**Question 1- PDDA and JDBC**

**Instructions:**

⦁ Copy the database driver to your Netbeans project.

⦁ Run the movieFinderScript.txt script file to create and populate the mysql movie\_finderdb

database and movie table.

**Overall purpose**:

The relational database **movie\_finderdb** contains a table with movie details. A 3-tier design application with **Problem domain**, **Data Access** and **GUI** classes is required that allows the user to view the information about the stored movies, Update specified records, remove specified records and add newly released movies to the database by clicking on various buttons. Take the **Movie** class you modified in **question 1** to be your problem domain class.

1. Create the data access class named **MovieDA** and in it write the following methods:
   * A method **initialise()** for connecting to the movie\_finderdb.
   * A method **terminate()** for closing the connection established by the Initialise() method
   * Method **getAll()** to return all the movies in the database table using an arraylist.
   * Method **returnByDirector()** to retrieve and return all the movies by a specified director. The method receives all or part of the director's name as a parameter.
   * **returnByTitle()** to return a movie given all or part of the title as a parameter.
   * Return all the movies that were released after a specified release year using an arraylist. The release year is sent as a parameter to the method **producedAfter()**.
   * Return all the movies that are produced before a specified release year using an arraylist. The release year is sent as a parameter to the method **producedBefore()**.
   * Method **updatePrice()** reduces the prices of all the movies released before 2000 whose prices are more than R120 using a specified percentage. The percentage is sent as a parameter to the method. For example if the user wants to reduce the prices by 50% they will send 50 as a parameter.
   * Method **DeleteMovie()** detetes a movie whose title is sent as a parameter to the to it.
   * **RetrieveTitles()** that returns all movie titles in an arraylist.
   * Method **AddBook()** adds a new movie to the data storage
   * Method **calculateAveragePrice()** that calculates and returns the average.

**GUI - Specific requirements.**

1. Create the GUI shown in **Figure 1.**

* When the form first loads, only the three panels shown in **Figure 2** should be displayed and the combo box should be populated with all the titles of the movies in the database**.** Create a method to hide the controls and call it in the form’s constructor. Create a method to populate the combo box.
* To view the movies stored in the database, the user needs to select the title of the movie from the combo box. As soon as the title is selected from the combo box, the details about the movie should be displayed in the text fields (see **Figure 3**). Write code for this.
* When the **View Movies by Release Year** button is clicked, the panel shown in **Figure 4** is displayed. The user can then enter the desired year in the text field and then click on the **Show All** button. When the Show All button is clicked, check if the release Year text field is not empty and that the year is valid, accept the release Year from the text field, and display all the movies produced before and after the specified release Year **(see Figure 5**) by invoking the appropriate methods. If the text field is empty or the year entered is invalid display an appropriate message to the screen and set focus to that text field.
* The **Calculate average button** calculates and displays the average price of the movies stored in the database and display on a message box.
* The **Search movie by director/title** button displays the panel with title, **Search movies by Title or Director** as shown in **Figure 6**. In this panel, when the **GetMovies** button is clicked, check which radio button is checked, get the user input from the text field and invoke the appropriate method and display the movie data for the specified director or title. Display an error message if the user clicks on the GetMovies without entering anything in the text field.
* The **Add new movie** button displays the **Save button, the title text field and its identifying label and disables the combo box** when clicked . When the **Save** button is clicked, check if any of the text fields are empty, accepts all movie details from their text fields, instantiate an object and invoke the method that copies data to the database. If any of the text fields is empty display an appropriate error message to the screen and set focus to that text field.
* When the **Edit movie details** button is clicked, display the Update operation panel. When the **update** button is clicked, check if the percentage text field is not empty, accept the percentage rate as an integer from the text field, call the appropriate method to update the movie price. Otherwise display an appropriate error message to the screen.
* The **Remove movie** button when clicked, displays an input box for the user to enter the title of the movie to be deleted. The button displays a dialog confirmation message box with Yes/No buttons for the user to confirm the deletion , if the Yes option is selected, the delete method is invoke otherwise cancel deletion operation

