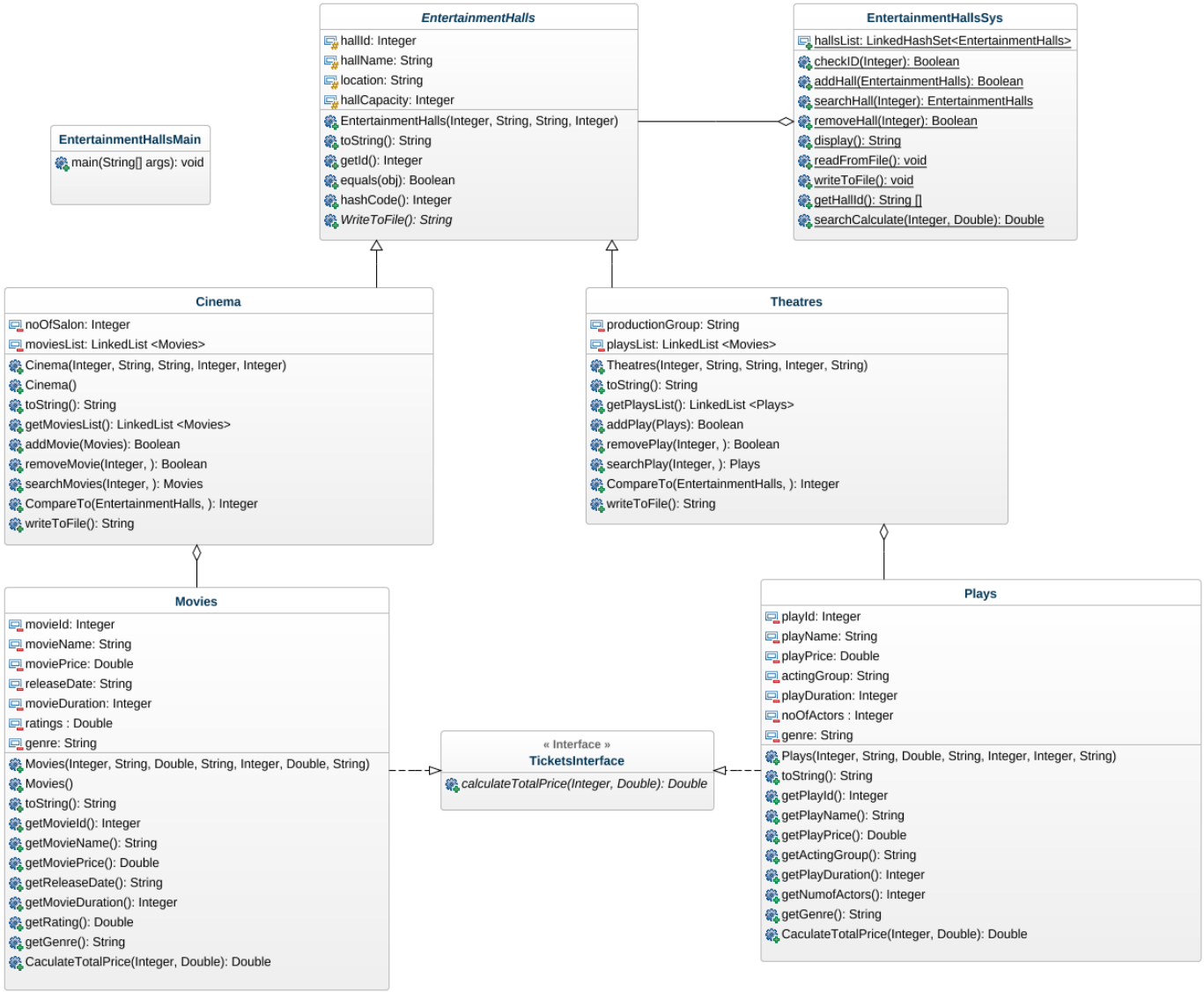
Team 14 : Project Step 4

The Movie Handler

Syed Abdullah Hassan

Elchin Latifli

Suheera Tanvir



Information of the abstract EntertainmentHalls class structure:

- EntertainmentHalls() : a non-default constructor for this class
- toString() : returns String
- getId : the getter for returning ID
- Equals & hashcode : methods for the Hash Set
- Abstract method writeToFile() : an abstract method which contains the code for writing the updated contents of the structures back onto the original file. This method will be implemented in Cinemas and Theatres classes.

