**Object Oriented Programming Java Final Project**



**By:**

Michael Christopher

2440047362

Class L2AC

**Lecturer:**

Jude Joseph Lamug Martinez

D4017

**Project Specification**

1. **What’s the project?**

In this project, I made a simple GUI using the Java programming language and JavaFX. It is a watch/read list GUI application that is connected to databases. The application also has a login menu to ask users to input their username and password.

1. **Purpose of the project**

In this project, I create a GUI in Java for the final project and this GUI allows users to record their watch/read list. For example, users can add movies that they haven’t watched and want to watch later, they can check what books they are reading, they can keep track on tv series that they have watched, they can update the records, and delete the record if it is no longer necessary.

1. **What’s the vision and mission for this project?**

To make a functional program that a lot of people will find it useful and they like using it. This program needs to be able to keep track of the users record by saving data to database.

**Solution Design**

1. **Introduction**

In this second semester majoring Computer Science in BINUS University International, we were taught about Object Oriented Programming with Java as its programming language. The lecturer taught us from the start such as data types, variables, conditional, loops, and eventually later on advanced to OOP. We have learned OOP back in our first semester using Python programming language, but we dive deeper to the world of OOP in this second semester course. We were taught basic about OOP, 4 pillars of OOP, how OOP works in Java, how to implement it in Java, how to design classes, creating UML diagram, abstraction, and many more. Just like the first semester, we need to create a final project implementing what we have learned during our class and we give presentation plus demonstration of our program to the lecturer.

For me, I didn’t think about this project until around 1 month before the submission date. I spend like almost a week to think what I want to make and searching what kind of framework I want to work with. Initially, I want to make this project using Springboot framework and make a web application but after watching some tutorial, I find the quite hard to implements since I need to also code the front end of the web using HTML, CSS, or other similar language. After that I switched my idea to make a GUI application and I tried to use Swing, but since Swing is quite old fashioned so I move on to a newer framework called JavaFX. It is a lot easier using the JavaFX than Springboot because I can drag and drop the application without coding. The idea of making the GUI application as a read/watch list application actually just came to my mind in the process of making the application.

I started doing this project in the middle of May 2021 and I use Apache Netbeans as my IDE for this project. This project will be uploaded in my GitHub account, <https://github.com/EuphosiouX/FinalOOP.git>

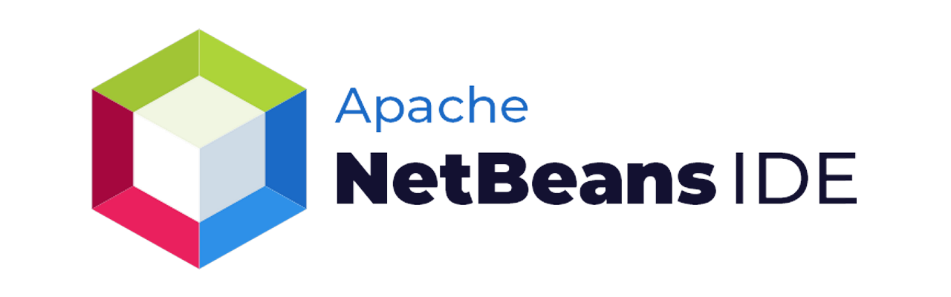


Image 1 – Apache Netbeans IDE

(https://www.andreszsogon.com/wp-content/uploads/logo\_apache\_netbeans\_cordova.png)

1. **Overview**



Image 2 – Application’s dashboard

I don’t have a title or name for this GUI application so I just call it “Read and Watch List Application” and it is an application where users can input movies, books, or tvseries to record them. This application can be used by people in various age range and this application might be useful to them and provide good services.

This GUI application is created in Java 8 using JavaFX framework, Gluon Scene Builder app to build the application appearance, MySQL for the database, and MySQL connector.

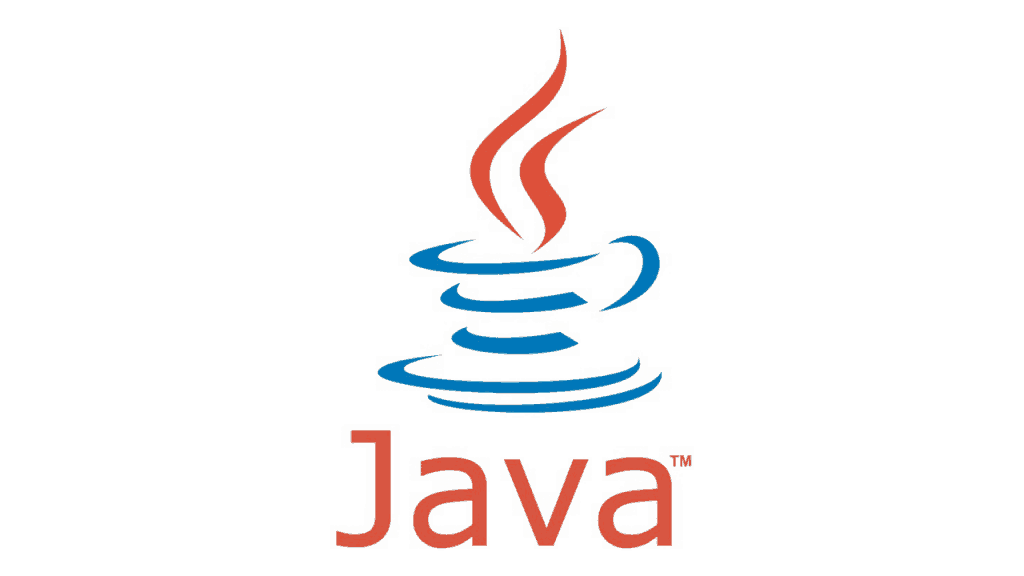


Image 3 -Java logo

(<https://spzone-simpleprogrammer.netdna-ssl.com/wp-content/uploads/2016/11/Untitled-1-6-1024x576.png>)



Image 4 – JavaFX logo

(<https://bangness.net/wp-content/uploads/2019/01/JavaFX_Logo.png>)



Image 5 – MySQL logo

(<https://upload.wikimedia.org/wikipedia/id/a/a9/MySQL.png>)



Image 6 – Gluon Scene Builder Logo

(<https://gluonhq.com/wp-content/uploads/2015/02/SceneBuilderLogo-300x300.png>)

1. **Code Flow**

The codes and classes for this program are all categorized into packages of their own category so there are a lot of packages in this project. For example controllers class are all in the same package, entertainments class with its child in the same package, and others. For the UML class diagram, I use simple classes since some of my class have a lot of variables and methods that it will not fit into the diagram but I also provided the complete diagram. The complete diagram have incomplete and incorrect relationship between class so it is used only to see what variables and methods are inside the classes and please use the simple diagram to see the relationship between classes. Also, the color on the lines and some classes is only to make it easier to see the lines and serve no other purposes. For clearer UML diagram, .png files are available in the GitHub link provided.

