Movie Ticket Booking Simulator (CLI)

Phase-End Project Problem Statement

Project Agenda:

Create a Movie Ticket Booking Simulator using command line interface, core java concepts, and data structures in Java.

Description:

This project is based on the core java concept such as, oops concepts, exception handling, IO handling, and focused on how to use data structures concepts in your project using Java collections.

The project is only created around the command line interface and focused on the implementation of data structure and java oops concepts.

For the sake of simplicity, it is considered that the theatre is single screen theatre and user used to prefer ticket booking from the ticket window (front desk).

Project Features:

- Front desk can login to theatre using username or password.
- Front desk can update password.
- Front desk can view the seating arrangement by entering the date and upcoming show time.
- Front desk book the ticket for a date and showtime by entering the seat selection.
- Front desk can view the auto calculated amount and book the ticket.
- Front desk can check or enquire the booking status anytime.

Detailed Scenario:

- Customer visits a theatre.
- Go to the front desk for booking a ticket.
- Front desk asks the customer for the booking date and show time.
- Then front desk can check the ticket availability by looking at the seating arrangement.
- Front desk asks the customer for the preferred seat selection and enters the seat number(B1,B2,B6) or (B1-B5).
- After the seat selection, front desk can view the total amount that should be paid by customer.
- Front desk asks the customer for payment and confirm the booking.

Source Code :

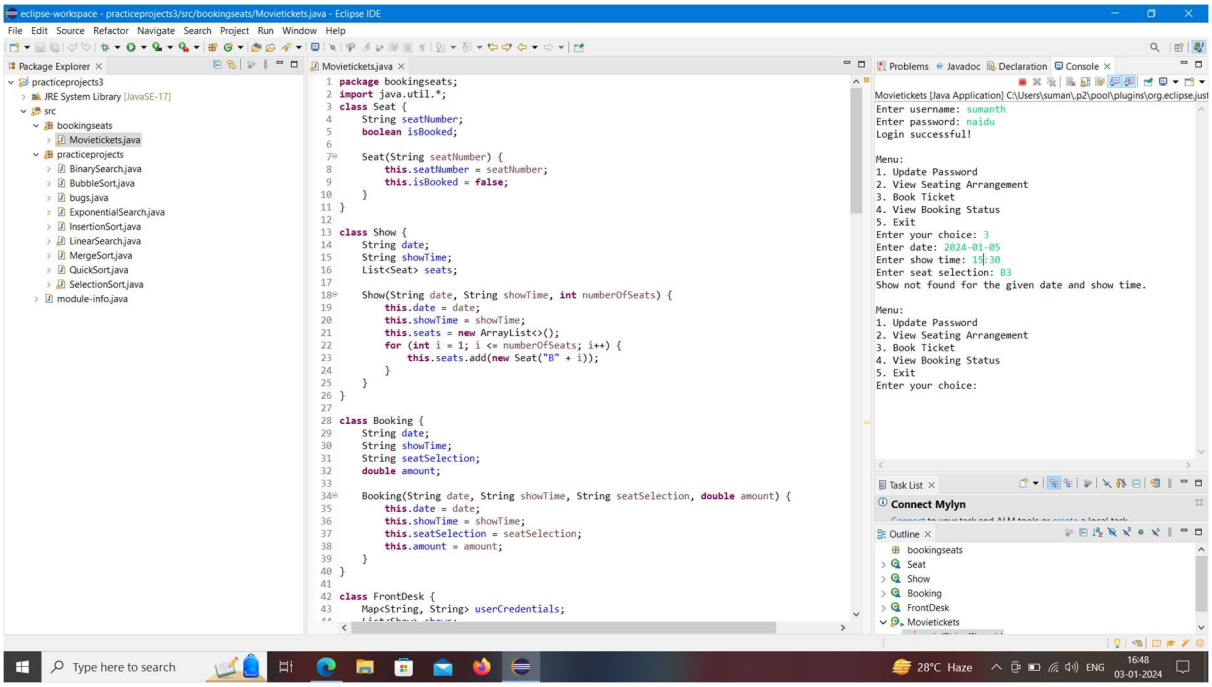
```
package bookingseats;
import java.util.*;
class Seat {
    String seatNumber;
    boolean isBooked;

    Seat(String seatNumber) {
        this.seatNumber = seatNumber;
        this.isBooked = false;
    }
}

class Show {
    String date;
    String showTime;
    List<Seat> seats;

    Show(String date, String showTime, int numberOfSeats) {
        this.date = date;
        this.showTime = showTime;
        this.seats = new ArrayList<>();
        for (int i = 1; i <= numberOfSeats; i++) {
            this.seats.add(new Seat("B" + i));
        }
    }
}

class Booking {
    String date;
    String showTime;
    String seatSelection;
    double amount;

    Booking(String date, String showTime, String seatSelection, double amount) {
        this.date = date;
        this.showTime = showTime;
        this.seatSelection = seatSelection;
        this.amount = amount;
    }
}

class FrontDesk {
    Map<String, String> userCredentials;
    List<Show> shows;
    List<Booking> bookings;

    FrontDesk() {
        this.userCredentials = new HashMap<>();
        this.userCredentials.put("sumanth", "naidu"); // credentials
        this.shows = new ArrayList<>();
        this.bookings = new ArrayList<>();
    }

    boolean login(String username, String password) {
        return userCredentials.containsKey(username) &&
userCredentials.get(username).equals(password);
    }
```

```
    void updatePassword(String username, String newPassword) {
        userCredentials.put(username, newPassword);
        System.out.println("Password updated successfully.");
    }

    void viewSeatingArrangement(String date, String showTime) {
        for (Show show : shows) {
            if (show.date.equals(date) && show.showTime.equals(showTime)) {
                for (Seat seat : show.seats) {
                    System.out.print(seat.seatNumber + "(" + (seat.isBooked ? "X"
: "O") + ") ");
                }
                System.out.println();
                return;
            }
        }
        System.out.println("Seating arrangement not found for the given date and
show time.");
    }

    void bookTicket(String date, String showTime, String seatSelection) {
        for (Show show : shows) {
            if (show.date.equals(date) && show.showTime.equals(showTime)) {
                for (Seat seat : show.seats) {
                    if (seat.seatNumber.equals(seatSelection) && !seat.isBooked) {
                        seat.isBooked = true;
                        double amount = calculateAmount(showTime); // You need to
implement this method
                        bookings.add(new Booking(date, showTime, seatSelection,
amount));
                        System.out.println("Ticket booked successfully. Amount: $"
+ amount);
                        return;
                    }
                }
                System.out.println("Seat already booked or not available.");
                return;
            }
        }
        System.out.println("Show not found for the given date and show time.");
    }

    void viewBookingStatus() {
        if (bookings.isEmpty()) {
            System.out.println("No bookings available.");
            return;
        }
        System.out.println("Booking Status:");
        for (Booking booking : bookings) {
            System.out.println("Date: " + booking.date + ", Show Time: " +
booking.showTime +
                    ", Seat: " + booking.seatSelection + ", Amount: $" +
booking.amount);
        }
    }

    private double calculateAmount(String showTime) {
```

```
        // Implement your logic to calculate the ticket amount based on the show
time.
        // For simplicity, let's assume a fixed amount for now.
        return 10.0;
    }
}

public class Movietickets {
       public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        FrontDesk frontDesk = new FrontDesk();

        System.out.print("Enter username: ");
        String username = scanner.next();
        System.out.print("Enter password: ");
        String password = scanner.next();

        if (frontDesk.login(username, password)) {
            System.out.println("Login successful!");

            while (true) {
                System.out.println("\nMenu:");
                System.out.println("1. Update Password");
                System.out.println("2. View Seating Arrangement");
                System.out.println("3. Book Ticket");
                System.out.println("4. View Booking Status");
                System.out.println("5. Exit");
                System.out.print("Enter your choice: ");
                int choice = scanner.nextInt();

                switch (choice) {
                    case 1:
                        System.out.print("Enter new password: ");
                        String newPassword = scanner.next();
                        frontDesk.updatePassword(username, newPassword);
                        break;

                    case 2:
                        System.out.print("Enter date: ");
                        String date = scanner.next();
                        System.out.print("Enter show time: ");
                        String showTime = scanner.next();
                        frontDesk.viewSeatingArrangement(date, showTime);
                        break;

                    case 3:
                        System.out.print("Enter date: ");
                        date = scanner.next();
                        System.out.print("Enter show time: ");
                        showTime = scanner.next();
                        System.out.print("Enter seat selection: ");
                        String seatSelection = scanner.next();
                        frontDesk.bookTicket(date, showTime, seatSelection);
                        break;

                    case 4:
                        frontDesk.viewBookingStatus();
                        break;
```

```
                    case 5:
                        System.out.println("Exiting program. Thank you!");
                        System.exit(0);

                    default:
                        System.out.println("Invalid choice. Please try again.");
                }
            }
        } else {
            System.out.println("Login failed. Incorrect username or password.");
        }
    }

}
```