Information of the Cinema class structure:

- Has **Is-A relationship** with *EntertainmentHalls.*
- Contains a Linked List of objects of type “movies”.
- Cinema(int, String, String, int, int) : The non-default constructor for Cinema
- Cinema() : the default constructor for Cinema
- toString() : returns String
- getMoviesList() : getter for returning the Movies type Linked List
- addMovie() : boolean method to add an object to the MovieList Linked List. Will be used in GUI. First takes the MovieId as input from the user(in the GUI part). Checks if a movie of that ID already exists in the List. If it exists, sends an appropriate output and returns false. If the ID does not exist, will take the remaining data members of the “movies” class from the user as input, create an object using those inputs and then add that object to the Linked List, then return True.
- removeMovie(int) : boolean method to remove an object of the type “Movies” from the movieList Linked List. Will take ID of the movie to be removed as a parameter, and search for the movie with that ID. If a movie of that ID does not exist, return false and give appropriate output message. If movie ID exists, delete the object from the List and return True.
- searchMovie(int) : method of return type Movies. Takes MovieId as parameter and searches for the movie object of that ID. If found, returns the object. If not found, returns NULL.
- compareTo(EntertaintaimentHalls) : the comparator method used for sorting the IDs
- writeToFile : method that returns a String consisting of all the elements of the moviesList. This method will be called in the writeToFile method in the system class.

Information of the Theatres class structure:

- Has **Is-A relationship** with *EntertainmentHalls.*
- Contains a Linked List of objects of type “plays”.
- Theatres(int, String, String, int, String) : The constructor for Theatres
- toString() : returns String
- getPlaysList() : getter for returning the Plays type Linked List
- addPlay() : boolean method to add an object to the PlaysList Linked List. Will be used in GUI. First takes the PlaysId as input from the user(in the GUI part). Checks if a play of that ID already exists in the List. If it exists, sends an appropriate output and returns false. If the ID does not exist, will take the remaining data members of the “Plays” class from the user as input, create an object using those inputs and then add that object to the Linked List, then return True.
- removePlay(int) : boolean method to remove an object of the type “Plays” from the playsList Linked List. Will take ID of the play to be removed as a parameter, and search for the play with that ID. If a play of that ID does not exist, return false and give appropriate output message. If play ID exists, delete the object from the List and return True.
- searchPlay(int) : method of return type Plays. Takes playId as parameter and searches for the play object of that ID. If found, returns the object. If not found, returns NULL.
- compareTo(EntertaintaimentHalls) : the comparator method used for sorting the IDs
- writeToFile() : method that returns a String consisting of all the elements of the playsList. This method will be called in the writeToFile method in the system class.

Information of the Movies and Plays class structure:

- Have a **Has-A relationship** with Cinemas and Theatres respectively
- Movies(String, String, int, double, String) **&** Plays(String, String, int, int, String) : Constructors for the objects of movies and plays respectively.
- toString() : returns String
- getters for every data member in each class : will be used in the writeToFile() methods in the superclass to return all data members
- CalculateTotalPrice() : the implementation of the method in the abstract class.

Information of the TicketsInterface structure:

- Contains abstract method *calculateTotalPrice(int).*

Information of the CalculateTotalPrice() method which is implemented from both Movies and Plays classes:

- *calculateTotalPrice(int) :* this function will take the number of tickets(this will be a global variable in main) that an individual wishes to buy of a particular movie or play as a parameter, and then will calculate the total price that individual has to pay. The total price will be "number of tickets * price(price is mPrice/pPrice of that specific movie/play respectively)(will be passed in the function using the getters getMoviePrice()/getPlayPrice().
  This function will also calculate discounts:
  For movies
  1) if you are buying 5 - 9 tickets, then 10% discount on the total price.
  2) if you are buying 10 - 15 tickets, then 15% discount
  3) if you are buying more than 15 tickets, then 20% discount.