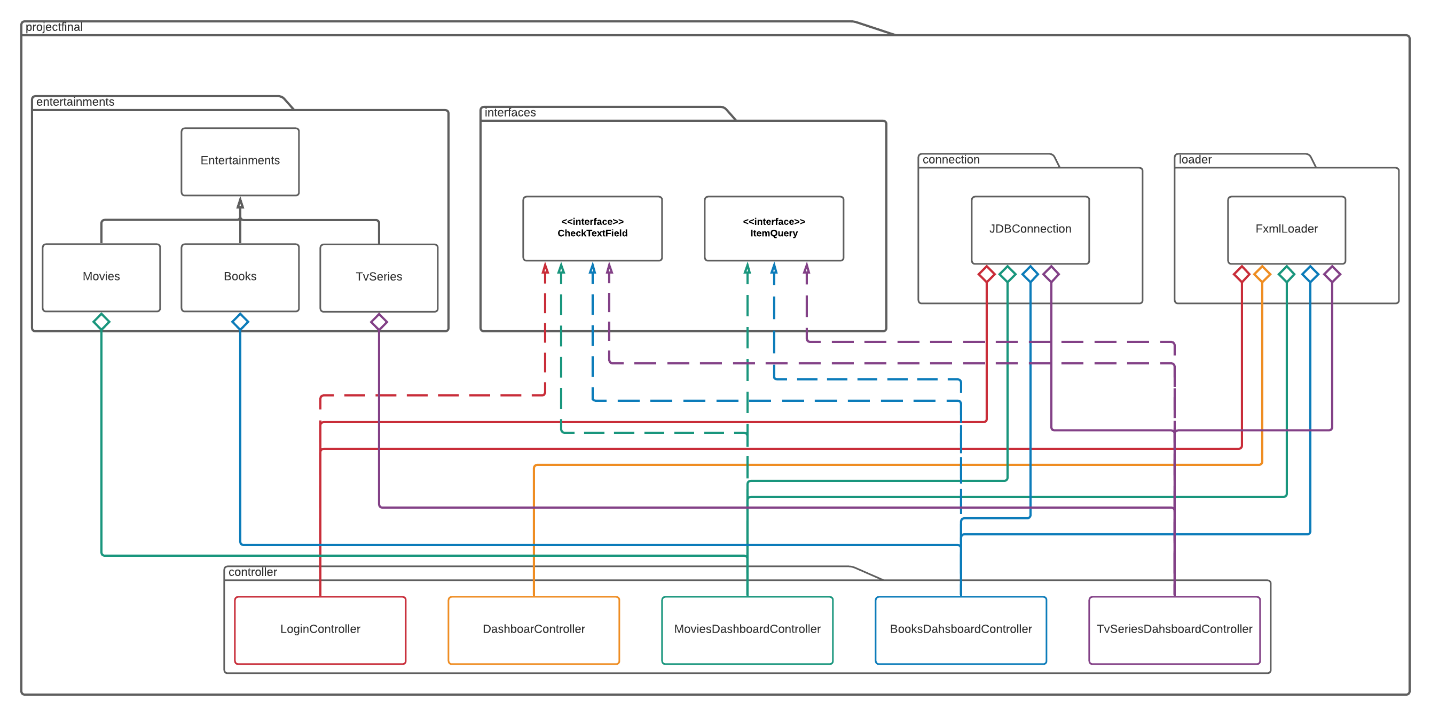


Image 7 – Simple UML Diagram

(simpleDiagram.png)

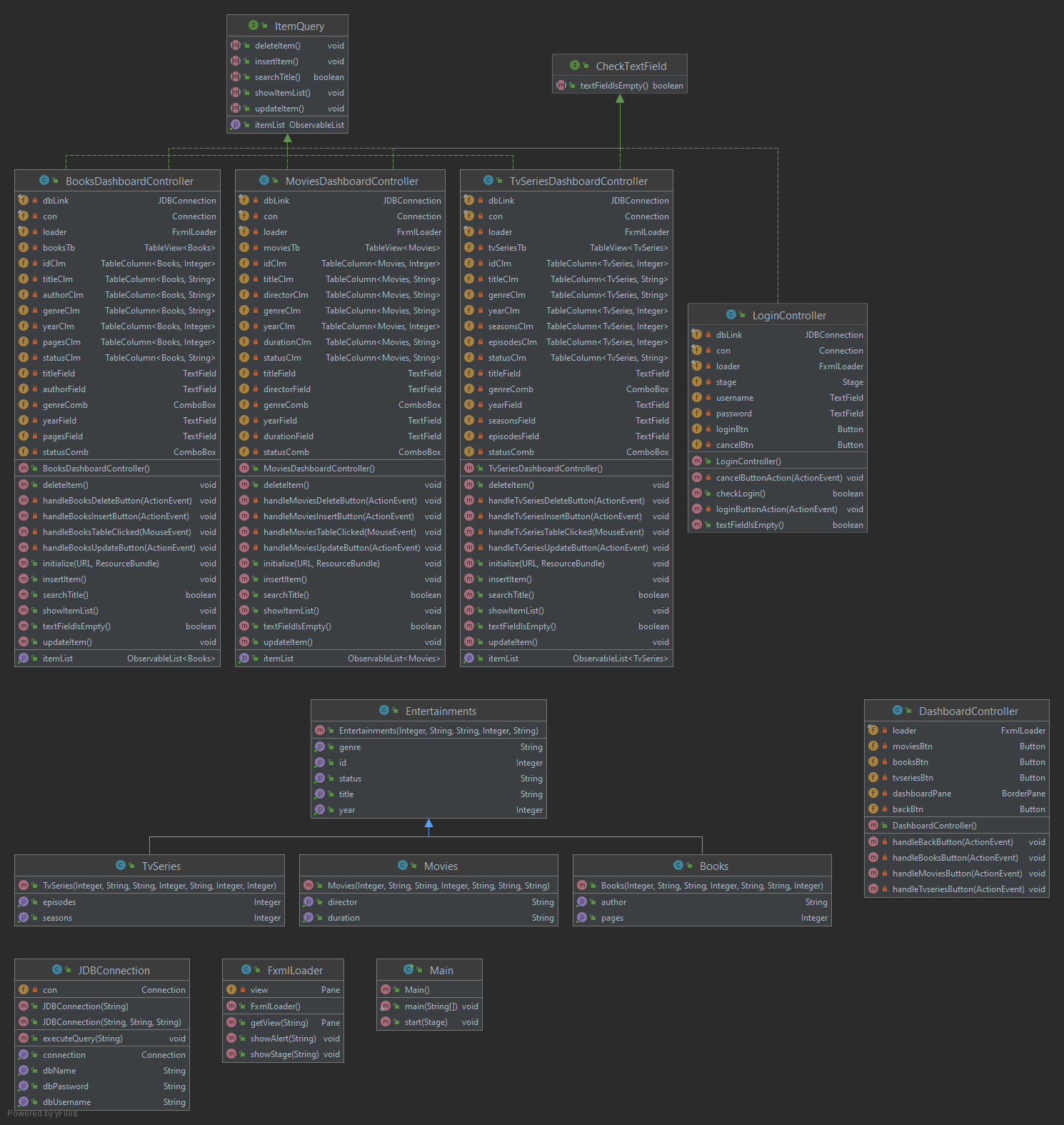
****

Image 8 – Complete diagram

(completeDiagram.png)

1. **Visual Design**

The visual design for this application is simple and straightforward using solid color for the appearance