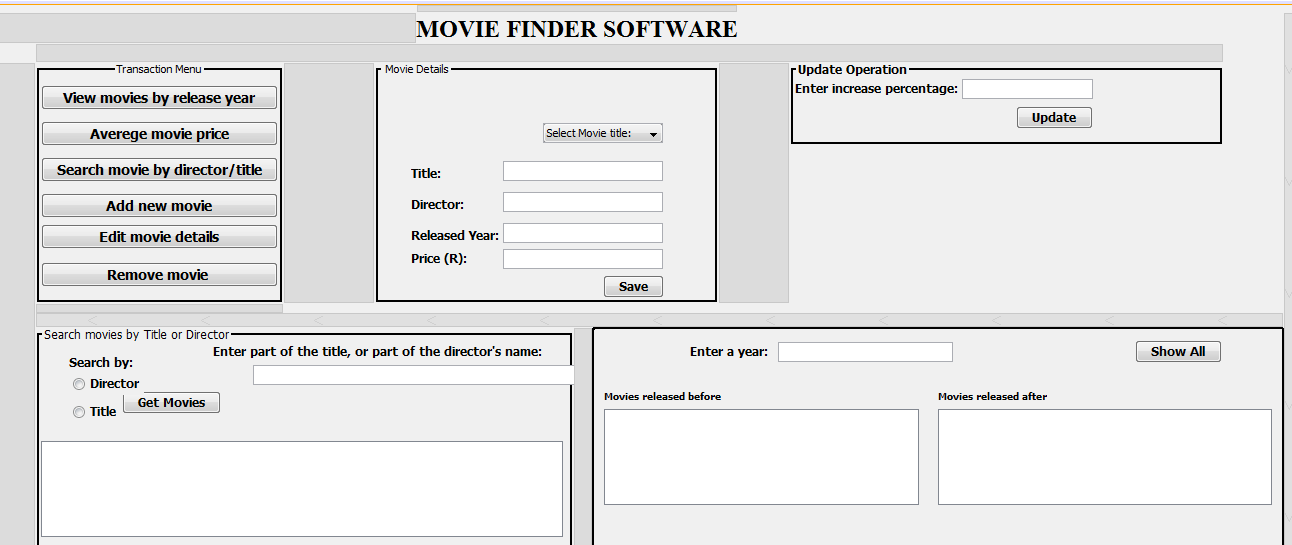


Figure 1

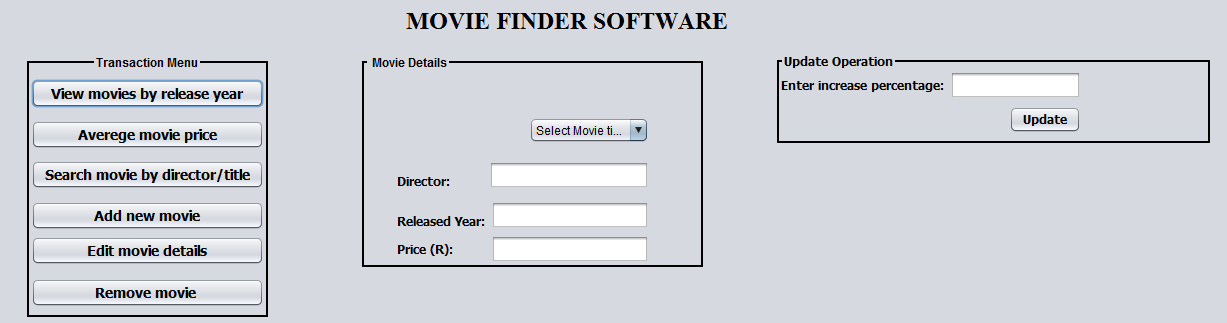
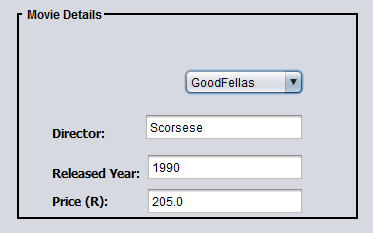
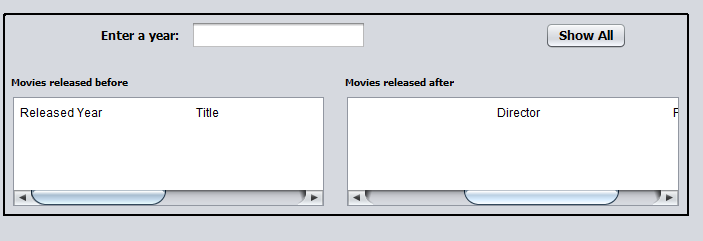
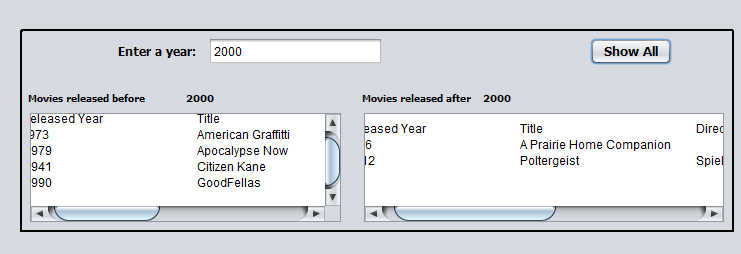
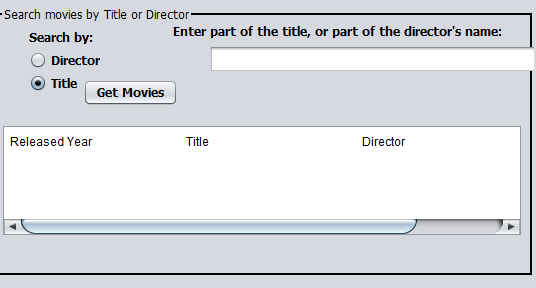


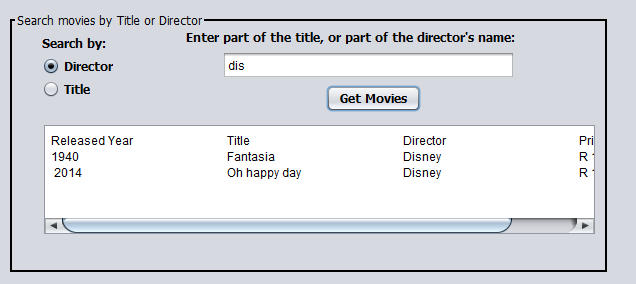
Figure 2

Figure 3

Figure 4

Figure 5

Figure 6

Figure 7