Programmieren 2

Exercise 1

Ziele der Übung

- Maven konfigurieren und JUnit-Dependencies hinzufügen
- Arbeiten mit Git und GitHub
- Schreiben von sinnvoll strukturierten Unit Tests
- Einsatz von OOP-Konzepten: Aufteilung der Features in sinnvolle und testbare Funktionen und Klassen. Anwendung von Encapsulation-Prinzipien etc.
- Coding Style Conventions anwenden

Aufgabenstellung

Im Zuge der Exercise 1 soll ein Grundgerüst für die FHMDb (FH Movie Database) implementiert werden. Dabei sollen gegebene Anforderungen (Requirements) in der Applikation umgesetzt werden und die Applikation mithilfe von Unit Tests ausgiebig getestet werden. Die Tests sollen automatisiert mit Maven ausführbar sein.

Gegeben ist ein JavaFX Template: https://github.com/leonardo1710/fhmdb-template

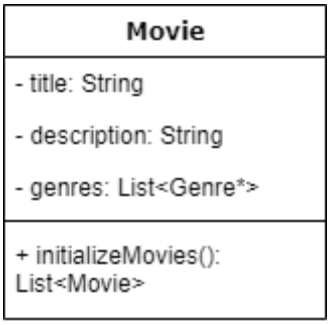
Dieses muss nicht verwendet werden, es kann auch ein neues Projekt mit Maven erstellt werden.

Rahmenbedingungen (K.O Kriterien):

- Das Projekt muss ein GUI bieten (keine Konsolenapp)
- Die Unit Tests müssen mit JUnit umgesetzt werden
- Das Projekt muss auf Maven basieren
- Das Projekt muss auf GitHub verfügbar sein, alle Teammitglieder müssen als Contributors zugeordnet sein

Requirements

Mithilfe der FHMDb Applikation sollen zukünftig Filme (Movies) verwaltet werden können. Ein Film hat vorerst folgende Attribute und Methoden:



* Genres: [`ACTION, ADVENTURE, ANIMATION, BIOGRAPHY, COMEDY, CRIME, DRAMA, DOCUMENTARY, FAMILY, FANTASY, HISTORY, HORROR, MUSICAL, MYSTERY, ROMANCE, SCIENCE_FICTION, SPORT, THRILLER, WAR, WESTERN`]

`initializeMovies()`: erstellt eine statische Liste mit Dummy-Movie Daten

In einem ersten Entwurf, soll die Applikation folgende Features bieten:

Anzeige

Am Screen der Applikation werden alle Filme mit ihrem Titel, der Beschreibung und ihren Genres in einer Liste angezeigt. Weiters bietet die GUI einen Button zum Sortieren, ein Eingabefeld für die

Suche, ein Dropdownmenü (Combobox) zur Auswahl eines Genres, einen Button, der die Filterkriterien anwendet. Es soll auch möglich sein, Filterkriterien wieder abzuwählen bzw. den Filter zu deaktivieren.