  For plays
  1) if you are buying 5 - 9 tickets, then 15% discount on the total price.
  2) if you are buying 10 - 15 tickets, then 25% discount
  3) if you are buying more than 15 tickets, then 30% discount.

  The function will return a double value of total price that the individual has to pay after making all necessary calculations.

Information of the EntertainmentHallsSys class structure:

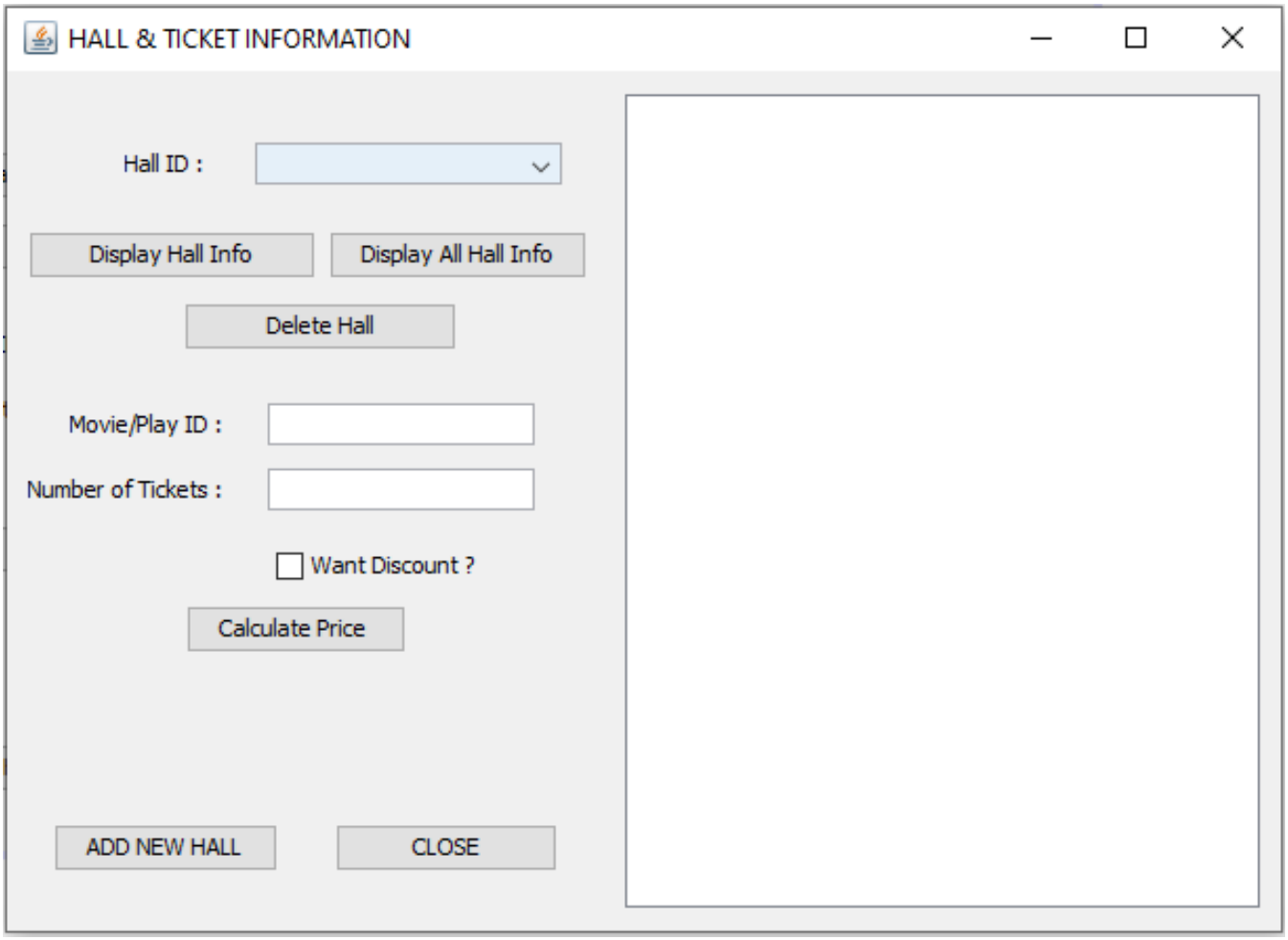
- Contains a Linked Hash Set of objects of type “EntertainmentHalls”
- checkID(int) : takes a hall ID as an input from the user, passes it as a parameter and then checks the HallList array to see if an EntertainmentHall object of the ID is present in the List. Returns true if ID is present, else returns false.