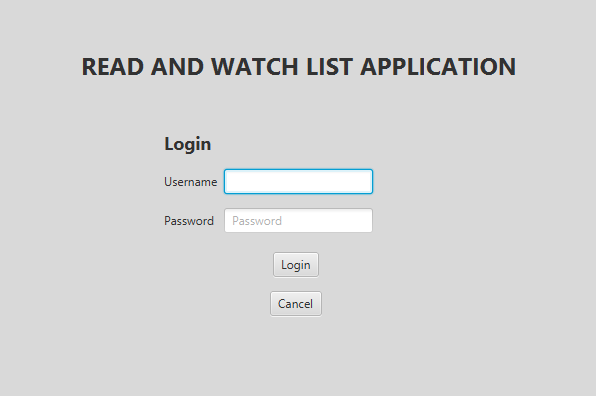


Image 9 – Login menu



Image 10 – Application’s dashboard

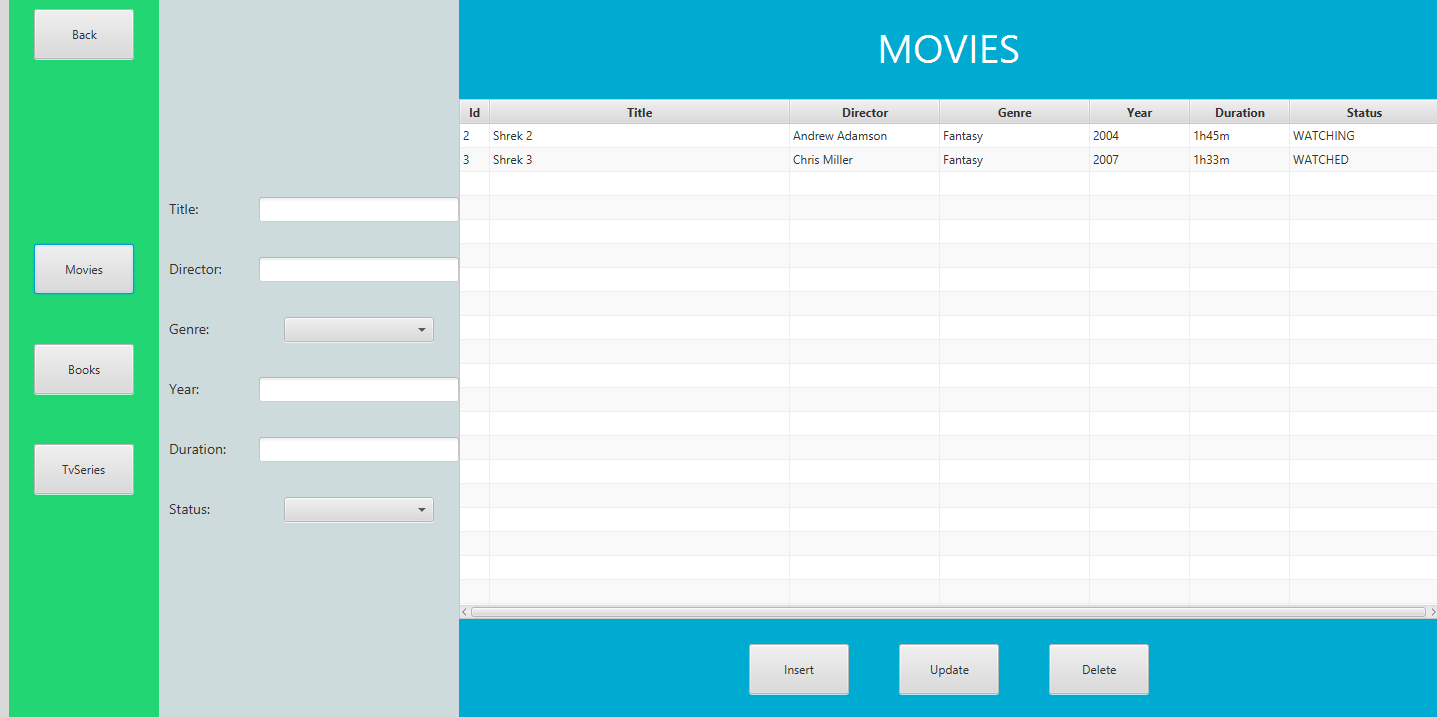


Image 11 – Movies dashboard

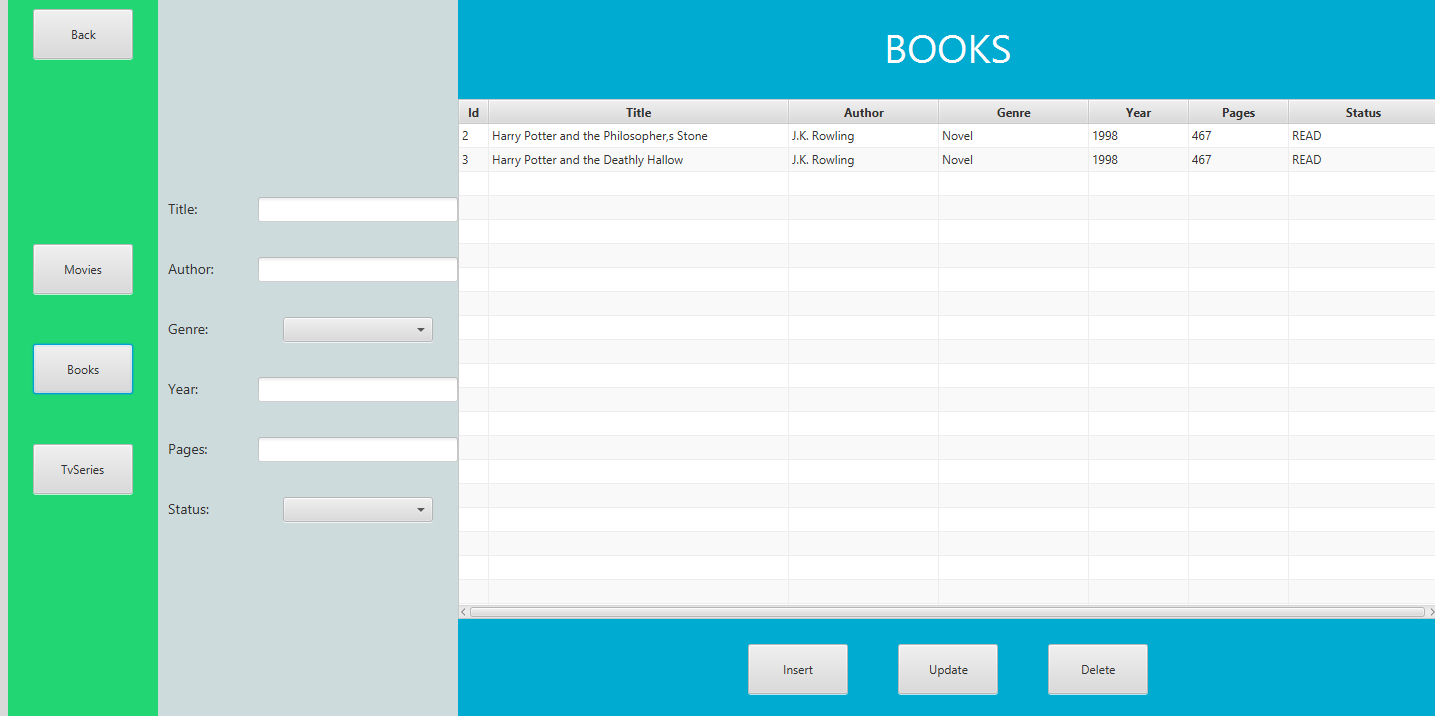


Image 12 – Books dashboard

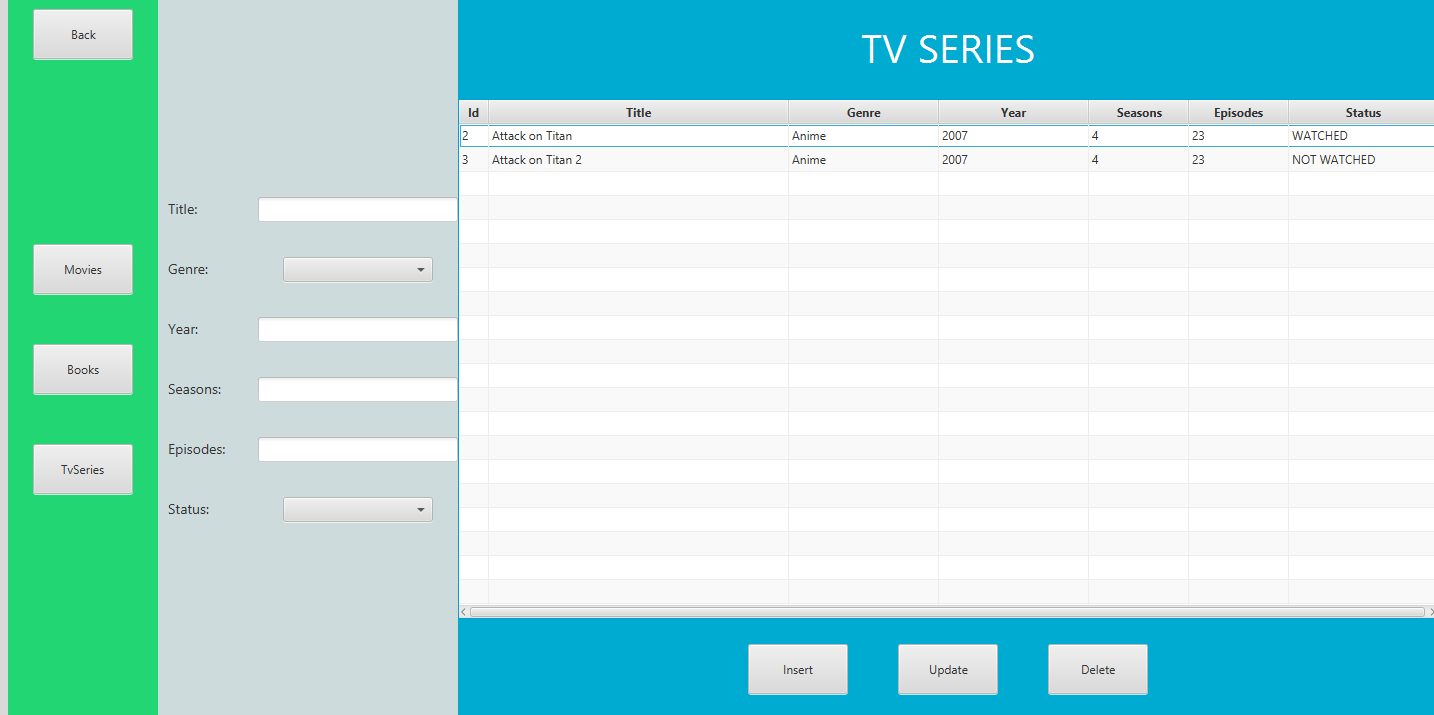
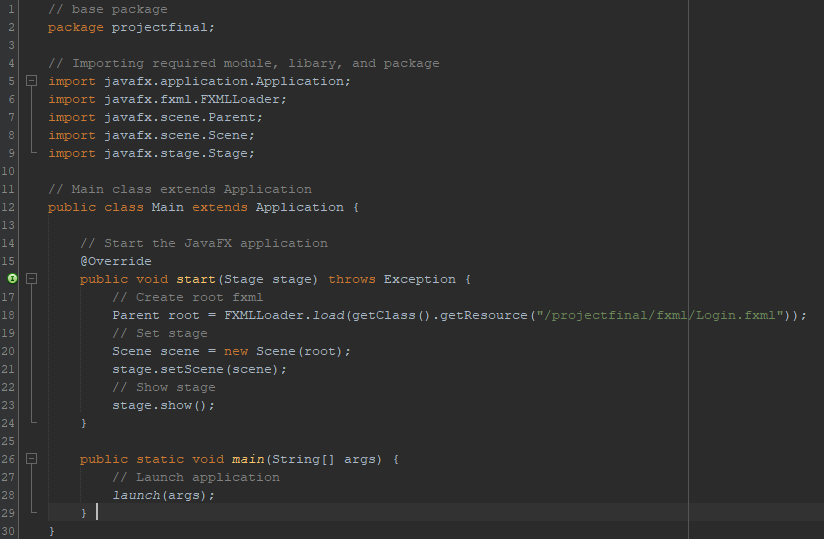


