Software-Qualitätssicherung

Übung 4 - Abgabe

Ihre Daten – Bitte füllen Sie alle Felder aus:

Nachname, Vorname:	Hagn, Maximilian
Matrikelnummer:	11808237
Studienkennzahl:	UE 033 534
E-Mail Adresse:	e11808237@student.tuwien.ac.at

Inhaltsverzeichnis

1 Vorgehensweise zur Qualitätssteigerung

1.1 Tools

1.1.1 Checkstyle

Für die Analyse mit Checkstyle wurde das IntelliJ Tool „CheckStyle-IDEA“ von Jamie Shiell verwendet. In diesem Tool kann die Auswahl zwischen den zwei Regelwerken „Sun Checks“ und „Google Checks“ getroffen werden. Im Zuge der Analyse habe ich beide Regelwerke ausprobiert und mich schlussendlich dazu entschieden „Sun Checks“ zu verwendet. Bei der Analyse der beiden Regelwerke fällt auf, dass die Regeln von Google strikter sind und unter anderem auch Regeln für die Einrückung der Zeilen verwendet werden. Des Weiteren kann festgestellt werden, dass bei „Google Checks“ alle Regelverstöße als Warnungen deklariert sind, wobei bei „Sun Checks“ alle Regelverstöße als Fehler markiert sind. Bei der Analyse des Projektes mit dem Regelwerk von „Sun Checks“ fallen vor allem folgende Fehler auf:

- Es fehlt ein Javadoc-Kommentar.
- Stern-Importe sollten vermieden werden.
- Zeile ist x Zeichen lang. (Obergrenze ist 80)
- Der Parameter x sollte als 'final' deklariert werden.
- Klasse x sieht wie zur Erweiterung entworfen aus.
- Die Variable x verbirgt ein Feld.
- [Logischer Operator] sollte in einer neuen Zeile stehen.
- Die magische Zahl x sollte als Konstante definiert werden.
- In der switch-Anweisung fehlt der default-Zweig.

1.1.2 SonarLint

Für die Analyse mit SonarLint wurde das IntelliJ Tool „SonarLint“ von SonarSource verwendet. Wird dieses Tool verwendet, um das Projekt zu überprüfen, fällt auf, dass alle gefundenen Regelverstöße als Minor, Major oder Critical deklariert werden. Des Weiteren fällt auf, dass die Analyse nicht so viele Fehler aufzeigt wie Checkstyle. Teilweise werden ähnliche Fehler gefunden, jedoch gibt es auch andere Fehler die eher auf technische Details wie zum Beispiel Generizität oder Lamda-Ausdrücke abzielen. Bei SonarLint werden unter anderem keine Fehler für fehlende Javadoc-Kommentare aufgezeigt. Es fallen vor allem folgende Fehler auf:

- Ersetze if-then-else Anweisung durch eine einzige Anweisung.

- In der switch-Anweisung fehlt der default-Zweig.
- Die Variable x sollte als 'static final constant' oder 'non-public' deklariert werden.
- Verwende 'isEmpty()', um zu überprüfen, ob eine Liste leer ist oder nicht.
- Ersetze 'System.out' durch einen Logger.
- Entferne das ungenutzte Import-Statement.

1.2 Strategie

Um die Verwendung der zwei vorgestellten Richtlinien-Prüfer in diesem Projekt einzuführen sollte zu Beginn eine automatische Überprüfung der Richtlinien festgelegt werden. Zum Beispiel könnten die beiden Tools in den Build-Prozess von Maven eingebunden werden, wodurch eine Version nur noch fehlerfrei gebaut werden kann, wenn alle Richtlinien erfüllt sind. Dafür ist es wichtig, dass die Tools ausgewählt werden und das sich die Programmierer auf ein Regelwerk einigen.
Nachdem die Tools und die Regelwerke ausgewählt wurden, muss überlegt werden, wie der aktuelle Stand verändert werden kann, um diesen Regeln zu entsprechen. Die entwickelnden Personen könnten dafür jeweils immer eine Klasse verbessern. Zuerst muss für jede Klasse analysiert werden, welche Fehler vorliegen. Des Weiteren muss sichergestellt werden, dass die Testabdeckung der jeweiligen Klasse hoch genug ist, um etwaige Fehler bei Veränderungen aufzudecken. Nachdem sichergestellt ist, dass die beschlossene Testabdeckung dieser Klasse ausreichend ist, müssen die entwickelten oder bereits bestehenden Testfälle ausgeführt werden. Die Testfälle sollten alle erfolgreich durchlaufen. Anschließend kann mit dem Refactoring begonnen werden. Tools wie SonarLint helfen den Programmiererinnen und Programmierern dabei Fehler aufzufinden und geben teilweise Vorschläge wie diese behoben werden können. Nach der Verbesserung eines Fehlers sollten alle Testfälle erneut ausgeführt werden, um zu überprüfen, ob sich durch das Refactoring keine neuen Fehler eingeschlichen haben. Sind alle Tests erfolgreich, kann der behobene Fehler in einer Versionsverwaltung festgehalten werden. Schlagen die Testfälle fehl, müssen die Fehler ausgebessert werden, bis schlussendlich alle Testfälle erfolgreich sind. Dieser Prozess wird wiederholt, bis die ausgewählten Tools zur Regel-Überprüfung in keiner Klasse Fehler erkennen.
Des Weiteren muss festgelegt werden, wie bei neuen Programmteilen mit den neuen Richtlinien umgegangen wird. Wie bereits erwähnt, können die beschriebenen Tools in den Build-Prozess eingebaut werden. Durch die automatische Überprüfung kann es somit nicht mehr vor kommen, dass Code der gegen die Richtlinien verstößt, akzeptiert wird. Des Weiteren sollten die entwickelnden Personen mit den Regelwerken vertraut gemacht werden.