- addHall() : Takes hallId as an input from the user. Uses checkID() method to check if a Hall with that ID is present in the HallList Linked List. If it is present, returns false with an appropriate output message. If it is not present, creates a Hall object after taking other members from the user as inputs, and adds the object to the HallList Linked List.
- removeHall(int) : Takes hallId as an input from the user. Uses checkID() method to check if a Hall with that ID is present in the HallList Linked List. If it is not present, returns false with an appropriate output message. If it is present, removes the object from the HallList Linked List.
- searchHall(int) : Takes hallId as an input from the user. Uses checkID() method to check if a Hall with that ID is present in the HallList Linked List. If it is present, returns the object, otherwise returns NULL.
- display() : Displays ALL the contents of the HallList, along with all the contents of the Linked Lists (moviesList and playsList) of each individual Hall.
- readFromFile() & writeToFile() : will read the information regarding entertainment halls and movies/plays from a text file. The write to file method will write the necessary information in the same text file after the program is closed.
- getHallId() : will return a string array consisting of EntertainmentHall IDs.
- searchCalculate(int, int) : this method will take the play/movie id and the noOfTickets as parameters, traverse through each EntertainmentHall object in within that object, traverse through the respective lists to search for which movie or play has the ID that matches the parameter. After the first ID has been found, this method will take the price of that movie/play and then input the price and the noOfTickets into the CalculateTotalPrice() function. In the end, it will return the total price as returned by the CalculateTotalPrice() function.

Information of the Main Class(EntertainmentHalls main):

- Reads data from file
- Opens the main frame.

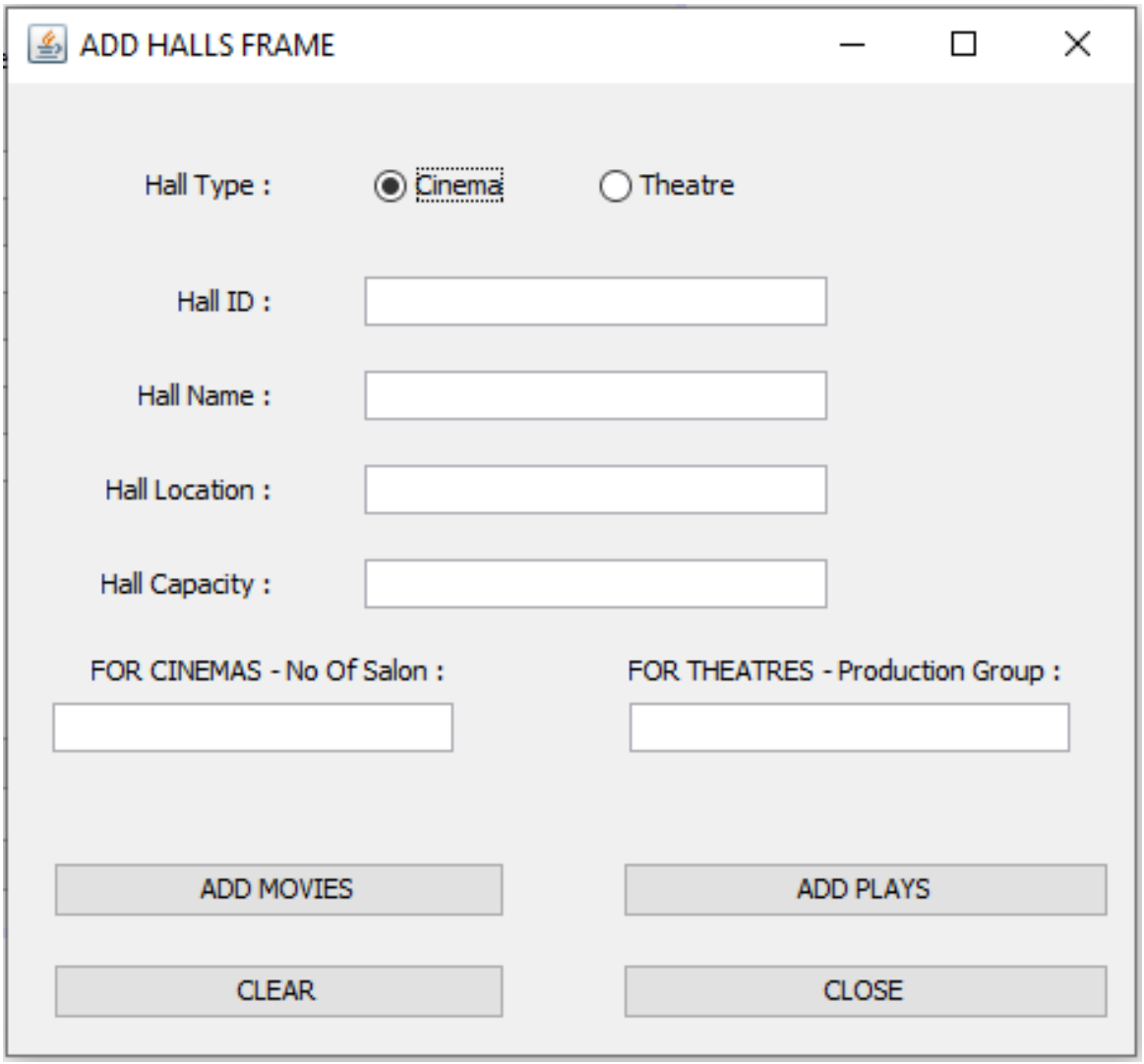
GUI PART :



