Kina Aleksandra, Santos Sousa Daniel

M226 Projekt

# Verzeichnis

Inhalt

[1) Verzeichnis 1](#_Toc75031865)

[2) Klassendiagramm 2](#_Toc75031866)

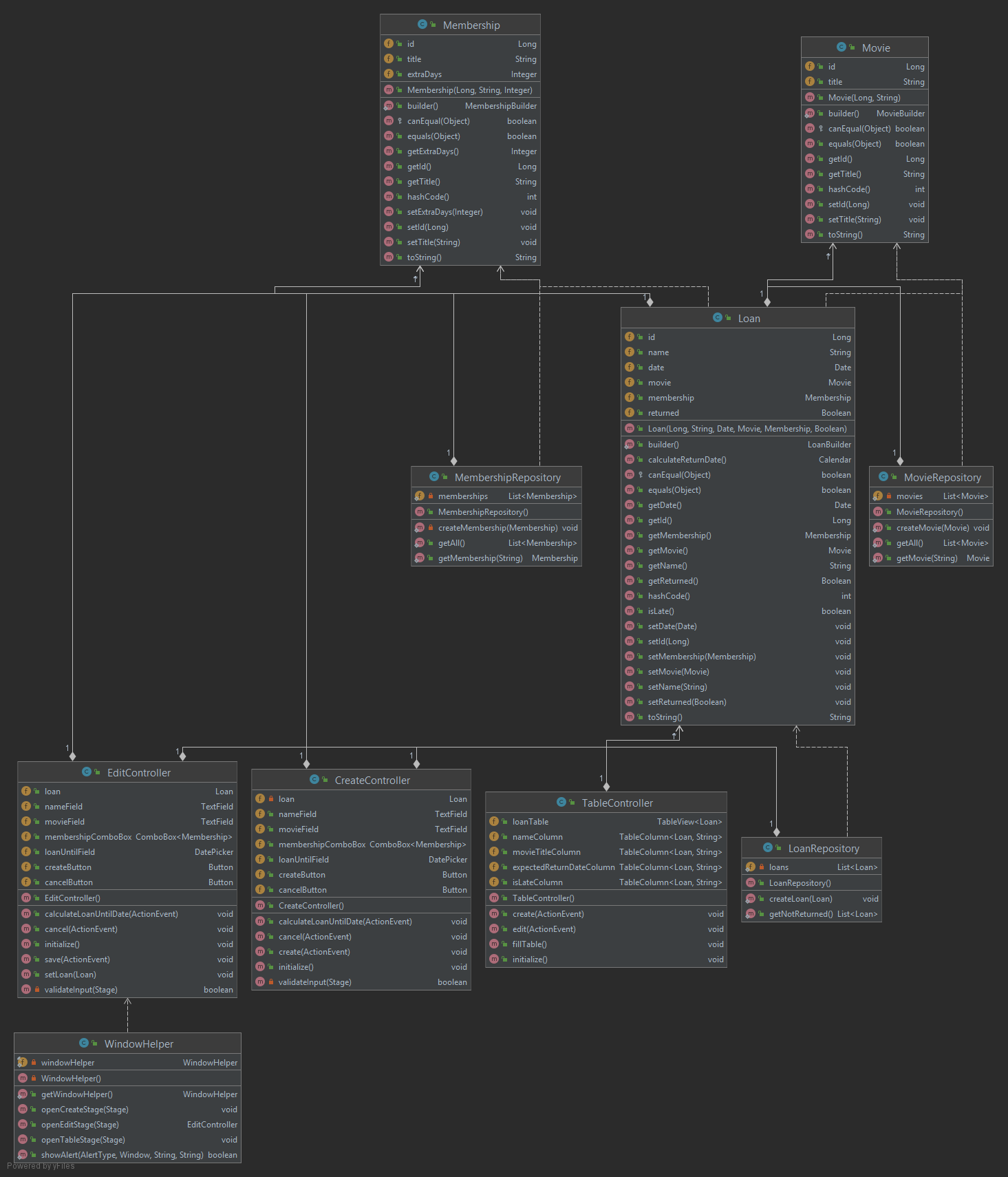
[1. Model 3](#_Toc75031867)

[2. View 3](#_Toc75031868)

[3. Controller 3](#_Toc75031869)

[3) Informationsquellen 4](#_Toc75031870)

# Klassendiagramm



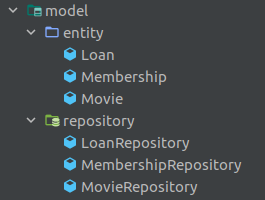
## Model

Das ganze “model” Package: Entities (Loan, Membership, Movie) und deren Repositories (LoanRepository, MembershipRepository, MovieRepository).

Die Entities sind Klassen, die einzelne Objekte repräsentieren (eine Ausleihe, eine Membership oder ein Film). Sie haben viele Membervariablen, und nützlichen Methoden wie Getters und Setters (werden generiert vor der Kompilation dank der lombok @Data Annotation).

Dazu haben unsere Entitäte alle auch das Builder Pattern implementiert (werden gleich wie Getters und Setters mithilfe der lombok @Builder Annotation).

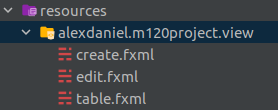
Die Repositories sind hier fürs Speichern und Verwalten von den Objekten. Aktuell werden die Daten mithilfe von einfachen Listen während der Laufzeit im RAM gespeichert (keine Daten Persistierung). Aber man könnte auch in so ein Repository Datenbank Code haben, oder die Daten in JSON serialisieren und in einer Datei speichern.



## View

Alle FXML Dateien im “resources” Ordner unter Package “view”.

Die Views diktieren das Aussehen und Struktur des GUIs.

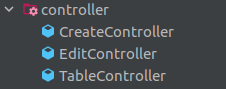


## Controller

Alle Controller Klassen im Package “controller”.

Jeder Controller gehört eine View.

Die Controller verarbeiten was bei den Aktionen des Benutzers passieren soll.



# Informationsquellen

Hier haben wir angeschaut, wie man das MVC umsetzen muss. Im Internet haben wir wenige Beispielprojekte angeschaut.

* vorherigen JavaFX Projekten
  + <https://github.com/alex-skxy/modul-318-student>
  + <https://github.com/alex-skxy/person-app>
* Internet

Sonstiges haben wir aus folgenden quellen:

* offiziele JavaFX Dokumentation <https://docs.oracle.com/javafx/2/>
* Stackoverflow

Aus [tutorialspoint](https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm) haben wir folgende Erklärung an welcher wir uns beim entwickeln gehalten haben:

### **Model**

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

### **View**

The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

### **Controller**

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.