Image 13 – TvSeries dashboard

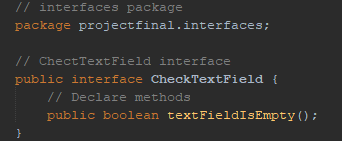
**Implementation**

1. **Main**



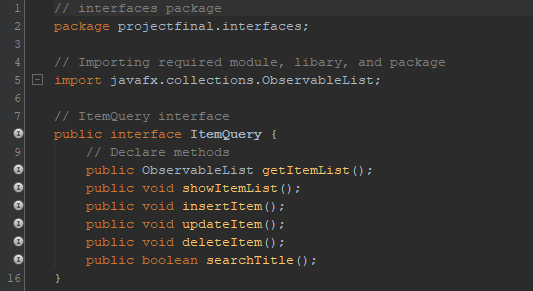
Firstly, I created the main class for this project and this class extends from the JavaFX Application class. To start the application, I need to implement the start() from the Application to start the application. In the start(), I set the parent root of the initial fxml file that I want to load for this case it is the Login.fxml file. Then I create a new Scene object of root and set the scene to the current stage. Finally, I can show the stage and launch the application.

1. **Interfaces**
2. **CheckTextField**



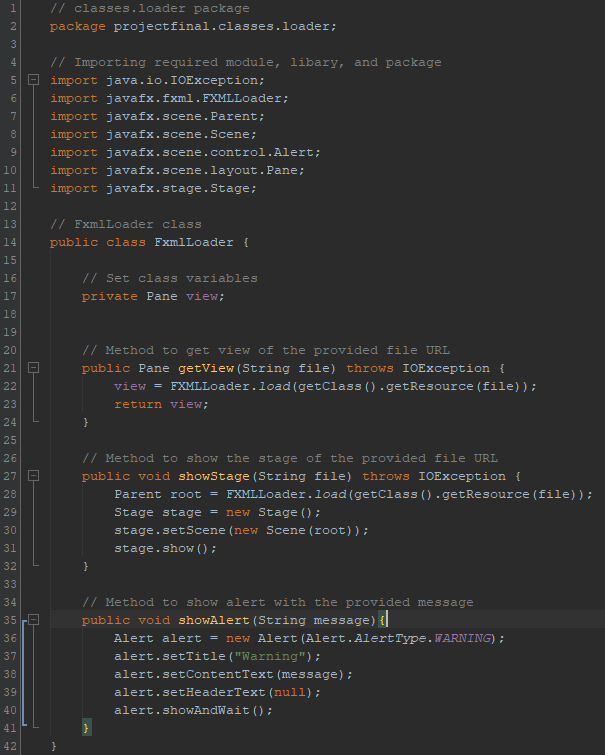
CheckTextField interface that has textFieldIsEmpty method that will be used later on by different class.

1. **ItemQuery**

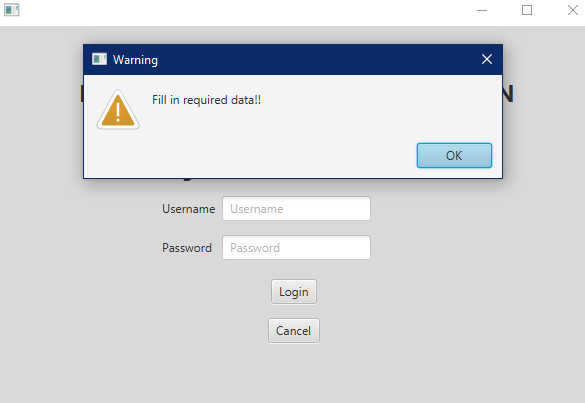


ItemQuery interface that has getItemList, showItemList, insertItem, updateItem, deleteItem, and searchTitle methods that will be used later on by different class.

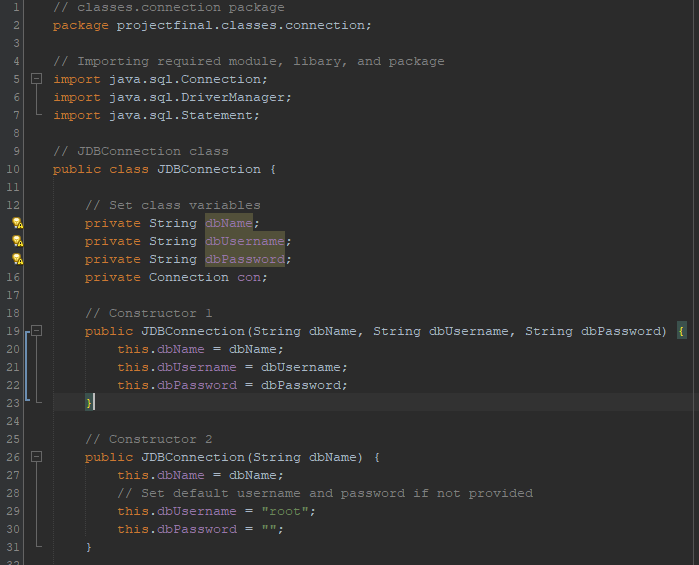
1. **Classes**
2. **FxmlLoader**



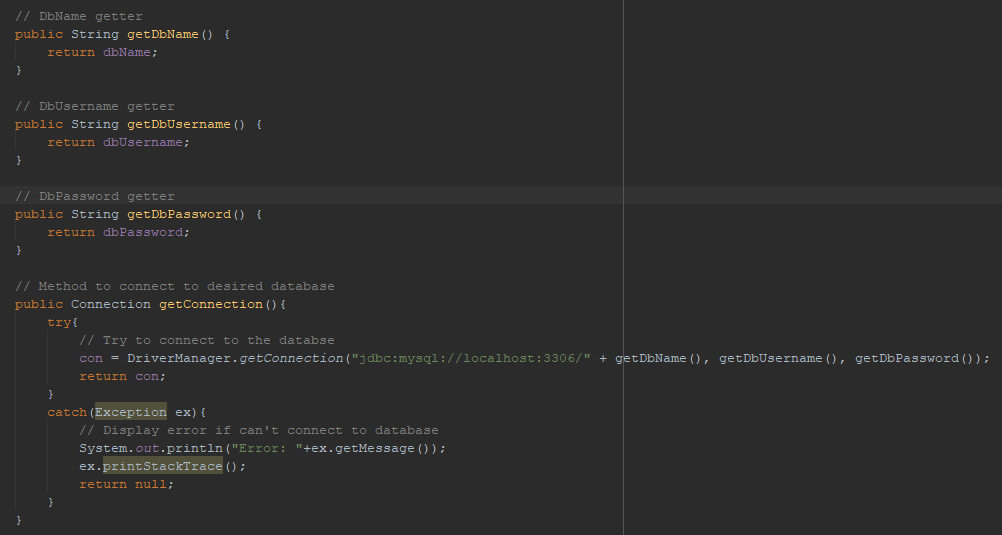
I created this FxmlLoader class to handle the application’s GUI and fxml files. I created 3 method in this class, the first one is getView() and this method is used for loading the Pane view of the fxml URL file provided and return the view so it can be used later on to show other fxml inside an fxml. Similar to getView(), showStage() is also used to load other fxml file with the provided URL but the differences is that this fxml file is saved as Stage object instead of Pane object and this method also directly show the new Stage and return nothing. This showStage() is used to open another fxml file in other window instead of loading it inside another fxml file like getView(). The last method that I created in this FxmlLoader file is showAlert() and it is a method that takes an error message of type String and the purpose of this method is to show an alert pop up window when something wrong happen for example if the user doesn’t fill all of the required data, it will show a pop up saying “Fill in required data!!”.