This is the MainFrame which opens when the user runs the project :

- **Click Event :** The text area on the right is uneditable and if the user clicks on it, a label under the "calculate Price" Button shows the prompt "Sorry, but this Text Area cannot be edited".
- When the user clicks on the "Display All Hall Info" button, the Text Area shows the complete set of HallList objects, along with their corresponding moviesLists or playsLists.
- The Hall ID comboBox has the first hall ID selected by default. This combo Box contains a list of all the hall IDs.
- When the user clicks on the "Display Hall Info" button, the information regarding the selected comboBox Hall ID (including the corresponding moviesList or playsList) is displayed in the Text Area.
- When the user clicks on the "Delete Hall" button, the object that corresponds to the selected comboBox Hall ID is deleted from the HallsList structure, and a prompt that says "Hall has been deleted" is shown.

- When the user clicks on the “Calculate Price” button :
    1. First the program will check whether both fields(Movie/Play ID and Number of Tickets) are filled or not. If either or both are not filled, it will display the prompt "PLEASE FILL IN BOTH FIELDS FIRST !"
    2. Then, the program will check if the “Want Discount ?” checkbox has been selected or not. If it has, the program will calculate the total price by calling the searchCalculate() method.
    3. If checkbox is not selected, the program will search for the price of the movie/play and calculate the price by the normal formula noOfTickets * price
    4. In both above situations, if the movie/play ID does not exist in the system, the totPrice will be set to -1
    5. If totPrice is -1, a prompt will be given "A MOVIE/PLAY WITH THIS ID DOES NOT EXIST!"
    6. If totPrice has been calculated successfully, a prompt "The customer needs to pay $ + totPrice + ." will be given
- When the user clicks on the “Add New Hall” button, the MainFrame will be closed, a new object for the AddHalls frame will be created and opened.
- When the user clicks on the “Close” button, all the current data in the structures will be written to the file and the MainFrame will close.

