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

Übungsblatt 7 (03.12.2019)

Klassendiagramm UI, Movie Manager Programmierung, Sequenzdiagramm, Vorbereitung Zustandsdiagramme

In dieser Übung:

- ✓ Erstellen Sie ein Klassendiagramm für den UI-Code des *Movie Managers.*
- ✓ Programmieren Sie ein Kontext-Menü für den *Movie Manager*.
- ✓ Ergänzen Sie ein Sequenzdiagramm zu einer Systemfunktion des *Movie Managers*.
- ✓ Bereiten Sie Zustandsdiagramme vor.

Aufgabe 7.1: Analyse der bestehenden Movie Manager App (Teil 8): Klassendiagramm für den UI-Code des Movie Managers

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

In dieser Aufgabe ergänzen Sie das Klassendiagramm der Vorlage *07-MovieManager-ClassDiagram.png* (Datei zu finden in Moodle), das eine Übersicht über den UI-Code der Detail-Ansichten enthält, um einen Teil des UI-Code des *Movie Managers*. Schauen Sie sich den Quelltext des *Movie Managers* an und ergänzen Sie das Klassendiagramm passend zum Quelltext. Sie können das Klassendiagramm entweder per Hand ergänzen und dann einscannen oder Sie verwenden ein Zeichenwerkzeug ihrer Wahl, z.B. Paint, Visio, draw.io o.ä.

1) Betrachten Sie alle Klassen aus dem Paket `moviemanager.ui.detail` und dessen Unterpaketen `modifiable` und `modifications`. Zeichnen Sie alle Klassen aus dem Paket `moviemanager.ui.detail` (ohne Klassen aus den Unterpaketen, da diese bereits vorhanden sind, und ohne die Klasse `LinkedPortrayableList`, die zu detailliert ist und nicht zur Übersicht beiträgt) in die Vorlage des Klassendiagramms mit allen Attributen und Operationen ein. Im JIRA-Projekt finden Sie für die obigen Klassen jeweils Issues vom Typ „Class“. Als Arbeitserleichterung können Sie die Attribute und Operationen aus den „Class“-Issues kopieren und in Ihr Klassendiagramm einfügen.
2) Erkennen Sie aus dem Quellcode die Assoziationen zwischen den ergänzten Klassen und den bereits vorhandenen Klassen und zeichnen Sie diese in das Klassendiagramm ein. Benennen Sie die Assoziationen und ergänzen Sie die Multiplizitäten. Ergänzen Sie alle Vererbungsstrukturen der Tiefe 1.
3) Beschreiben Sie in einem PDF-Dokument in 1-2 Sätzen pro Assoziation und Vererbung im Klassendiagramm wie die Klassen zusammenarbeiten und womit Sie die Multiplizität begründen.
4) Ihr Klassendiagramm enthält nun Klassen der Datenschicht, der Oberflächenschicht und der Frameworkschicht. Färben Sie die Klassen innerhalb des Klassendiagramms **rot** für Oberflächenklassen, **blau** für Klassen des Android-Frameworks und der Java Standardbibliothek, und **gelb** für Hilfsklassen des Movie-Manager-Frameworks.

Ergebnis:
Speichern Sie bitte das Ergebnis als .zip-Datei bis **Montag 09.12.2019 um 10.00 Uhr** in Moodle bestehend aus:

- 1x PNG-Datei des eingefärbten Klassendiagramms mit den ergänzten Klassen, Attributen, Operationen und Assoziationen
- 1x PDF-Datei mit Beschreibung der Assoziationen und Vererbungsbeziehungen

Aufgabe 7.2: Analyse der bestehenden Movie Manager App (Teil 9): Programmierung Kontext-Menü

Präsenz: Nein	**Punkte:** 10	**Team:** Nein	**Testat**

In dieser Aufgabe implementieren Sie eine **neue Sortierung** für die Movie Master Liste, die Filme danach sortiert, ob und wann diese zuletzt angesehen wurde. Verwenden Sie als Grundlage für die Implementierung die in Confluence bereitgestellte Version des Movie Managers. Betrachten Sie dazu auch die Systemfunktion *Filter and Sort Movies* in Jira.
Implementieren Sie für die Sortierung nach `watchDate` eine neue Klasse, die von `Categorizer` erbt, von der anschließend im `MovieMasterFragment` eine neue Instanz erstellt und zu den bestehenden Sortiermöglichkeiten `orders` hinzugefügt wird. Diese neue Klasse stellt Operationen zur Verfügung, die vom Sortierkriterium abhängen und wichtig für die Sortierung sind. Gehen Sie dabei wie folgt vor:

1) Erstellen Sie im Paket `de.moviemanager.ui.masterlist.categorizer` eine neue Klasse `Watched`, die von der abstrakten Klasse `Categorizer<String, Movie>` (im gleichen Paket) erbt. Orientieren Sie sich dabei an den bereits vorhandenen Klassen `Alphabetical`, `Numeric` und `Rated`. Erstellen Sie einen geeigneten Konstruktor und überschreiben Sie die folgenden Operationen:
   - `getCategoryNameFor()` zum Bestimmen der Kategorie, in die der Film einsortiert wird.
   - `createHeader()` zum Erstellen eines `HeaderElement`s einer Kategorie.
   - `createContent()` zum Erstellen des `ContentElement`s, das den Film in der Liste darstellt. Ein `ContentElement` erhält im Konstruktor den Namen des darzustellenden Objekts und die Meta-Informationen, die unterhalb des Namens dargestellt werden, jeweils als `String`. Als Meta-Information sollte hier das Datum angezeigt werden, an dem der Film zuletzt gesehen wurde.
   - `createDivider()` zum Erstellen eines `DividerElement`s.
   - `compareCategories()` zum Sortieren der Kategorien.
   - `compareContent()` zum Sortieren der Filme innerhalb einer Kategorie.

