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.

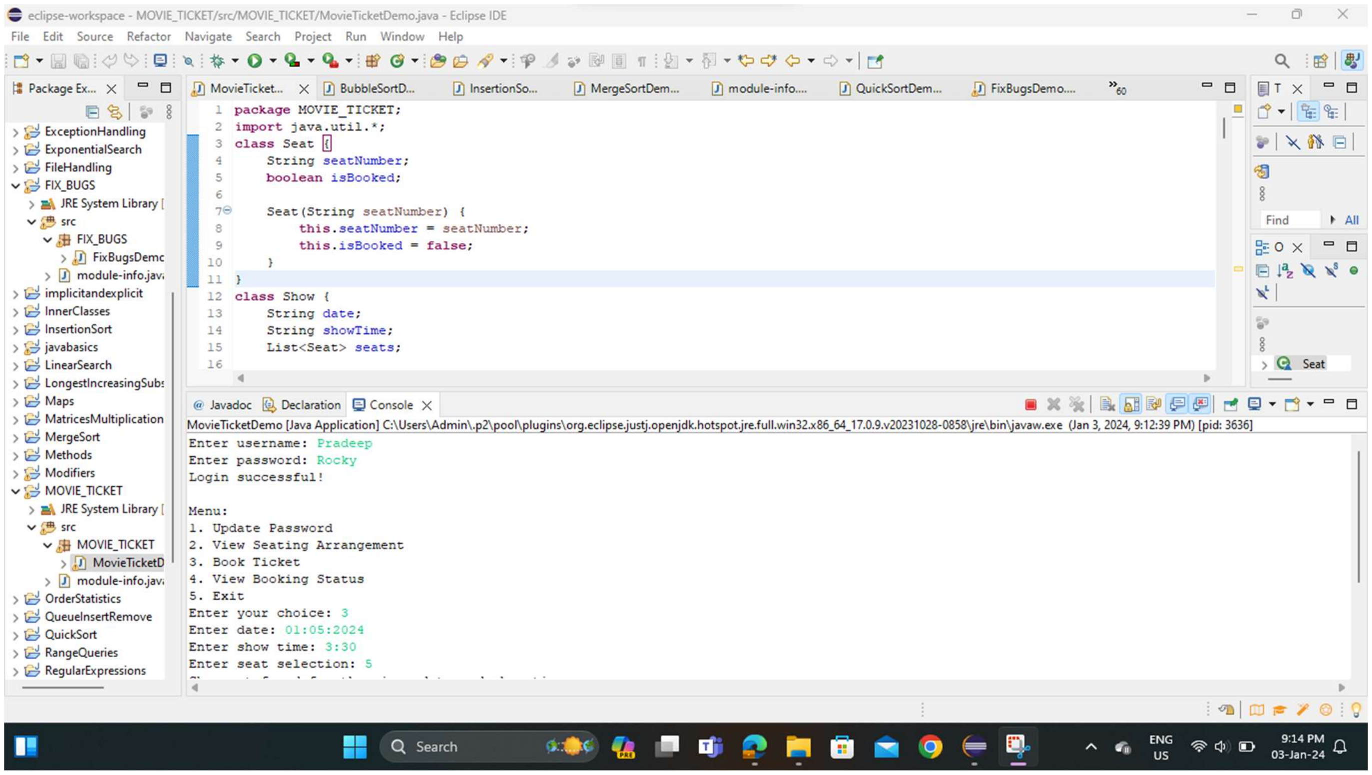
Source Code:

```java
package MOVIE_TICKET;
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("Pradeep", "Rocky"); // Default 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 MovieTicketDemo {
        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:



eclipse-workspace - MOVIE_TICKET/src/MOVIE_TICKET/MovieTicketDemo.java - Eclipse IDE

```
package MOVIE_TICKET;
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;
```

MovieTicketDemo [Java Application] C:\Users\Admin\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\javaw.exe (Jan 3, 2024, 9:12:39 PM) [pid: 3636]

```
Enter username: Pradeep
Enter password: Rocky
Login successful!

Menu:
1. Update Password
2. View Seating Arrangement
3. Book Ticket
4. View Booking Status
5. Exit
Enter your choice: 3
Enter date: 01:05:2024
Enter show time: 3:30
Enter seat selection: 5
```

Algorithm:

1. Start

2. Display the login screen:

    1. Ask for username and password

    2. If credentials are correct, proceed to the main menu

    3. If not, display an error message and ask again

3. Display the main menu with options:

    1. Update Password

    2. Show Seating Arrangement

```
   3. Book Tickets
   4. Check Booking Status
   5. Exit
4. If user selects 'Update Password':
   1. Ask for the old password
   2. If the old password is correct, ask for the new password and update it
   3. If not, display an error message and go back to the main menu
5. If user selects 'Show Seating Arrangement':
   1. Ask for the date and show time
   2. Display the seating arrangement for the given date and time
   3. Go back to the main menu
6. If user selects 'Book Tickets':
   1. Ask for the booking date and show time
   2. Display the seating arrangement for the given date and time
   3. Ask for the preferred seat selection
   4. If the selected seats are available, display the total amount and ask for payment
   5. If the payment is successful, confirm the booking and update the seating arrangement
   6. If not, display an error message and go back to the main menu
7. If user selects 'Check Booking Status':
   1. Ask for the booking ID
   2. Display the booking status
   3. Go back to the main menu
8. If user selects 'Exit', end the program
9. End
```

Description:

1)The Ticket Booking Simulator uses a HashMap to store the movie details and the number of tickets available for each movie. The HashMap is a part of Java’s collections framework and it stores key-value pairs.

2)The bookTicket method is used to book tickets for a specific movie.

It checks if the movie is available and if there are enough tickets for the user's request.

3)The main method is the entry point of the application. It creates an instance of the

MovieTicketBooking class and uses a Scanner to read the user's input from the command line.

4)At the output we need to enter the movie name and number of seats and view movies and select the movie we want to book a ticket for...

Introduction: The goal of this project is to create a command-line interface (CLI) application for a movie ticket booking system. The application will simulate the process of booking movie tickets at a theatre's front desk.

Requirements and Objectives: The application should allow the front desk to:

- Login using a username and password
- Update their password
- View the seating arrangement for a specific date and show time
- Book tickets by selecting seats
- View the auto-calculated amount for the booking
- Check or inquire about the booking status

Implementation Plan:

1. **Data Structures:** Use appropriate data structures like arrays or ArrayLists for storing movie details, seating arrangements, and bookings.
2. **Classes:** Create classes for Movie, Show, Seat, Booking, etc. to encapsulate related data and operations.
3. **User Interface:** Use the Scanner class for command line inputs and System.out.println for outputs.
4. **Authentication:** Implement a simple authentication system for the front desk login.
5. **Booking System:** Implement the logic for viewing seating arrangements, booking tickets, and checking booking status.

Testing and Deployment: Write unit tests for your methods and classes. Test the application thoroughly to ensure it works as expected. Once testing is complete, the application can be deployed.

Maintenance and Updates: Regularly update the application to add new features or fix bugs. Maintain the codebase by refactoring code when necessary and keeping the documentation up-to-date.

Conclusion: This project will help you understand the practical application of core Java concepts and data structures. It will also give you experience in developing a complete software solution from requirements gathering to deployment and maintenance.