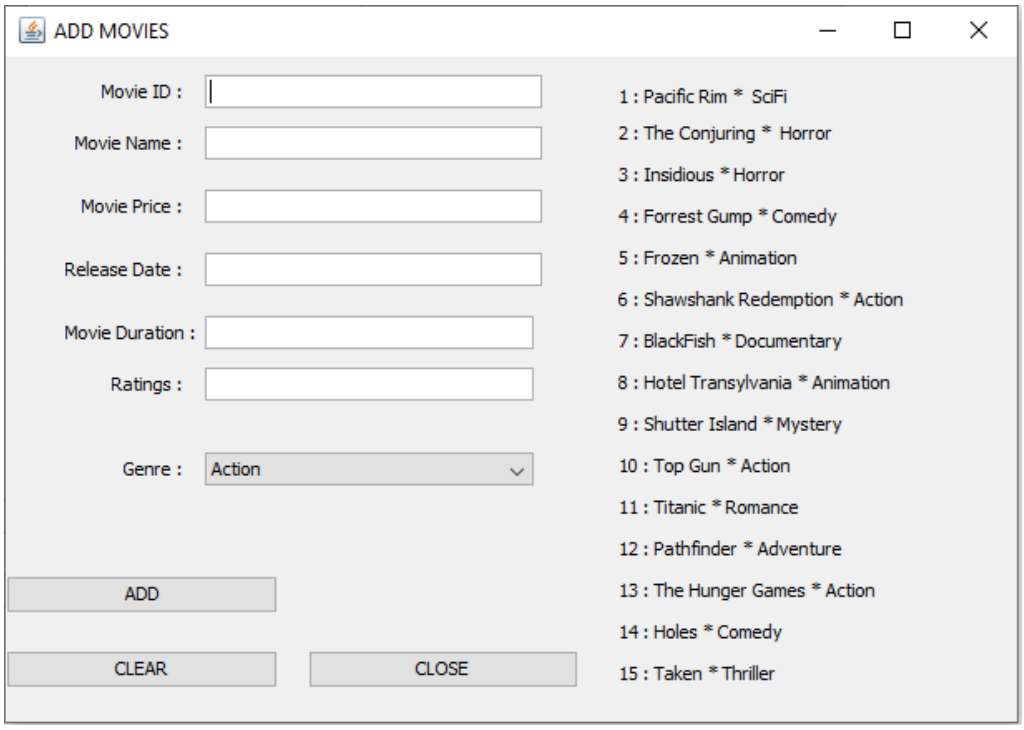


This is the AddHalls Frame

- When adding a hall, only one radio button, either Cinema or Theatre may be selected.
- When the user clicks on the “Clear” button, all the values currently in the textfields will be erased.
- When the user clicks on the “Add Movies” button:
    1. The program will first check which radio button is selected. If Theatres is selected, a prompt "A Theatre cannot add Movies!" (in a label right under the no of Salon and Production Group textFields)will be shown. Otherwise, will continue
    2. Then, the program will check if any fields have been left empty. If any field is empty, a prompt "PLEASE FILL IN ALL THE FIELDS FIRST " will be shown
    3. Then, the program will search the Hall ID listed to check if a Hall with this ID already exists. If it does, a prompt "A Hall with this ID already exists !" will be shown

