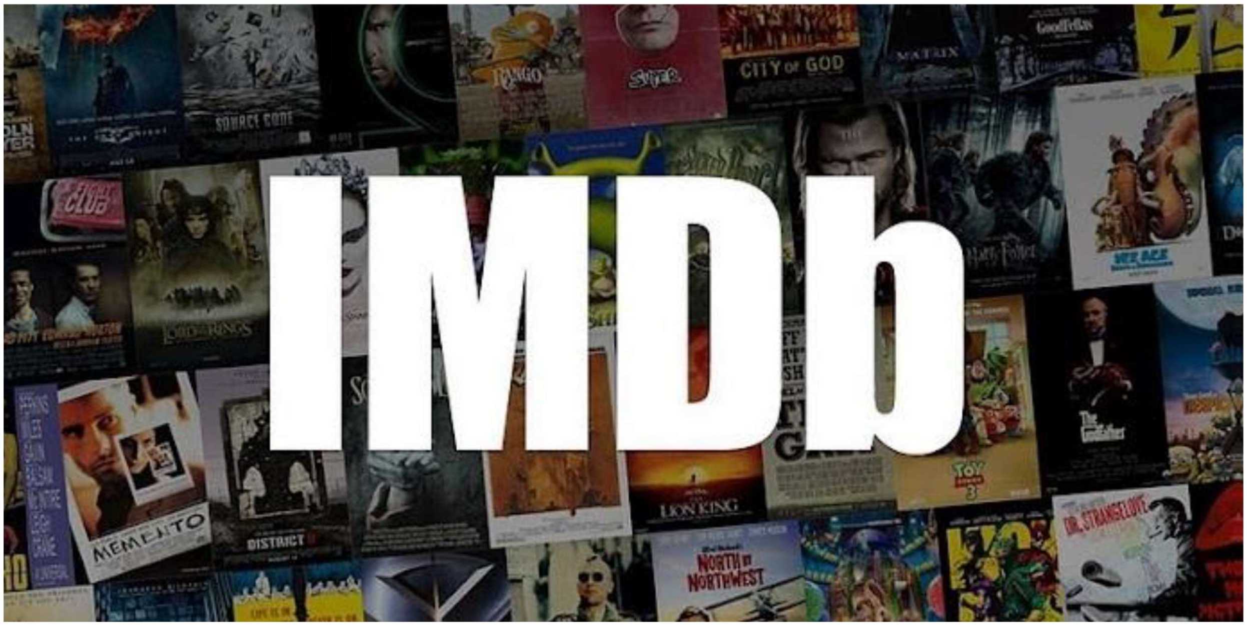

HUCS Cinema Reservation System

Taner Çelikkıran

2210356099

Index

Defining the Problem

A Cinema reservation system is requested. Users will be able to log in, select films, watch trailers, choose between film halls and reserve the seat they want. New users can sign up. There will also be a club membership system and member users will be able to reserve at a discount. Admin users will be able to edit users' statuses. They will be able to add films, remove films, and add or delete halls from films. They will also have full access to the seats. They will be able to reserve the seats they want for the users that they choose.

Solution Description

Main Classes

In the context of the problem, 4 different classes were created. These are “User”, “Movie”, “Hall” and “Seat”. All classes have overridden “toString” methods for the backup process. Also, all of the classes have getters, and setters except the final variables.

User Class

The “User” class has variables for username, password, admin status and club membership status. The password is hashed in the MD5 format and stored in Base64 format. There are static methods called “signUp” and “logIn” inside the class. These methods take and evaluate the necessary parameters for login and registration.

Film Class

The “Film” class holds the film's name, duration, relative path, and its halls. Films can be added or removed with the static “addFilm” and “removeFilm” methods.

Hall Class

The “Hall” class has variables for the film that the hall belongs to, the hall’s name, price, number of rows and columns, and the seats that the hall belongs to. It has the “addSeat” method for adding seats to the hall. Also, it has “addHall” and “removeHall” methods for adding new halls to the chosen film and removing the hall.

Seat Class

The “Seat” class is the last. It has variables for the hall that the seat belongs to, seat’s row, column, bought price and the owner user of the seat.

Exception Classes

There are 4 exception classes. These are “BannedAttemptException”, “EmptyKeyException”, “ExistingKeyException” and “UnmatchedKeyException”.

BannedAttemptException

This exception is thrown when the person is banned to log in but still tries to log in.

EmptyKeyException

This exception is thrown if there is a text field to fill and it’s empty.

ExistingKeyException

This exception is thrown if the new name of something (new user, new film, new hall eg.) already exists. For example, if the user tries to add a new hall named “S1” and there is a hall that is already named “S1” in the database, this exception is thrown.

UnmatchedKeyException

This exception is thrown if the passwords are not matching on the login screen or sign up screen.

Pane Classes

In this project, one pane class is written for each visible screen. These are all static and in the "Main" class. Thus, a structure that is both easy to read and open to innovation but closed to change is provided.

Problems Encountered and Solutions to These Problems

First Problem: Positioning of the Nodes

The first problem was the positioning of the nodes. This problem has been resolved with nested pane structures and alignment adjustments where necessary.

Second Problem: Switching Between the Scenes

The second problem was switching between windows. This problem has been resolved by accessing the primary stage and changing the scene of the stage.

Third Problem: Banning the User

The third problem was banning the user when the incorrect login attempt exceeds the limit. This problem is solved by this algorithm:

- The system millisecond of the moment the user presses the key in the wrong login attempt is kept.
- In subsequent attempts to enter, the difference between the milliseconds when the key is pressed and the milliseconds when the wrong entry is exceeded is checked.
- If the difference has not exceeded the penalty period, the user cannot enter. If it's over, the user can try again.

Benefits of the System

It would be profitable if such a system could be used sustainably. It will also eliminate the need to go to the hall itself to buy a seat. So users will go to the halls just to watch. This means speeding up the ticketing process as well.

Benefits of the GUI

The person or people who made this system can use the same system without a GUI and operate it this way. But the people who will use this system are ordinary people and they do not know how this system works. Therefore, showing the complex code stack on the back in a simple way is the main and most important job of the GUI. Also, it will be more comfortable to use a program in this way, even for the coder.

Extra Feature

In real life, users may or may not like movies. Having this information can be a guide for users who will buy new tickets. In addition, unpopular movies are determined and new ones can be added when necessary. That's why a "like-dislike" system has been added. In this way, the system will be more profitable and effective.

Resources

- Introduction to Java Programming and Data Structures, 12th Edition, Daniel Liang
- Learn JavaFX 17, Second Edition, Kishori Sharan, Peter Späth
- Stack Overflow - https://stackoverflow.com/
- Geeks for Geeks - https://www.geeksforgeeks.org/
- Java Tutorial - https://docs.oracle.com/javase/tutorial/