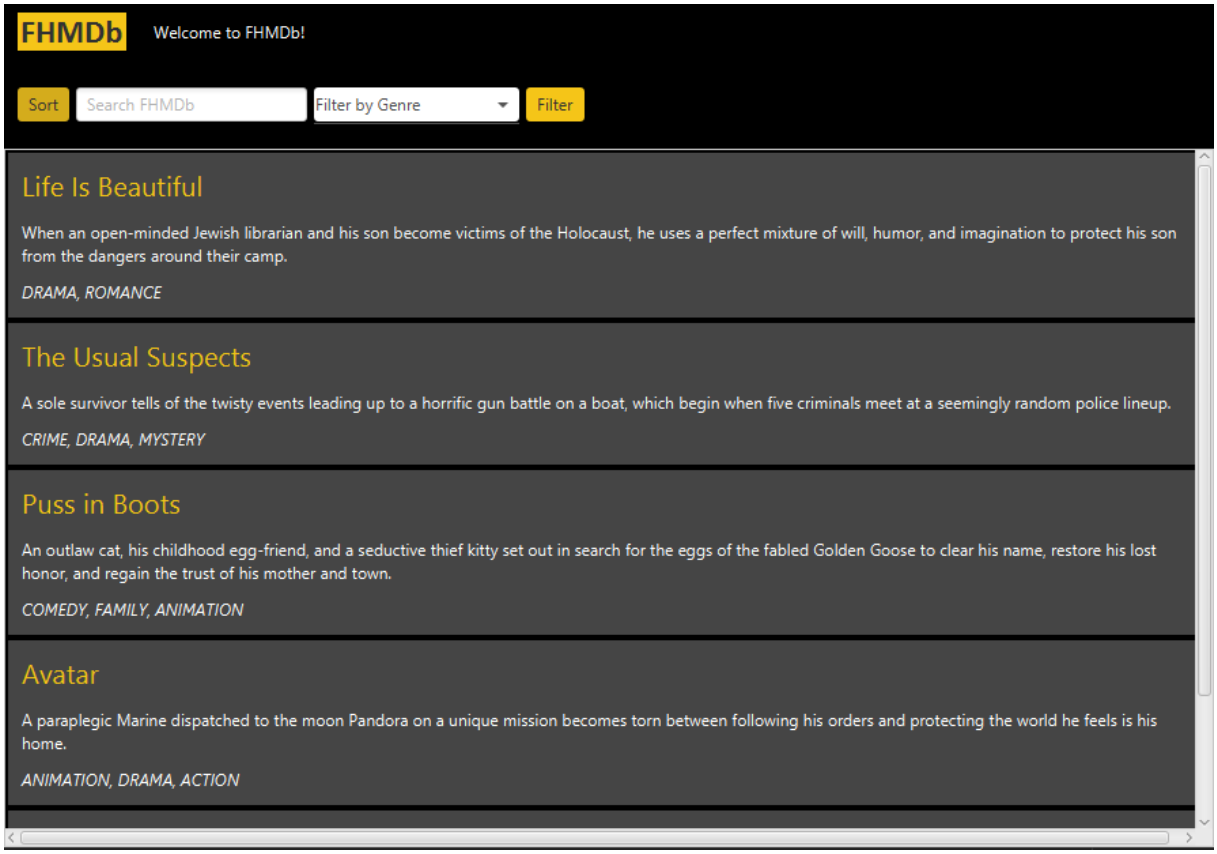


Abbildung 1: Beispiel Anzeige des Home Screen

Filter

In einem Eingabefeld können User*innen ein Query (Suchstring) eingeben, nachdem die Filme gefiltert werden können. Dabei gilt: das Query muss im Titel oder in der Beschreibung enthalten sein, Groß- oder Kleinschreibung soll nicht beachtet werden.

Außerdem können Filme auch nach einem Genre gefiltert werden. Sind sowohl das Query, als auch der Genrefilter gesetzt, werden beide Filterkriterien angewandt.

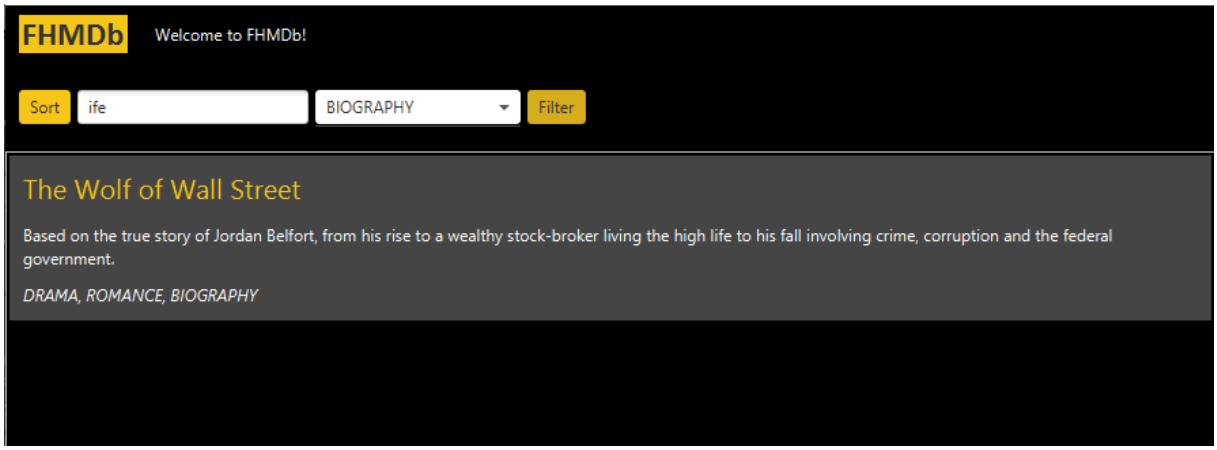


Abbildung 2: Query "ife" und Genre Filter "Biography"

Sortierung

Die angezeigte Liste kann über einen Button alphabetisch (absteigend und aufsteigend) nach Titel sortiert werden. Auch eine zuvor gefilterte Liste kann sortiert werden.

Tipp: Eine sinnvolle Aufteilung der Zuständigkeiten ist z.B. das Model-View-Controller (MVC) Pattern. Im Falle einer JavaFX Applikation könnte das MVC-Pattern wie folgt aussehen:

- Model: die zugrundeliegende Datenstruktur (eg. Movie)
- View: das für User*innen sichtbare .fxml-File
- Controller: eine Klasse, die Model und View zusammenführt und je nach getätigten Usereingaben Funktionen/Algorithmen ausführt. Dabei verwendet der Controller die Daten des zugrundeliegenden Models und aktualisiert die View dementsprechend.

Bewertungskriterien

- Die Applikation erfüllt die vorgegebenen Requirements (2 Pkt.)
- Der Code ist sinnvoll strukturiert, Coding Style Conventions werden umgesetzt (1 Pkt.)
- Sortierung und Filter werden ausgiebig mit Unit Tests getestet, sinnvolle Testszenarien abgeleitet (3 Pkt.)
- Unit Tests sind sinnvoll strukturiert und benannt und können automatisiert mit Maven („*mvn test*" command) ausgeführt werden. Siehe: https://www.baeldung.com/maven-surefire-plugin (1 Pkt.)
- Alle Teammitglieder arbeiten gemeinsam (mit GIT) an der Applikation. Jedes Teammitglied kann den Code bei der Abnahme erklären und Designentscheidungen nachvollziehbar begründen (KO-Kriterium)

Empfohlene Vorgehensweise

- Richtet ein GitHub Remote Repository für euer Team ein
- Es muss zusammen mit GIT gearbeitet werden (die Contributions der einzelnen Teammitglieder werden kontrolliert)
- Klont das Beispieltemplate oder erstellt ein neues Projekt (K.O.-Kriterien beachten)
- Implementiert die vorgegebenen Requirements. Überlegt welche Methoden sinnvoll sind und beachtet, dass diese weitestgehend testbar sind. Schreibt Unit Tests für die Funktionalitäten.

  Tipp: wenn ihr nach Test-Driven-Development Konzept entwickelt, sind die Funktionen von Beginn an testbar ;)

Abgabe

- Link zum Repository auf Moodle abgeben (letzter Commit vor Exercise-Abnahme wird bewertet)
- Das Repository muss public sein!
- Ist mind. eines der K.O.-Kriterien nicht erfüllt, wird die Abgabe negativ bewertet