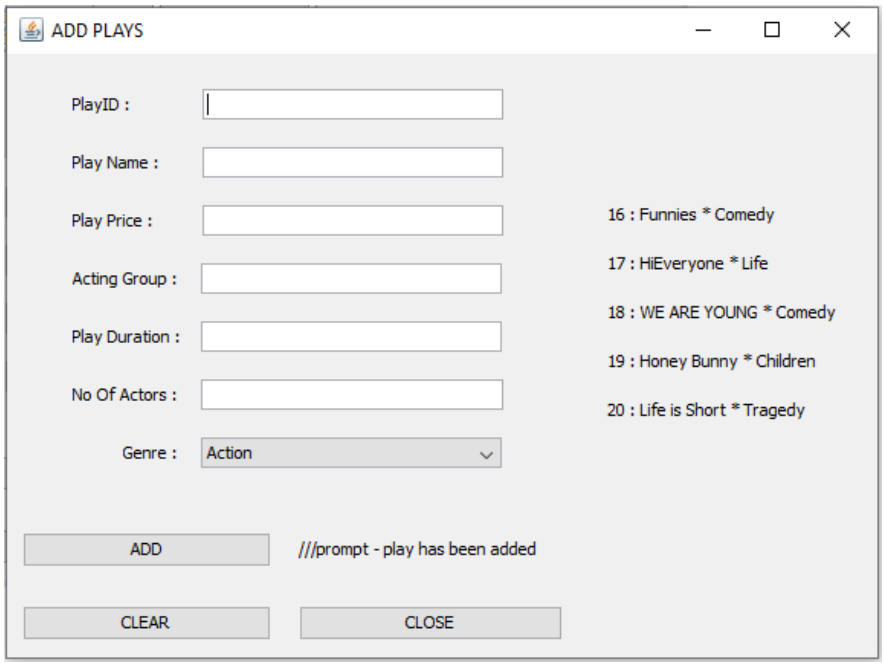
4. Finally, if all else is normal, the program will take the necessary information from the textFields, create a hall object and add that information to the hall object it created
5. The new hall ID will also be added to the Hall IDs comboBox in the mainframe
6. Then, the current frame will close and the AddMovies frame will open

- When the user clicks on the “Add Plays” button:
    7. The program will first check which radio button is selected. If Cinema is selected, a prompt "A Cinema cannot add Plays!" !" (in a label right under the no of Salon and Production Group textFields) will be shown. Otherwise, will continue
    8. Then, the program will check if any fields have been left empty. If any field is empty, a prompt "PLEASE FILL IN ALL THE FIELDS FIRST " will be shown
    9. Then, the program will search the Hall ID listed to check if a Hall with this ID already exists. If it does, a prompt "A Hall with this ID already exists !" will be shown
    10. Finally, if all else is normal, the program will take the necessary information from the textFields, create a hall object and add that information to the hall object it created
    11. The new hall ID will also be added to the Hall IDs comboBox in the mainframe
    12. Then, the current frame will close and the AddPlays frame will open
- When the user clicks on the “Close” button, the AddHalls frame will close and the MainFrame will open.



This is the AddMovies Frame. Movies with their IDs already saved in the system are written for reference.

- **<u>On Key Event :</u>** When user writes the Movie ID and *presses **Enter,*** the program will search if a movie with that ID is already in the system. If present, the remaining text fields will automatically fill with the information of that movie. If a movie with that ID is not found, nothing will happen and the user will have to input the data manually.
- When the user clicks on the “Clear” button, all the values currently in the textfields will be erased.
- When the user clicks on the “Add” button:
    1. The program will first check if all the text fields are filled. If any text field is not filled, a prompt "Please Fill in ALL the fields! " will be given in a label to the right of the ADD button
    2. Then , the program will search the current hall object’s moviesList to see if a movie with the given ID already exists in the list. If it exists, a prompt "A movie with this ID already exists!" will be given
    3. Otherwise, a movie object will be created with the data members from the textfields, and added to the moviesList of the hall that was created in the previous frame. Then, a prompt "movie has been added" will be given
- When the user clicks on the “Close” button, the AddMovies frame will close and the AddHalls Frame will open.



This is the AddPlays Frame. Plays with their IDs already saved in the system are written for reference.

- **<u>On Key Event :</u>** When user writes the Play ID and *presses **Enter,*** the program will search if a play with that ID is already in the system. If present, the remaining text fields will automatically fill with the information of that play. If a play with that ID is not found, nothing will happen and the user will have to input the data manually.
- When the user clicks on the “Clear” button, all the values currently in the textfields will be erased.
- When the user clicks on the “Add” button:
    4. The program will first check if all the text fields are filled. If any text field is not filled, a prompt "Please Fill in ALL the fields! " will be given in a label to the right of the ADD button
    5. Then , the program will search the current hall object’s playsList to see if a play with the given ID already exists in the list. If it exists, a prompt "A play with this ID already exists!" will be given
    6. Otherwise, a play object will be created with the data members from the textfields, and added to the playsList of the hall that was created in the previous frame. Then, a prompt "movie has been added" will be given
- When the user clicks on the “Close” button, the AddPlays frame will close and the AddHalls Frame will open.