RUPRECHT–KARLS–UNIVERSITÄT HEIDELBERG
INSTITUT FÜR INFORMATIK – SOFTWARE ENGINEERING

Übungen zu „Einführung in Software Engineering“ (WS 19/20)
Prof. Dr. Barbara Paech, Astrid Rohmann, Anja Kleebaum
http://se.ifi.uni-heidelberg.de

Übungsblatt 9 (17.12.2019)

Black-Box Test, Komponententests(JUnit), Vorbereitung Inspektion

In dieser Übung:

- ✓ Üben Sie Operationstest durch Betrachtung von Äquivalenzklassen.
- ✓ Programmieren Sie Komponententests (Unit Test) mit JUnit.
- ✓ Bereiten Sie das Thema Inspektion vor.

Bitte zur Abgabe:
Aufgrund der Vorlesungsfreien Zeit ist die offizielle Abgabe der Lösungen erst am 07.01.2020. Wir bitten Sie Ihre Lösungen **vor dem 07.01.2020** abzugeben, sodass die Tutoren die Möglichkeit haben, diese vor der Testatabnahme anzuschauen.

Aufgabe 9.1: Analyse der bestehenden Movie Manager App (Teil 13): Black-Box-Test: Äquivalenzklassen

Präsenz: Nein	**Punkte: 12**	**Team:** Ja	**Testat**

Betrachten Sie die Spezifikation der Operation `calculateOverallRating` der Klasse `RatingUtils` zur Berechnung der Gesamtbewertung für einen Film unter Berücksichtigung der Bewertungen der verlinkten SchauspielerInnen. Die Spezifikation finden Sie im Jira-Projekt unter MMAPSTUD-44 in den Regeln (R1) bis (R3) (die übrigen Regeln dienen nur der Darstellung).

Verwenden Sie bei der Bearbeitung der Aufgabe die folgenden abkürzenden Notationen:

- Bewertung eines Filmes m: `m.rating`
- Film hat keine Bewertung: `m.rating = -1`
- Verlinkte SchauspielerInnen eines Filmes: `m.performers`
- Bewertung eineR SchauspielerIn p: `p.rating`
- SchauspielerIn hat keine Bewertung: `p.rating = -1`

1) Geben Sie für die Operation `calculateOverallRating` die **gültigen und ungültigen Eingabeäquivalenzklassen** an. Dabei ist der Film als Eingabe zu betrachten. Ungültig bedeutet, dass eine Eingabe nicht den richtigen Datentyp hat. Identifizieren und beschreiben Sie weiterhin für die gültigen Eingaben der Operation weitere **Äquivalenzklassen**, **um wichtige Ausgaben und Ausnahmen** abzudecken. Beachten Sie, dass Ihre Äquivalenzklassen insbesondere die Grenzwerte berücksichtigen.
2) Skizzieren Sie für die Operation alle **Testfälle**, die durch **sinnvolle Kombinationen der Äquivalenzklassen** entstehen, in einer Tabelle, d.h. geben Sie für jeden Testfall die Äquivalenzklassen der Eingaben an und beschrieben Sie das erwartete Ergebnis. Nummerieren Sie diese Testfälle und verwenden Sie diese Nummern im Folgenden.

3) Wählen Sie aus der Tabelle aus 9.1.2 die Testfälle aus, die zusammen eine **minimale Testfallmenge** („jede gültige Äquivalenzlasse in mindestens einer Kombination“) bilden. Verwenden Sie dabei die in 9.1.2 vergebene Nummer.

4) Die Excel-Datei *09-ComponentTestcases.xlsx* (Datei zu finden in Moodle) beinhaltet bereits die Spezifikation zweier logischer und passender konkreter Testfälle zur Operation `calculateOverallRating`. Wählen Sie mindestens 4 Testfälle aus 9.1.2. aus (verschieden von dem vorgegebenen), sodass Sie mit mindestens einem Testfall die Regel (R2) mit mehr als eineR SchauspielerIn abdecken, und spezifizieren Sie diese analog zu den vorgegebenen Testfällen. Beschreiben Sie jeweils einen **logischen Testfall** und einen **konkreten Testfall**. Verwenden Sie dabei die in 9.1.2 vergebene Nummer. (Die Nummern der vorgegebenen Testfälle dürfen dabei an die eigene Nummerierung angepasst werden.)

Ergebnis:
Speichern Sie bitte eine .zip-Datei bis **Dienstag 07.01.2020 um 10.00 Uhr** in Moodle bestehend aus:

- PDF-Dokument mit der Angabe der Äquivalenzklassen, der Kombination der Äquivalenzklassen zu Testfällen sowie der minimalen Testfallmenge
- Ausgefüllte Excel-Datei **09-ComponentTestcases.xlsx** mit spezifizierten logischen und konkreten Testfällen

Testumgebung:
Für die nachfolgende **Aufgabe 9.2: „Analyse der bestehenden Movie Manager App (Teil 14): JUnit-Tests“** benötigen Sie eine Testumgebung. Diese Testumgebung liefert Ihnen das Testobjekt sowie einen Testrahmen. Der Testrahmen besteht aus einer JUnit Klasse mit den Testfällen sowie einer Laufzeitumgebung.

