Übungsblatt 3

Ausgabe: 10.12.2021 Besprechung: 20.-23.12.2021 7,5 Punkte

Bitte laden Sie die Übungsbeispiele rechtzeitig am Tag vor der Übungsstunde bis spätestens 23:55 als Zip Datei (`<Matrikelnummer>_<Name>_SE1_ÜB3`) in Moodle hoch und tragen Sie Ihre Kreuze in ZEUS ein. Die Zip Datei muss alle gelösten Beispiele und deren Präsentationen beinhalten. Da es in ZEUS keine halben Punkte gibt, wurde die Anzahl der Punkte für ZEUS verdoppelt → laut ZEUS können Sie für diese Übung 15 Punkte bekommen. Für die Übung selbst werden aber die am Übungsblatt angegebenen Punkte gezählt → es können maximal 7,5Punkte erreicht werden (= Anzahl der Punkte in ZEUS / 2). Sollten Sie Fragen zur Übung haben oder Unklarheiten auftreten, nutzen Sie bitte das Moodle-Forum.

Generelle Anmerkung: Benutzen Sie für die Implementierungen dieser Übung das im Moodle bereitgestellte Projekt. Vermerken Sie bitte Ihren **Namen** und Ihre **Matrikelnummer** als Kommentar in all Ihren Klassen. Weiters werden in dieser und den folgenden Einheiten die Tools JUnit zum automatisierten Testen[1], JaCoCo[2] zur Messung der Code Coverage verwendet. Es wird empfohlen sich vor Beginn der Übung mit den Tools auseinander zu setzen. Eine Einführung zu JUnit5 finden Sie z.B. unter folgendem Link[3]. **Hinweis:** Mit dem Befehl `mvn clean test` werden die Tests für das gesamte Projekt ausgeführt und der Coverage Bericht erstellt.

3.1 Äquivalenzklassenzerlegung und Grenzwertanalyse (2 Punkte)

Gegeben ist folgender Methodenkopf:

Listing 1: Codebeispiel

```
1 public RiskLevel getEstimatedRisk(int kW) throws
    IllegalArgumentException{...}
```

Die Methode ist wie folgt spezifiziert: Der Eingabeparameter `kW` beschreibt die Motorleistung eines Autos in Kilowatt. Abhängig von dieser Leistung berechnet die Methode das Risko eines selbstverschuldeten Verkehrsunfalls. Bei einer Motorleistung von 1 bis 56 kW wird das Risiko auf `NIEDRIG` geschätzt. Bei einer höheren Leistung bis inklusive 96 kW steigt das Risiko auf `MODERAT`. Für eine Motorleistung bis 150 kW soll die Methode das Risiko auf `HOCH` schätzen. Weist das Fahrzeug eine höhere Leistung auf, wird das Risiko auf `SEHR HOCH` geschätzt. Sie können davon ausgehen, dass kein Fahrzeug mehr als 2000 kW Motorleistung aufweist. Sollten die angegebenen kW nicht der Spezifikation entsprechen, wirft die Methode eine InvalidBudgetException.

Bearbeiten Sie folgende Aufgaben und treffen/dokumentieren Sie, sofern notwendig, sinnvolle Annahmen.

1. Bestimmen Sie alle **gültigen** Äquivalenzklassen für die Variable `kW`.
2. Bestimmen Sie alle **ungültigen** Äquivalenzklassen für die Variable `kW`.
3. Wenden Sie die grundlegende Idee der **Grenzwertanalyse** auf alle gefundenen Klassen an, um **Testfälle** abzuleiten. Geben Sie alle gefundenen Testfälle an.

[1] `https://junit.org/junit5/`
[2] `http://www.eclemma.org/jacoco/`
[3] `http://www.vogella.com/tutorials/JUnit/article.html` (Speziell ab Kapitel 4)

3.2 Kontrollflussorientiertes Testen (2 Punkte)

Listing 2: Codebeispiel

```
1  public static int checkRelation(int x, int y) {
2      int u, v, w;
3      if (x*2 < y) {
4          u = y;
5          w = 2;
6      } else {
7          u = x;
8          w = 4;
9          }
10     if (x + y >= 25) {
11         v = u + y;
12     } else {
13         v = u + x / w;
14         }
15     return v;
16 }
```

Gegeben ist Listing 2. Bearbeiten Sie folgende Aufgaben:

1. Gegeben sind folgende Tests der Struktur (x,y): {(0,0); (8,17)}

   (a) Kann hier ein Anweisungsüberdeckungstest durchgeführt werden?

   (b) Kann hier ein Zweigüberdeckungstest durchgeführt werden?

   (c) Kann hier ein Pfadüberdeckungstest durchgeführt werden?

   Begründen Sie Ihre Antwort falls die jeweilge Überdeckung erreicht werden kann, andernfalls geben Sie an welche zusätzlichen Tests notwendig wären.

2. Implementieren Sie für folgende Aufgaben alle Testfälle mit JUnit (`at.aau.ue3.bsp2.RelationCheckerTest`) und messen Sie nach jedem Schritt die Coverage mit JaCoCo:

   (a) Finden Sie eine minimale Anzahl konkreter Testfälle damit ein Anweisungsüberdeckungstest durchgeführt werden kann.

   (b) Finden Sie eine minimale Anzahl konkreter Testfälle damit ein Zweigüberdeckungstest durchgeführt werden kann.

   (c) Finden Sie eine minimale Anzahl konkreter Testfälle damit ein Pfadüberdeckungstest durchgeführt werden kann.

3.3 Bedingungsüberdeckungstests (1,5 Punkte)

Listing 3: Pseudocode

```
1 int x = ...;
2 boolean y = ...;
3 ...
4 if((x<0 && x>-12) || ((x<10) && y) || (x==77 && !y)){
5 ...
6 }
```

Gegeben ist Listing 3. Bearbeiten Sie folgende Aufgaben:

1. Wieviele Testfälle reichen bei der Bedingungen aus Listing 3 minimal für eine Einfachbedingungsüberdeckung aus? Geben Sie diese Testfälle an.
2. Zeigen Sie anhand von Testfällen und der in Listing 3 gegebenen Bedingung, warum die Einfachbedingungsüberdeckung schwächer als die Zweigüberdeckung ist.
3. Führen Sie einen Mehrfachbedingungsüberdeckungstest durch. Wieviele Testfälle können in diesem Beispiel real getestet werden? Geben Sie Beispiele für diese Testfälle an.
4. Wieviele Testfälle sind für eine minimale Mehrfachbedingungsüberdeckung durchzuführen? Geben Sie die Testfälle an.

3.4 Reverse Engineering (2 Punkte)

Laden Sie die Datei 'gamestore.zip' herunter und importieren Sie das Projekt. Machen Sie sich mit dem Code vertraut indem sie das First Contact Pattern 'Read the code in one hour' anwenden. Reverse engineeren Sie den Code daraufhin in ein UML Klassendiagramm.