzeugt wurde.

Nicht statische Member Klassen sind eng mit einem von der äußeren Klasse erzeugten objekt verbunden. Instanz kann nur über ein von der äußeren Klasse erzeugtes Objekt erzeugt werden.

Kommentar:
Klassen sind nicht gebunden!

Frage 6

Richtig

Erreichte Punkte 20,00 von 20,00

Die folgende Operation *copyTo* kopiert die Elemente der Liste *from* in die Liste *to*. Bei der Typangabe für den Parameter *from* wird ein Wildcard verwendet.

```
public static <T> void copyTo(List<? extends T > 
from, List<T> to) {
  for (T e : from) {
    to.add(e);
  }
}
```

Wie lautet der Typconstraint?

Frage 7

Richtig

Erreichte Punkte 20,00 von 20,00

Was zeichnet Funktionale Interfaces aus?

Wählen Sie eine oder mehrere Antworten:

- [ ] Haben nur statische Methoden
- [ ] Haben nur Default-Methoden
- [x] Haben nur eine abstrakte Methode
- [ ] Haben nur eine Methode

Die Antwort ist richtig.

Die richtige Antwort lautet: Haben nur eine abstrakte Methode

Frage 8

Vollständig

Erreichte Punkte 40,00 von 40,00

Erklären Sie in einer kurzen Beschreibung die Prinzipien von JUnit-Tests.

Unit Test bedeutet kleine Einheiten (Units) des Programms zu teste, z.b. einzelne Objektmethoden. Dabei wird z.b. ein Objekt auf gewisse weise initialisiert, dann eine Methode darauf angewendet und dann überprüft ob das ergebnis (z.b. wert eines Felds des Objekts) dem erwarteten Wert entspricht.

Ein Test läuft also z.b. so ab:
initialisieren (@BeforeAll, setUp())
Ergebnis Prüfen (assert...)
Ende (@AfterAll, @AfterEache)

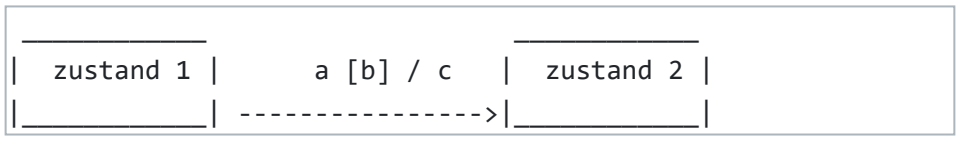
Kommentar:

Frage 9

Richtig

Erreichte Punkte 20,00 von 20,00

Bei UML Zustandsdiagrammen werden Kanten in folgender Form mit drei Teilen *a*, *b* und *c* beschriftet:



Für was stehen die drei Teile?

trigger ✔ [conditon ✖] / action ✖

Kommentar:

Frage 10

Vollständig

Erreichte Punkte 30,00 von 30,00

Erklären Sie was mit den folgenden UML Diagrammen modelliert wird:

- UML Sequenzdiagrammen
- UML Zustandsdiagrammen

Sequenzdiagramm modelliert ein Szenario, dabei wird abgebildet welche Objekte beteiligt sind und welche Nachrichten ausgetauscht werden

Zustandsdiagramm beschreibt die Zustände eines Objekt, also in denen es sich von Erzeugung bis Freigabe befinden kann.

Kommentar:

Frage 11

Vollständig

Erreichte Punkte 0,00 von 20,00

Was wird beim Moduldeskriptor *module-info.java* mit folgenden beiden Anweisungen spezifiziert?

- exports
- requires

exports:

requires:

Kommentar:

exports: exportierte Packages

requires: benötigte Module

Frage 12

Vollständig

Erreichte Punkte 30,00 von 30,00

Gegeben sei eine generische Klasse *List* mit Methoden *get* und *add* wie folgt:

```
public class List<E> {
  public E get(int i) { ... }
  public void add(E e) { ... }
}
```

Des Weiteren seien Klasse *Person* und Subklasse *Student* gegeben:

```
class Person {...}
class Students extends Person {...}
```

Von dieser Klasse wird eine Instanz mit Elementtyp *Student* erzeugt:

```
List<Student> students = new List<Student>();
```

Begründen Sie, warum eine kovariante Zuweisung wie folgt nicht erlaubt ist?

```
List<Person> persons = students;
```

die Zuweisung ist nicht erlaubt weil sie beim Schreiben nicht typsicher ist. In die Liste persons könnte man Person Objekte einfügen. students müsste aber garantieren dass die get() Methode ein Objekt vom Typ Student liefert und nicht von Person.

Kommentar:

◄ Klausur 1: Praktischer Teil

Direkt zu:

00.Inhalt ►