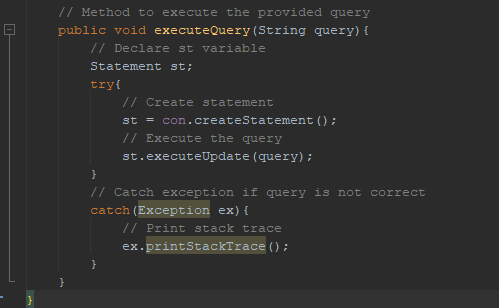
1. **JDBConnection**



Next. I created this JDBConnection class and it is a class that will handle the connection to a database(for this project I use MySQL). There are 4 variables in this class which is dbName, dbUsername, and dbPassword with all of them being a String data type and con variable with Connection data type from sql.Connection. There are 2 constructor in this class and the first one takes dbName, dbUsername, and dbPassword and the second one which only takes dbName as its parameter and use “root” as the default dbUsername and “” as the default dbPassword.

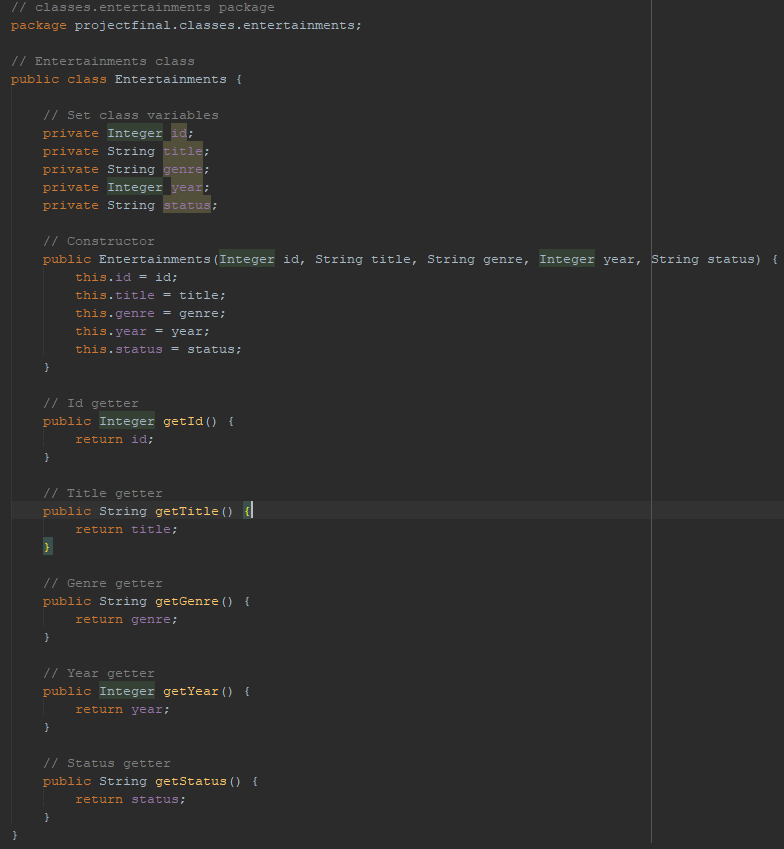


Here is the getter of the class and the first 3 getter for dbName, dbUsername, and dbPassword is the generic getter where it just return back its own variable but for the getConnection(), I use a try and catch statement. First, the program will try to connect to the database using DriverManager.getConnection(”jdbc:mysql://localhost:3306/{database name}”, {username}, {password}). The port number may change in different machine but it is using port 3306 in my machine. Return the connection if it is successful but if there is something wrong when connecting to the database, it will catch what is the exception and print out the error message with the stack trace and return nothing.



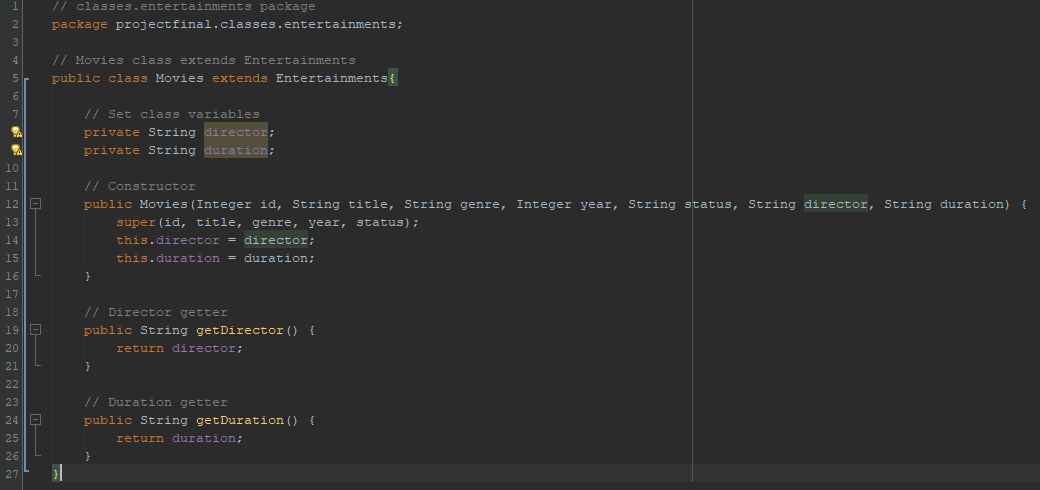
The last method in this JDBConnection class is executeQuery(). The purpose of this method is to execute a query given in the parameter to edit the table in the database. Query is a syntax used by SQL to access and show data in a SQL database system. How this method work is it will try to create a statement and executing the query provided in the parameter. If the query is eligible, SQL will understand them and run them to edit the database but if the query is not eligible or wrong, it will catch the exception and print the stack trace.

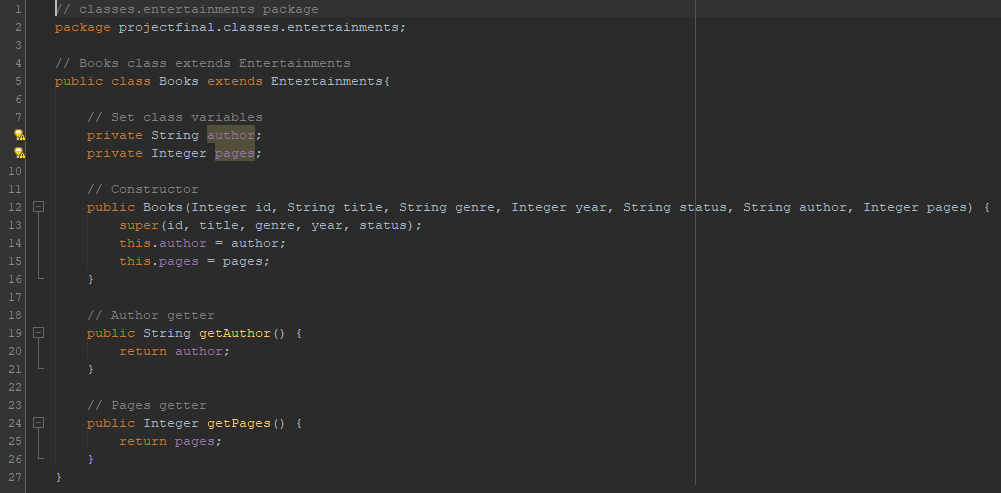
1. **Entertainments**

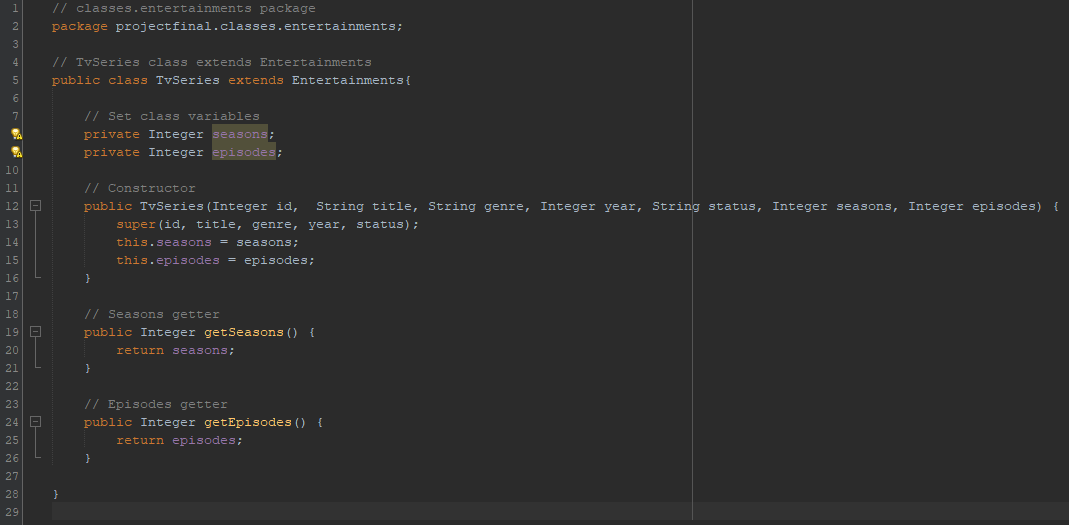


Next, I created the Entertainments class and this class will be the parent class of Movies, Books, and TvSeries class and these class later on will contain the variables that will be inserted to the table database later on. This entertainments class have 5 variables, id, title, genre, year, and status with a constructor taking every variable as its parameters. I also implement the getter for each variable.