   Hinweis: In der Klasse de.util.DateUtils finden Sie Operationen zur Umwandlung von Date in String und umgekehrt.

2) Erweitern Sie in der Klasse `MovieMasterFragment` im Paket `de.moviemanager.ui.masterfragments` die Operation `createOrders()`, indem Sie über `addOrder()` eine neue Sortierung zu den bereits bestehenden hinzufügen. Diese Operation erwartet als Parameter den anzuzeigenden Namen Ihrer Sortierung, eine Instanz Ihrer neuen Sortierung `Watched` und eine Filter-Logik, wofür Sie `nameContainsInput` wie bei den anderen Sortierungen verwenden können.

3) Führen Sie eine statische Codeprüfung mit SonarLint durch und beheben Sie die gefundenen Fehler.

4) Überlegen Sie sich, welche Fälle wichtig sind, um die neue Funktionalität ausführlich zu testen und erfassen Sie diese in einem PDF-Dokument. Testen Sie anschließend Ihre geänderte Version der Movie Manager App, indem Sie diese auf dem Emulator starten und die

beschriebenen Testfälle ausführen. Dokumentieren und beheben Sie alle Fehler Ihrer Programmierung, die Sie beim Testen gefunden haben.

Orientieren Sie sich bei Ihrer Implementierung auch immer an den bereits vorhandenen Klassen und Operationen sowie der Verwendung von Android.

Ergebnis:
Speichern Sie Ihre Ergebnisse bitte als **.zip-Datei** bis **Montag 09.12.2019 um 10.00 Uhr** in Moodle, bestehend aus:

- Ihrem exportierten Android Studio-Projekt
- PDF-Dokument mit Ihren Testfällen und ggf. gefundenen Fehlern.

Aufgabe 7.3: Analyse der bestehenden Movie Manager App (Teil 10): Sequenzdiagramm

Präsenz: Nein	**Punkte:** 4	**Team:** Nein	**Testat**

In dieser Aufgabe ergänzen Sie das noch unvollständige Sequenzdiagramm zur Systemfunktion Unlink Performer, dessen ersten Teil Sie in der Datei 07-MovieManager-SequenceDiagram.png finden. Verwenden Sie hierzu den bereitgestellten Code der Movie Manager App. Gehen Sie dabei wie folgt vor:

1) Machen Sie sich mit dem Code vertraut, der die Systemfunktion Unlink Performer umsetzt. Die Funktion wird aufgerufen, indem man im Movie Detail Edit Mode in der Performer-Liste eines Filmes einen Eintrag zur Seite schiebt und das erscheinende Unlink-Symbol antippt. Die Programmierung der Performer-Liste befindet sich in der Klasse `LinkedPortrayableList`. In der Methode `onUnlinkSelected(ViewHolder viewHolder)` ist das Unlinken implementiert. Für die Funktionalität, dass man einen Performer in der Liste zur Seite schieben kann, sodass das Unlink-Symbol erscheint, ist die Klasse `SwipeController` zuständig.

2) Ergänzen Sie nun das unvollständige Sequenzdiagramm in der Vorlage mithilfe des betrachteten Codes. Beachten Sie dabei, dass in der Vorlage die Datenklassen **grün,** die Oberflächenklassen **rot**, Klassen des Android-Frameworks und der Java Standardbibliothek **blau** und Hilfsklassen des Movie-Manager-Frameworks **gelb** gefärbt sind. Sie können das Sequenzdiagramm entweder per Hand ergänzen und dann einscannen oder Sie verwenden ein Zeichenwerkzeug ihrer Wahl, z.B. Paint, Visio, draw.io o.ä. Orientieren Sie sich bei der Erstellung des Sequenzdiagramms an den vorhandenen Sequenzdiagrammen im JIRA-Projekt.

Ergebnis:
Bitte speichern Sie Ihre Ergebnisse als PDF- oder PNG-Datei bis **Montag 09.12.2019 um 10.00 Uhr** in Moodle.

Aufgabe 7.4: Vorbereitung – Zustandsdiagramme

Präsenz: Nein	**Punkte:** 6	**Team:** Nein	

1) Bereiten Sie sich auf die nächsten Inhalte der Vorlesung vor, indem Sie sich die Konzepte und Notationen der Zustandsdiagramme erarbeiten. Verwenden Sie dazu die Materialien des Buches UML@Classroom. Die Materialien sind unter dem Link http://www.uml.ac.at/de/lernen abrufbar. Eine identische Version der Folien „Zustandsdiagramme“ mit gedrucktem Sprechertext finden Sie unter *Literatur – UML@Classroom* in **Dateien und Materialien** in Confluence.

Lesen Sie die Folien 1 bis einschließlich Folie 36 des Foliensatzes „Zustandsdiagramme“.

2) Halten Sie anschließend die **Inhalte zu den folgenden 3 Punkten anhand der Informationen aus dem Foliensatz in einem** PDF-Dokument fest:
   - Was sind die wichtigsten Elemente eines Zustandsdiagramms?
   - Welche 3 Arten von internen Transitionen gibt es? Worin unterscheiden sie sich?
   - Welche Konzepte erlauben die Beschreibung eines komplexen Zustands?

Ergebnis:
Speichern Sie bitte Ihr PDF-Dokument bis **Montag 09.12.2019 um 10.00 Uhr** in Moodle.