**Java Cinema Management project**

**General Summary:**

This project is designed in Java programming language using JavaFx framework and Object-oriented programming features. The data used in the program, for example, movies, user information, tickets and payment information are kept in separate files.

When the program is newly executed, a login or registration screen will appear in front of the user. The Sign-Up screen provides the opportunity to create a new user (Customer) account. The phone and e-mail address entered when adding a new user should not be registered already for another user in the system.

There are two users in the program, Customer and Administrator. All movies available in the system are displayed on the home page of the user logging in with the customer account. The customer can buy a ticket, choose his seat in the movie theater. when buying a ticket customer can pay with cash and card. Also, if the customer has already purchased a movie ticket, they can view their ticket. In the application, a search button has been added for the customer. User can search for movies by Movie name, movie type, movie release date and movie language.

On the home page of the user logged in as an admin, all the movies available in the system are displayed. The administrator can also view the purchased payments in addition to adding and deleting movies to the system. When adding a movie, the user is expected to enter the movie name, language, release year, type, ticket price, show date, movie start and end time. In addition to these, the ability to add movie photos to the manager when adding movies is also defined.

**Definition:**

Classes and methods used throughout the project:

• Person class:

o Variables:

* name: Defined to hold the username.
* email: defined to hold user email address.
* phoneNumber: defined to hold user phone number.
* account: (another class defined) defined to hold user account information.

o Methods:

* readAllUsers(): returns all users on the system as an ArrayList.

• Admin class (inherits the Person class):

o Variables:

* Inherits all variables of the Person class.

o Methods:

* readAdmins(): It returns all the admin users registered in the system as an ArrayList.
* getType(): returns the user's type as a String.
* It also inherits all methods of the Person class.

• Client class (inherits the Person class):

o Variables:

* Inherits all variables of the Person class.

o Methods:

* addClient(): It is a method of adding a new user (Customer) to the system.
* readClients(): ArrayList of all Client users registered in the system.
* getType(): returns the user's type as a String.

• Account class (used by your Person class):

o Variables:

* userName: It is defined to hold the user name in order to log in to the system.
* email: defined to hold user email address.
* password: It is defined to keep the password of the user in order to log in to the system.

Diagram

Description automatically generated

• Film class:

o Variables:

* name: defined to hold the movie name.
* language: defined to keep the movie language.
* type: defined to hold the movie type.
* releasedDate: defined to hold the movie release date.
* price: defined to hold the movie price.
* imgSrc: defined to hold the path of the movie photo.
* show: (another class defined) is defined to hold movie display information.
* hall: (another class defined) defined to hold the information of the movie theater.

o Methods:

* returnSelectedFilm(): returns the single movie information selected by the user throughout the system.
* searchFilmByName(): returns a searched movie by movie name.
* searchFilmByType(): It returns all the movies that exist in the system as an ArrayList according to the movie type entered.
* searchFilmByLanguage(): It returns all the movies that exist in the system as an ArrayList according to the movie type entered.
* searchFilmByReleasedDate(): It returns all existing movies in the system as ArrayList according to the movie type entered.
* addFilm(): It is a method of adding a new movie to the system.
* validateFilmForAddition(): checks whether it is suitable for adding a new movie to be added (as if there should not be a movie in the system with the same name).
* deleteFilm(): It is the method used to delete movies from the system.

• Show class (used by your movie class):

o Variables:

* showDate: Defined to keep the movie showing date.
* startTime: Defined to keep the display start time.
* endTime: Defined to hold the display start time.

• CinemaSalon class (used by your movie class):

o Variables:

* A picture containing diagram

  Description automatically generatedName: defined to keep the name of the movie theater.

• Booking class:

o Variables:

* clientName: defined to hold the name of the user who bought the ticket for the movie.
* movieName: defined to hold the name of the movie for which the ticket was purchased.
* seat: defined to keep the reserved seat information in the movie theater.
* date: defined to keep the movie showing date.
* time: defined to keep the movie start time.
* phoneNumber: defined to hold the phone number of the user who bought the ticket.

o Methods:

* addBooking(): saves the information to the file when the user buys a ticket.
* Shape

  Description automatically generated with medium confidencegetClientBookings(): returns an ArrayList of information of tickets purchased by a particular user.

A picture containing diagram

Description automatically generated

• Seat class:

o Variables:

* name: defined to hold the name of each seat in the movie theater.
* isBooked: defined to check whether the seat is reserved or not.

o Methods:

* getBookedSeats(): returns an ArrayList of seats reserved for a particular movie.

• Payment class:

o Variables:

* Id: an id is assigned for each payment.
* Amount: represents the payment amount.
* paymetnDate: represents the payment date.
* paymentStatus: represents the payment status.
* payer: represents the name of the payer.

o Methods:

* getAllPayments(): Returns all received tickets and payment information to the manager as an ArrayList.
* Diagram

  Description automatically generatedaddPayment(): When the customer buys a ticket, it saves the payment information to the system.

Diagram

Description automatically generatedAll classes