Prüfung Software Engineering II (IB)

Datum	:	01.02.2016, 14:30 Uhr
Bearbeitungszeit	:	90 Minuten
Prüfer	:	Prof. Dr. Oliver Braun
Hilfmittel	:	Keine
Erreichbare Punkte	:	90

Name: _____

Vorname: _____

Matrikelnummer: _____ Studiengruppe: _____

Hörsaal: _____ Platz Nr.: _____

Unterschrift: _____

Bitte kontrollieren Sie, ob Sie eine vollständige Angabe mit 8 Aufgaben auf 11 Seiten erhalten haben.

Aufgabe	1	2	3	4	5	6	7	8	Summe
max. Punkte	10	10	12	20	9	11	6	12	90

Anmerkungen:

- Nutzen Sie einen dokumentenechten Stift für alles was bewertet werden soll. Auch bei Skizzen ist die Verwendung eines **Bleistifts nicht** zulässig.
- Schreiben Sie die Lösungen in die dafür vorgesehenen Kästchen bzw. direkt zur Aufgabe. Sollte Ihnen der Platz dabei nicht reichen, benutzen Sie die Rückseite **und vermerken Sie das bei der entsprechenden Aufgabe!**

Aufgabe 1 (10 Punkte)

Erklären Sie die folgenden Begriffe im Zusammenhang mit dem Testen von Software kurz und in eigenen Worten.

(a) Softwareinspektionen (2)

(b) Validierung (2)

(c) Entwicklertests (2)

(d) statische Verifikations- und Validierungstechniken (2)

(e) Blackbox-Test (2)

Aufgabe 2 (10 Punkte)

Gegeben sei folgender Scala-Code:

```
import scala.collection.immutable.Queue

class Stove(capacity: Int = 1) {

  private var queue = Queue[List[Pizza]]()

  def +=(pizza: Pizza): Stove = {
    if (queue.isEmpty) {
      queue = Queue(List(pizza))
    } else {
      val last: List[Pizza] = queue.last
      queue = if (last.length < capacity) {
        queue.init :+ (pizza :: last)
      } else {
        queue :+ List(pizza)
      }
    }
    this
  }

  def +=(listOfPizza: List[Pizza]): Stove =
    listOfPizza.foldLeft(this) { (stove, pizza) => stove += pizza }

  def next(): List[Pizza] =
    if (queue.isEmpty) {
      List()
    } else {
      val first = queue.head
      queue = queue.tail
      first
    }

}
```

und dazu folgende Spec:

```
1  import models.{Stove, Pizza}
2  import org.junit.runner.RunWith
3  import org.specs2.mutable.Specification
4  import org.specs2.runner.JUnitRunner
5  import org.specs2.ScalaCheck
6
7  @RunWith(classOf[JUnitRunner])
8  class StoveSpecification extends Specification with ScalaCheck {
9
```

```
10    "The stove" should {
11
12      "... (Test 1)" in {
13        val stove = new Stove(aaaaa)
14        val listOfPizza = List(new Pizza, new Pizza, new Pizza)
15        stove += listOfPizza
16        stove.next().length must_== bbbbb
17        stove.next() must beEmpty
18      }
19
20      "... (Test 2)" in {
21        val stove = new Stove(ccccc)
22        val listOfPizza = List(new Pizza, new Pizza, new Pizza)
23        stove += listOfPizza
24        stove.next().length must_== ddddd
25        stove.next().length must_== eeeee
26        stove.next() must beEmpty
27      }
28
29      "returns pizza in a correct way" in {
30        prop { (capacity: Int, noOfP: Int) =>
31
32          var capa = Math.abs(capacity % 10)
33          if (capa == 0) capa = 1
34          var noOfPizza = Math.abs(noOfP % 50)
35          if (noOfPizza == 0) noOfPizza = 25
36
37          val stove = new Stove(capa)
38          stove += List.fill(noOfPizza)(new Pizza)
39
40          var returnedPizza = 0
41          var noOfNext = -1
42          var nextPizza: Int = 0
43          do {
44            nextPizza = stove.next().length
45            noOfNext += 1
46            returnedPizza += nextPizza
47          } while (nextPizza != 0)
48
49          returnedPizza must_== noOfPizza
50          noOfNext must_== (noOfPizza / capa +
51              (if (noOfPizza % capa > 0) 1 else 0))
52        }
53      }
54    }
55  }
```

Beantworten Sie in Bezug auf obigen Code die folgenden Fragen.

(a) Welche Werte müssen die Variablen

i. `aaaaa =` (1)

i. _____

ii. `bbbbb =` (1)

ii. _____

iii. `ccccc =` (1)

iii. _____

iv. `ddddd =` (1)

iv. _____

v. `eeeee =` (1)

v. _____

annehmen, damit Test 1 und Test 2 erfolgreich sind?

(b) Beantworten Sie die folgenden Fragen bezüglich des Tests `returns pizza in a correct way`.

i. Was bedeutet der Code in der Zeile 30? (2)

ii. Was berechnet der Code in den Zeilen 40 bis 47? (3)

Aufgabe 3 (12 Punkte)

(a) Was sind Regressiontests? wie werden Sie üblicherweise durchgeführt und warum? (6)

(b) Was ist szenariobasiertes Testen? Geben Sie ein Beispiel an. (6)

Aufgabe 4 (20 Punkte)

(a) Erklären Sie in eigenen Worten das Abstract Factory Pattern. (10)

(b) Geben Sie ein Beispiel für das Abstract Factory Pattern in Form von Scala Code an. (10)

Aufgabe 5 (9 Punkte)

Nennen Sie 3 von Lehmans Gesetzen und erklären Sie diese kurz.

Aufgabe 6 (11 Punkte)

(a) Welche Arten Wiederverwendung von COTS-Produkten gibt es? (4)

(b) Nennen Sie drei Probleme die bei COTS-Wiederverwendung auftreten können. (3)

(c) Was ist COTS-Integration? (4)

Aufgabe 7 (6 Punkte)

(a) Was ist eine Serviceorientierte Architektur? (3)

(b) Was ist WSDL? (3)

Aufgabe 8 (12 Punkte)

(a) Erklären Sie in eigenen Worten was *ressourcenbasiert* im Zusammenhang mit REST bedeutet. (4)

(b) Ein REST-Service für eine Pizzawebseite antwortet auf einen GET-Request mit der URL `http://pizzainitial.herokuapp.com/api/user/1` mit folgendem JSON:

```
{
  user: {
    id: 1,
    name: "Hugo"
  },
  links: [
    {
      rel: "self",
      href: "http://pizzainitial.herokuapp.com/api/user/1",
      method: "GET"
    },
    {
      rel: "remove",
      href: "http://pizzainitial.herokuapp.com/api/user/1",
      method: "DELETE"
    }
  ]
}
```

i. Welche Attribute hat die zurückgegebene Ressource `Person`? (2)

ii. Warum werden Links in die Antwort eingebettet? (3)

iii. Wie genau kann Hugo gelöscht werden? (3)