1. **Movies, Books, and TvSeries**

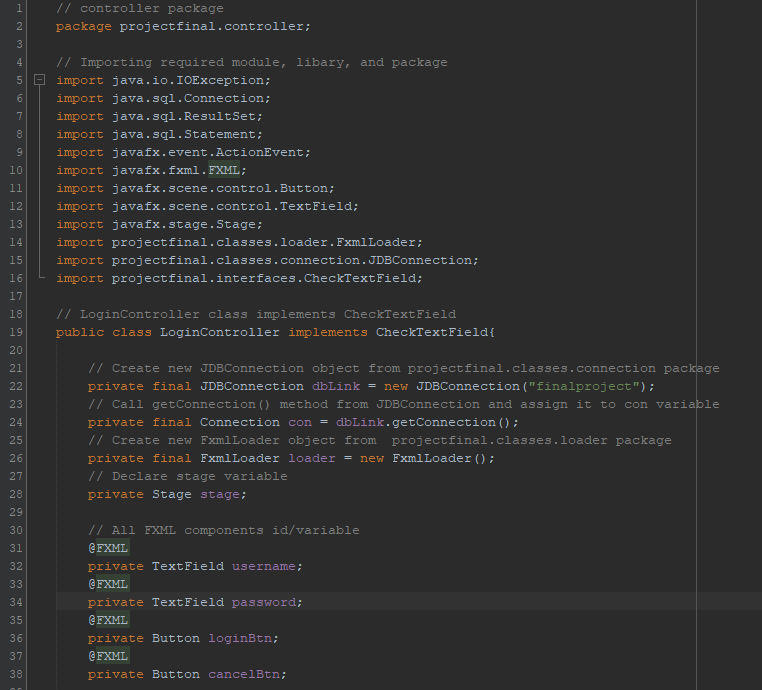


****

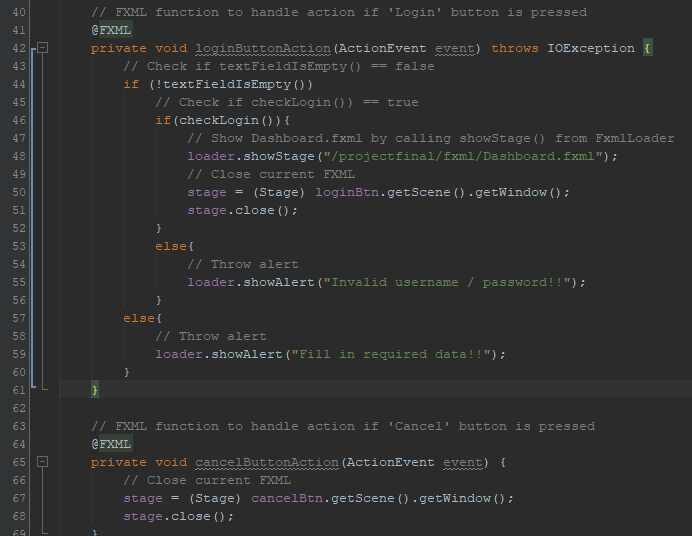
****

As mentioned before, Movies, Books, and TvSeries class are a child class of Entertainments class so they extends from Entertainments. Each of them have two additional variables inside them, director and duration for Movies, author and pages for Books, and finally seasons and episodes for TvSeries. I also include getter for each variable in each class.

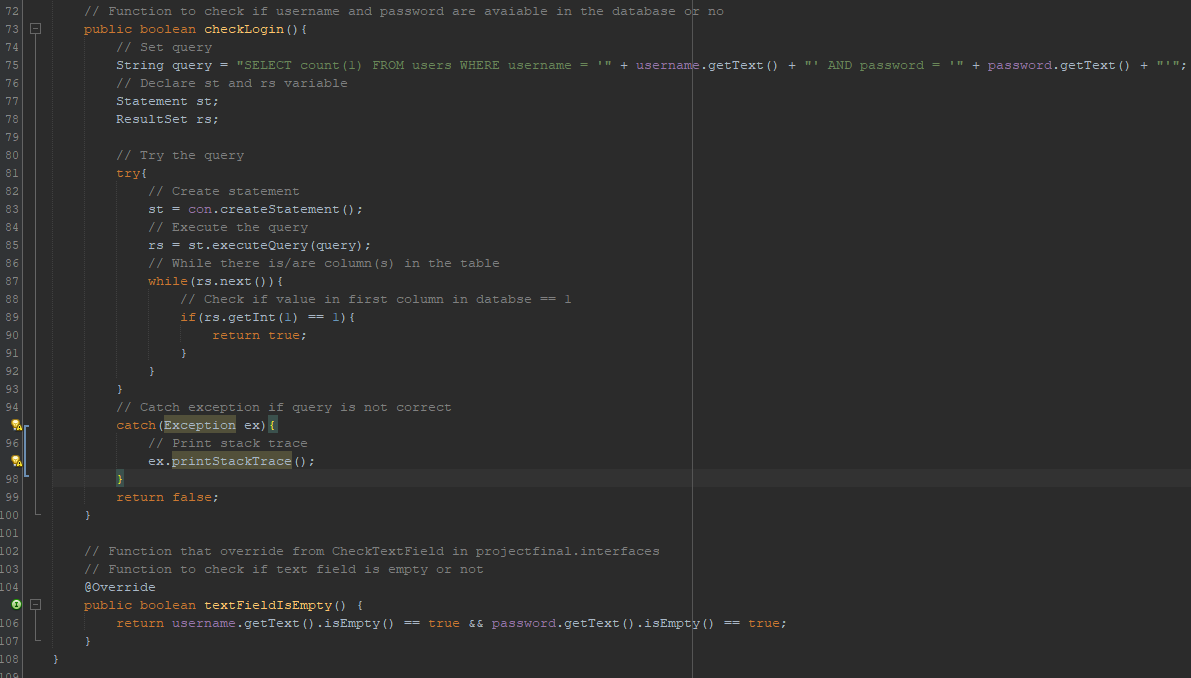
1. **Controller**
2. **LoginController**



This is a class called loginController and just like its name, the purpose of this class is to handle and control the fxml and application’s appearance and it is also the same for other controller. First, I imported all the necessary library, module, and package and this class will implements CheckTextField interface. This class also have objects from other classes that have mentioned before and that is from JDBConnection to get connection to the database and from FxmlLoader to handle loading other fxml files. There is also variable calles stage with Stage data type and I created all of the fxml related variables.

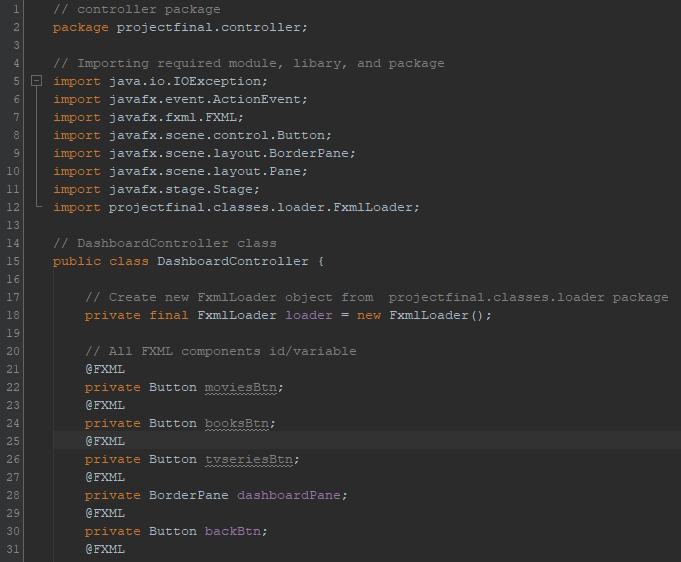


This next two methods that handle action of certain button if pressed. The first method is to handle login button action and it will check if the text field empty or not and if it is empty, throw an alert and if it is not empty, check if username and password exist in table called “users” or not and if it doesn’t exist, throw another alert. After checking the user’s username and password and if they are correct, close current fxml and open Dashboard.fxml. The second method is to close the application if cancel button is pressed.