2 Refactoring Testbericht

Um zu überprüfen, ob die Anforderungen auch nach dem Refactoring noch eingehalten werden, müssen zuerst Anforderungen ausgewählt werden, die auf die geänderten Funktionen eingehen. Im Zuge des Refactoring-Prozesses wurden vor allem die Benutzerschnittstellen der Filme und Kunden Verwaltung verändert. Des Weiteren wurde die Altersfreigabe und die Genres durch neue Objekte ersetzt. Schlussendlich wurden ebenfalls Funktionen, die zur Berechnung des Preises benötigt werden in neue Klassen verschoben. Folgende Anforderungen müssen somit erneut geprüft werden:

- 2. Anforderung: Kunden Stammdaten
- 3. Anforderung: Film Stammdaten
- 4. Anforderung: Ausleihvorgang
- 5. Anforderung: Buchungsbeleg
- 8. Anforderung: Kostenberechnung nach Tagen
- 9. Anforderung: Preismodifikation nach Genre
- 12. Anforderung: Prüfung der Altersfreigabe
- 13. Anforderung: Benutzerfreundlichkeit - Applikation
- 21. Anforderung: Erweiterbarkeit

2.1 Akzeptanztest

2.1.1 Kunden Stammdaten

Um die Erstellung eines Kunden zu überprüfen, wurde die Applikation gestartet und auf das Fenster Kundenverwaltung gewechselt. Zuerst wurde überprüft, ob ein Kunde angelegt werden kann. Dafür wurden alle notwenigen Felder mit Daten befüllt und anschließend auf Speichern geklickt. Der Kunde wurde erfolgreich angelegt. Des Weiteren wurde überprüft, ob die Fehlermeldungen korrekt ausgegeben werden. Dafür wurde in jedem Feld eine ungültige Eingabe getätigt und es ist jeweils die korrekte Fehlermeldung zurückgegeben worden. Schlussendlich wurde noch ein Kunde ausgewählt und dessen Daten wurden geändert und anschließend gespeichert. Auch bei diesem Testfall konnten keine Fehler festgestellt werden.

2.1.2 Film Stammdaten

Um die Verwaltung eines Films zu testen, wurde die Applikation gestartet. Anschließend wurde ein neuer Film mit gültigen Daten erstellt, wobei das Rating aus der Datenbank geladen wurde. Der Film wurde nach dem Speichern korrekt in der Tabelle angezeigt. Des Weiteren wurde ein Film ausgewählt und dessen Daten geändert. Auch dieser Test konnte erfolgreich abgeschlossen werden. Des Weiteren kann festgehalten werden, dass bei diesem Test die Anzeigenamen der Altersfreigabe und der Genres genau betrachtet wurden, da an dieser Funktion Änderung vorgenommen wurden.

2.1.3 Ausleihvorgang, Buchungsbeleg

Um diese Anforderung zu testen wurde die Applikation gestartet und das Fenster Ausleihe / Rückgabe geöffnet. Diese Anforderung wird überprüft, da bei dieser Funktion viele getätigte Änderungen ineinander greifen. Der Faktor der Genres, der Ort der Preisberechnung und die Altersfreigabe wurden verändert. Um den Ausleihvorgang zu überprüfen wurden mehrere Kunden ausgewählt und jeweils verschiedene Filme ausgeliehen. Es wurde überprüft, ob die Altersfreigabe den Ausleihvorgang verhindert, ob der Preis pro Tag korrekt angezeigt wird und ob eine Buchung abgeschlossen werden kann. Die Tests wurden erfolgreich durchgeführt.

2.1.4 Kostenberechnung nach Tagen, Preismodifikation nach Genre

Diese Anforderung muss überprüft werden, da die Genres mit Hilfe einer Enumeration implementiert wurden. Des Weiteren wurde die Kostenberechnung in eine neue Klasse verschoben, wodurch auch diese Funktionalität erneut getestet werden muss. Um diese Anforderung zu testen wurde die Applikation gestartet und auf das Fenster Ausleihe / Rückgabe gewechselt. Es wurde ein Kunde, der bereits einen Film ausgeliehen hat, ausgewählt. Der Film wurde zurückgegeben und es wurde auf der Rechnung überprüft, ob der Preis je nach Genre und Ausleihdauer angepasst wurde. Dieser Test wurde erfolgreich bestanden.

2.1.5 Ausleihvorgang, Prüfung der Altersfreigabe

Der Ausleihvorgang musste betrachetet werden, da Änderungen bei der Altersfreigabe angefallen sind. Um die Funktion zu überprüfen, wurde eine junge Kundin ausgewählt, die mehrere Filme mit verschiedenen Altersfreigaben ausgeliehen hat. Bei allen ungültigen Leihen ist korrekterweise eine Fehlermeldung ausgegeben worden. Filme mit FSK 0 / 6 / 12 konnten ausgeliehen werden, da die Kundin zwölf Jahre alt ist.

2.1.6 Benutzerfreundlichkeit - Applikation

Es gab keine Änderungen an der Benutzerschnittstelle, wodurch immer noch alle Funktionen mit drei Mausklicks erreichbar sind. Es wurde jedoch getestet, ob die Altersfreigaben

und Genres in der gleichen Form, wie vor der Änderung angezeigt werden. Dieser Test wurde erfolgreich bestanden.

2.1.7 Erweiterbarkeit

Wie in den vorherigen Abgaben beschrieben, ist diese Anforderung nicht überprüfbar. Es ist jedoch zu erwähnen, dass die Erweiterbarkeit erhöht wurde, indem die Validierungen in eine separate Klasse ausgelagert wurden.