Output :



To create the Movie Ticket Booking Simulator using the command line interface, core Java concepts, and data structures, you can follow the steps outlined below. This project assumes a basic understanding of Java, object-oriented programming (OOP), and data structures.

Algorithm :

Step 1: Design Classes and Objects

Define classes such as Theatre, FrontDesk, Booking, Show, Seat, etc. Consider using proper encapsulation and abstraction to represent entities in the project.

Step 2: Implement Data Structures

Use Java collections for data structures. For example, use ArrayList to store bookings, HashMap for user authentication, etc.

Step 3: Implement Ticket Booking Logic

Implement methods in the FrontDesk class for logging in, updating password, viewing seating arrangement, booking tickets, checking booking status, etc.

Step 4: Implement Command Line Interface

Create a simple command line interface to interact with the user and call the methods in the FrontDesk class.

Step 5: Implement Exception Handling

Handle exceptions using try-catch blocks to ensure robust error handling.

Step 6: Compile and Run

Compile your Java classes and run the main program to simulate the movie ticket booking process.

Testing:

Testing is a critical phase to ensure the reliability and correctness of your Movie Ticket Booking Simulator. Consider creating test cases to cover various scenarios, including edge cases. Here's a basic outline for testing

Conclusion:

In conclusion, the Movie Ticket Booking Simulator is a basic yet functional program that demonstrates core Java concepts, object-oriented programming, and data structures. Throughout the development process, you've implemented features like login, password update, viewing seating arrangement, booking tickets, and checking booking status. The integration of data structures like lists and maps enhances the efficiency and organization of the simulator.