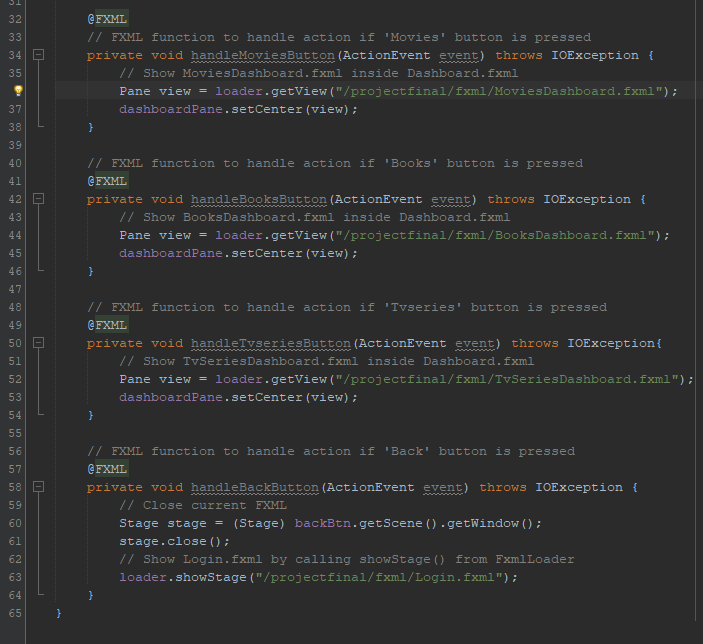


This next two methods is basically the methods to check if username and password exist in the table and check if input fields empty or not respectively. The textFieldIsEmpty method overrides and implements the CheckTextField interface.

1. **DashboardController**

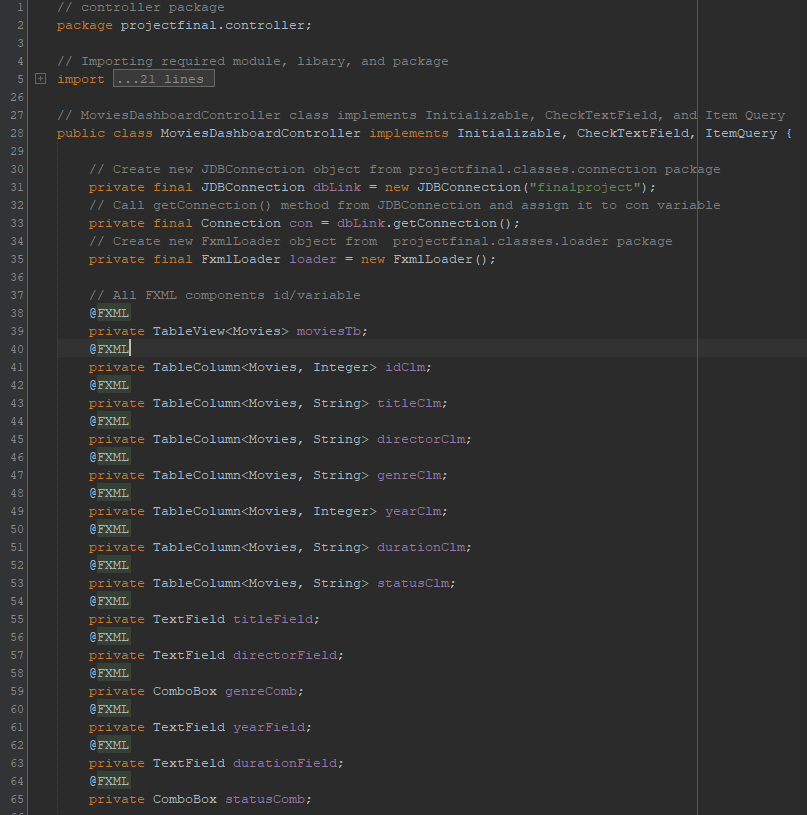


This is the class that control the dashboard and the class variables is similar to LoginController like it have fxml related variables and loader from FxmlLoader but the differences is that this class doesn’t implement any interface, DashboardController doesn’t need an object from JDBConnection like in LoginController since this class doesn’t need any database connection , and this class doesn’t need Stage.



These methods are to handle actions of each button pressed. If movies button is pressed, open MoviesDashboard.fxml to the current fxml, If books button is pressed, open BooksDashboard.fxml to the current fxml, If tvseries button is pressed, open TvSeriesDashboard.fxml to the current fxml, and go back to login menu if back button is pressed.

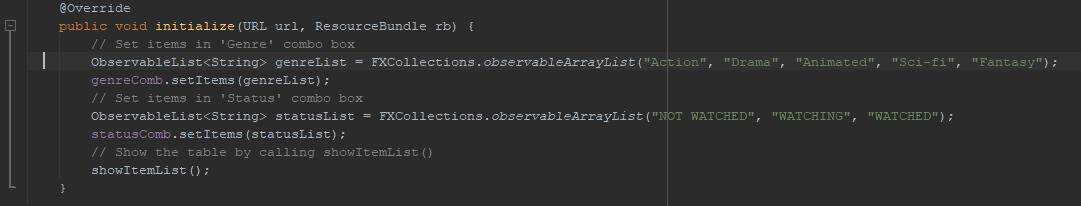
1. **MoviesDashboardController**



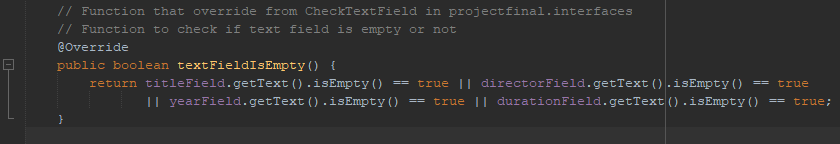
Just like LoginController, this class implements CheckTextField interface, have object from JDBConnection and FxmlLoader, and have fxml related variables, but this class also implements additional interfaces, which is Initializable and ItemQuery.



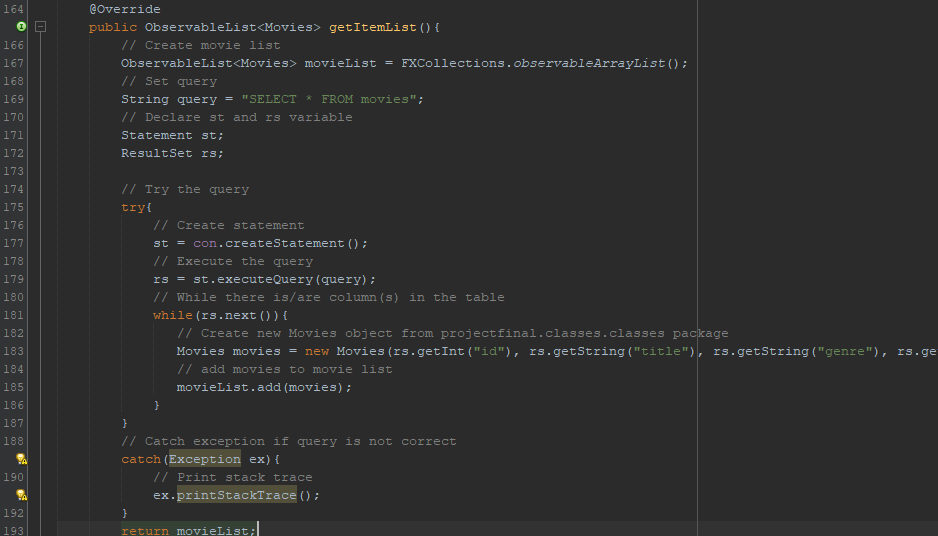
These are the method that will handle clicked action for insert, update, delete button, and table row. After the insert button is pressed, throw alert if text fields are empty and throw alert if title already existed in the table. For update and delete button, instead of throwing alert if title exist, it will throw alert if title NOT existed in the table. If any row in the table is pressed, it will print the value of each column back to the text fields.



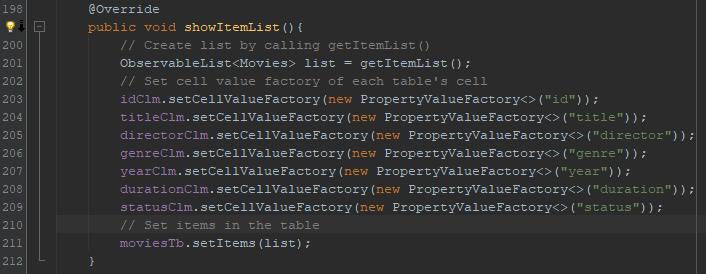
This is the method that initialize and do actions when the fxml first showed in the screen and the actions are setting the combo box value and show the table.