Eine bereits implementierte und lauffähige Testumgebung für die Aufgabe 9.2 finden Sie im aktualisierten Projekt des Movie Managers in Confluence.

JUnit ist in dieser Testumgebung **bereits aktiviert**. Die Testumgebung besteht aus:

- **Testobjekt:** `de.moviemanager.util.RatingUtils` (im Ordner `main`).
- **Testfälle:** `de.moviemanager.util.RatingUtilsTest` (im Ordner `test`).
- **Laufzeitumgebung:** Alle anderen notwendigen Klassen im Ordner `main`.

Aufgabe 9.2: Analyse der bestehenden Movie Manager App (Teil 14): JUnit-Tests

Präsenz: Ja	**Punkte:** 8	**Team:** Nein	**Testat**

In dieser Aufgabe testen Sie die Operation `calculateOverallRating` der Klasse `RatingUtils`. Die Operation kennen Sie bereits aus Aufgabe 9.1.
Verwenden Sie für diese Aufgabe die aktualisierte Version des Movie Managers in Confluence.

Für die Operation `calculateOverallRating` sind bereits zwei Testfälle mit JUnit implementiert worden (siehe Klasse `RatingUtilsTest` im Ordner `test` im Package `de.moviemanager.util`. Machen Sie sich mit den Testfällen vertraut und führen Sie sie anschließend in JUnit (Run ‚RatingUtilsTest‘) aus.

1) Testen Sie die Operation `calculateOverallRating` der Klasse `RatingUtils`. Implementieren Sie dafür **2 von Ihnen definierte konkrete Testfälle aus Aufgabe 9.1.4** in der Klasse `RatingUtilsTest`. Geben Sie den Testfallklassen sprechende Namen, d.h.

benennen Sie Ihre Operationen entsprechend, z.B. `public void testCalculateOverallRatingWithUnratedMovieAndNoPerformers()`. Beachten Sie, dass Sie bei der Wahl der Testfälle insbesondere die Regel (R2) mit mehr als eineR SchauspielerIn abdecken.

2) Führen Sie die zwei Testfälle in Android Studio mit den Eingaben aus den konkreten Testfällen aus Aufgabe 9.1.4. aus, und protokollieren Sie die tatsächlichen Ergebnisse in der konkreten Testfallspezifikation der Vorlage `09-ComponentTestcases.xlsx` aus Aufgabe 9.1.4.

   Hinweis: Sollte die Ausführung Ihrer implementierten JUnit-Tests fehlschlagen, protokollieren Sie die Fehlermeldung und suchen nach der Ursache. Handelt es sich dabei um einen Fehler in der Implementierung Ihres Tests, so beheben Sie diesen. Protokollieren Sie in jedem Fall die Fehlermeldung und die Ursache.

Ergebnis:
Speichern Sie bitte das Ergebnis als .zip-Datei bis **Dienstag 07.01.2020 um 10.00 Uhr** in Moodle bestehend aus:

- 1x Android Studio-Projekt **09-unit-moviemanager.zip** mit den implementierten Testfällen
- Ausgefüllte Excel-Datei **09-ComponentTestcases.xlsx** mit Ihren konkreten Komponententestfällen

*Hinweis: In der Abgabe können Sie zur Reduzierung der Speichergröße den Ordner **build** (Unterordner von app) löschen, da dieser aus den Dateien in src erneut erzeugt werden kann.*

Aufgabe 9.3: Vorbereitung – Inspektion

Präsenz: Nein	**Punkte:** 8	**Team:** Nein	

Bereiten Sie sich auf die Inhalte der übernächsten Vorlesung vor, indem Sie Kapitel 13, Abschnitte 13.4.5 – 13.6.4 aus dem Buch von Ludewig und Lichter zur Inspektion lesen (Literatur – Buchkapitel Ludewig und Lichter in **Dateien und Materialien** in Confluence).

1) Halten Sie anschließend die Inhalte zu den folgenden Punkten anhand der Informationen aus dem Kapitel in einem PDF-Dokument fest:
   - Welche Arten der Inspektion gibt es?
   - Wie läuft der technische Review ab? Welche Rollen sind beteiligt? Welche wichtigen Regeln gibt es? Was sind typische Probleme und Abhilfen?

Ergebnis:
Speichern Sie bitte Ihr PDF-Dokument bis **Dienstag 07.01.2020 um 10.00 Uhr** in Moodle.

Wir wünschen Ihnen und Ihrer Familie frohe Festtage und für das neue Jahr Gesundheit, Glück und Erfolg!