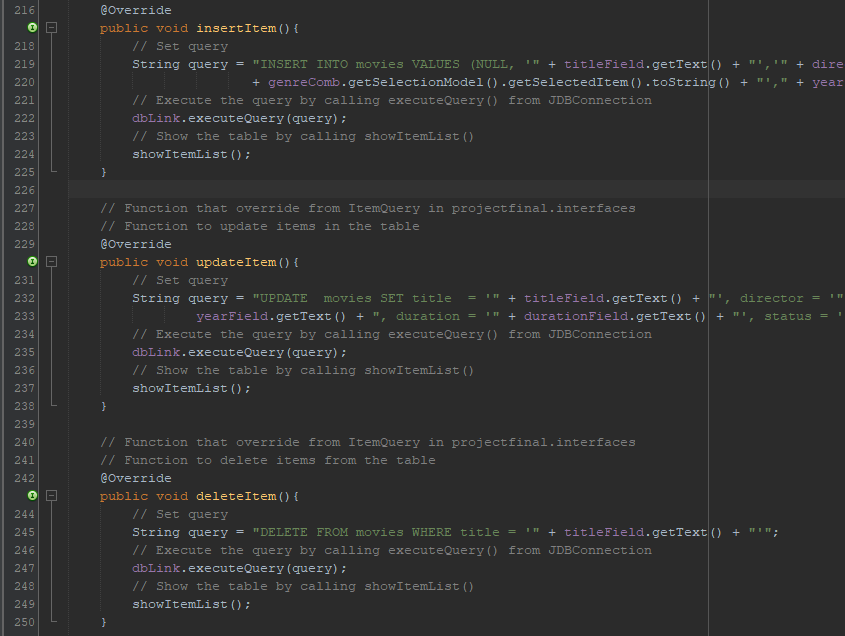
This is a method that will return true if there are any empty text field.



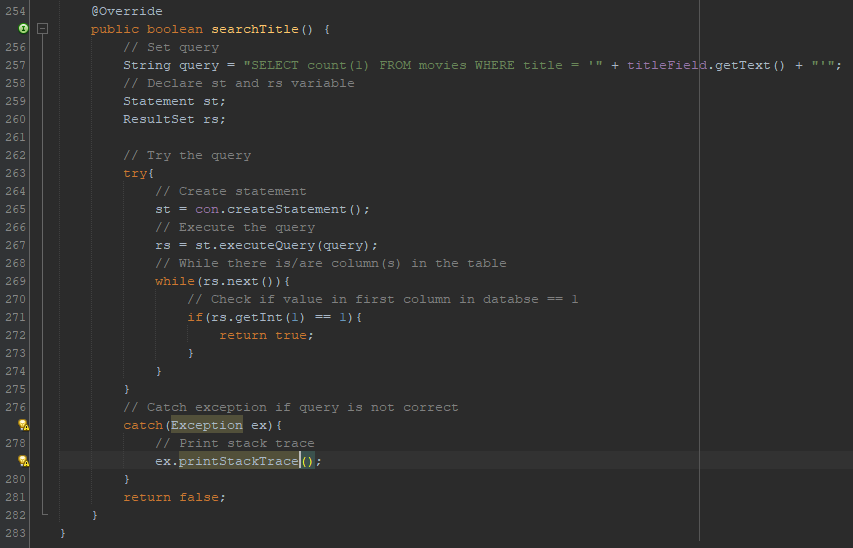
This method is to get item list from the table and how it works is first, create movieList variable of type Observable list. Next, create a query string and statement and execute that try to execute the string query statement. If there are rows in the table, create movies object with inputted data in the text field as its value then add them to the observable list and return it. If the string query is not correct catch the exception and print stack trace.



This method is to show the item list from the table and how it works is first, create a list of item by calling getItemList(). After that, set cell value factory of each table column with the corresponding column name and then set the items into the table.



Insert, update, and deleteItem methods are actually the same the only difference is the string query in each method. How it works is that it will edirectly execute the string query and show the changes into the table.



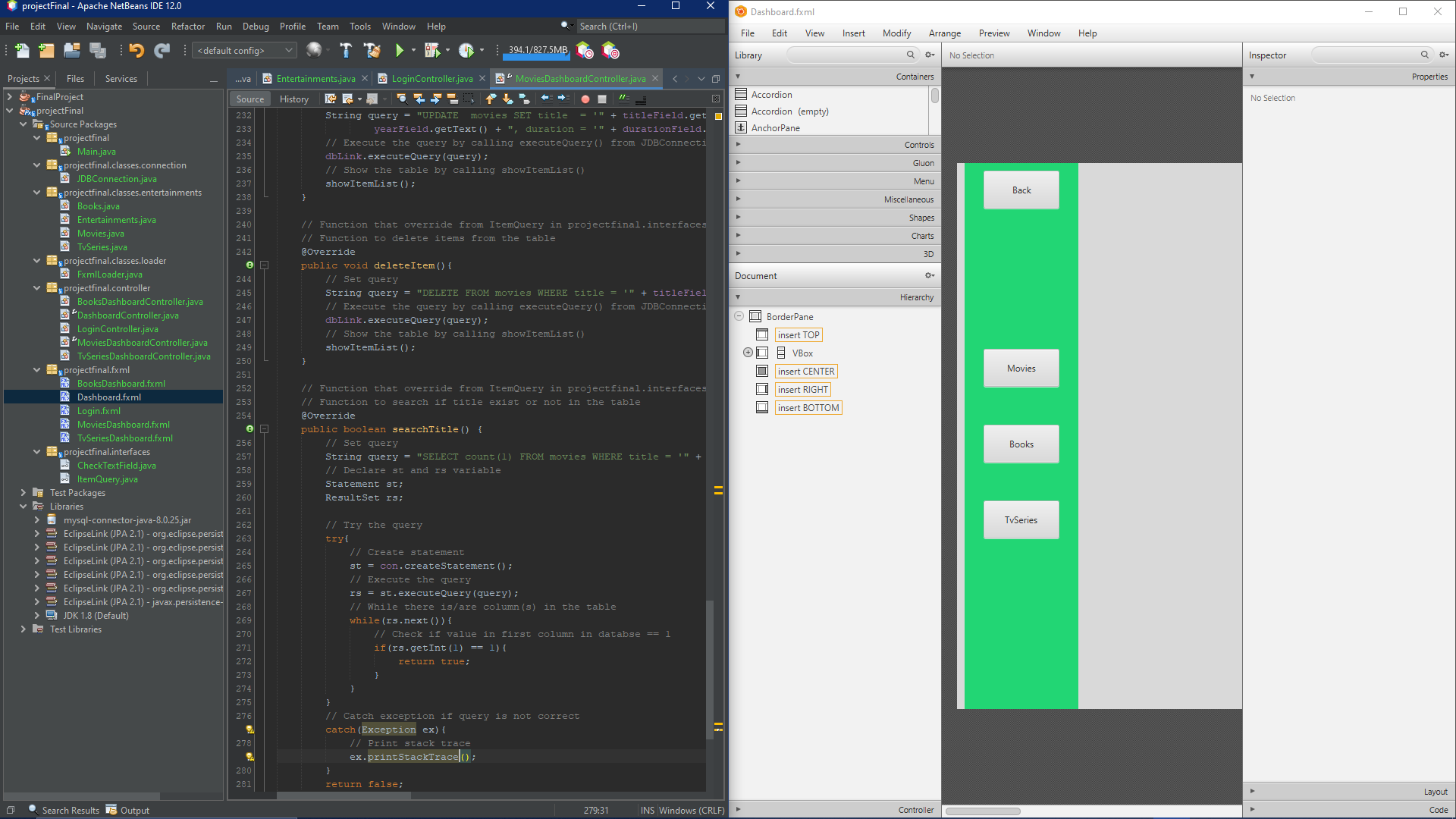
Lastly, this is the method to search title and return true if title exist in the table and return false if it doesn’t

1. **Books and TvSeriesDashboardController**

For BooksDashboardController and TvSeriesDashboardController class, they are exactly the same with the MoviesControllerClass and what’s different is that in get item list, instead of creating new Movies object it will create Books or TvSeries object instead and the string query will be different because in MoviesDashboardController, the string query will edit “movies” table but it will edit “books” table in BooksDashboardController, and it will edit “tvseries” table in TvSeriesDashboardController.

**Evidence Of working Program**

Every screenshot is taken from my IDE and taken by myself

****

**Summary and Self Reflection**

This project is a fun project to do since I also learn more about databases and SQL language. It is also a challenging project for me to work on but it is actually enjoyable and it allows me to understand more about object oriented programming in Java language.

**Resource and Reference**

Reference for some of the code and tutorial for this project:

<https://www.youtube.com/watch?v=CGWRwpeihE8&t=283s>

<https://www.youtube.com/watch?v=ejwzueIZo70>

<https://www.youtube.com/watch?v=rKv8eavrAio>

<https://www.youtube.com/watch?v=HJC_JxpHTeU>

<https://www.youtube.com/watch?v=5yQbt6lYRqk&t=708s>

<https://www.youtube.com/watch?v=J0IE5LRyzx8>