Hotel Management System - MVC Architecture

Introduction

The Hotel Management System is designed using the Model-View-Controller (MVC) architecture. This architecture separates the application's concerns into three main components: Model, View, and Controller. This approach ensures a clean and scalable structure, making it easier to maintain and extend.

MVC Architecture

The MVC architecture divides the system into three interconnected components:
- **Model:** Handles the core business logic and data.
- **View:** Manages the presentation layer and interacts with the user.
- **Controller:** Acts as an intermediary between Model and View, processing user input.

Code Examples

1. Model

The Model encapsulates the application's data and business logic. For example, the `Customer`, `Room`, and `Hotel` classes define the data structure and implement business operations like room booking and payment processing.

Example Code for `Customer` class:

```java
public class Customer {
    private static int idCounter = 0;
    private final String name;
    private final String phone;
    private final String email;
    private final String address;
    private final String checkInDate;
    private final String checkOutDate;
    private Room bookedRoom;

    public Customer(String name, String phone, String email, String address, String
checkInDate, String checkOutDate) {
        idCounter++;
        this.name = name;
        this.phone = phone;
        this.email = email;
        this.address = address;
        this.checkInDate = checkInDate;
        this.checkOutDate = checkOutDate;
```

```
    }

    // Getter and Setter Methods
}
```

Example Code for `Room` class:

```java
public class Room {
    private final int roomNumber;
    private final String roomType;
    private final double price;
    private boolean isBooked;

    public Room(int roomNumber, String roomType, double price) {
        this.roomNumber = roomNumber;
        this.roomType = roomType;
        this.price = price;
        this.isBooked = false;
    }

    public void book() {
        this.isBooked = true;
    }

    public void checkout() {
        this.isBooked = false;
    }
}
```

Example Code for `Hotel` class:

```java
public class Hotel {
    private final String name;
    private final List<Room> rooms = new ArrayList<>();
    private final List<Customer> customers = new ArrayList<>();

    public Hotel(String name) {
        this.name = name;
        initializeRooms();
    }
```

```
    private void initializeRooms() {
        rooms.add(new Room(101, "Standard Non-AC", 3000));
        rooms.add(new Room(102, "Standard AC", 3500));
        rooms.add(new Room(103, "3-Bed Non-AC", 4500));
        rooms.add(new Room(104, "3-Bed AC", 5000));
    }

    public void bookRoom(Customer customer, int roomTypeChoice) {
        for (Room room : rooms) {
            if (!room.isBooked() && mapRoomTypeToChoice(room.getRoomType()) ==
roomTypeChoice) {
                room.book();
                customer.setBookedRoom(room);
                customers.add(customer);
                System.out.println("Room booked successfully!");
                return;
            }
        }
        System.out.println("No available rooms of selected type.");
    }
}
```

2. View

The View component is responsible for displaying options to the user and collecting their input. It serves as the interface between the user and the system.

Example Code for `HotelView` class:

```java
public class HotelView {
    public int showMainMenu() {
        System.out.println("\n1. Booking");
        System.out.println("2. Rooms Info");
        System.out.println("3. Payment");
        System.out.println("0. Exit");
        System.out.print("Please Enter Your Choice: ");
        return new Scanner(System.in).nextInt();
    }

    public Customer collectCustomerData() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter Customer Name: ");
        String name = scanner.nextLine();
```

```
        System.out.print("Enter Phone: ");
        String phone = scanner.nextLine();
        return new Customer(name, phone, "", "", "", "");
    }
}
```

3. Controller

The Controller connects the Model and the View. It processes user input, updates the Model, and adjusts the View. The `HotelController` class implements this logic.

Example Code for `HotelController` class:

```java
public class HotelController {
    private final Hotel hotel;
    private final HotelView view;

    public HotelController(Hotel hotel, HotelView view) {
        this.hotel = hotel;
        this.view = view;
    }

    public void start() {
        int choice;
        do {
            choice = view.showMainMenu();
            switch (choice) {
                case 1:
                    Customer customer = view.collectCustomerData();
                    hotel.bookRoom(customer, 1); // Example: Always booking room type 1.
                    break;
                case 0:
                    System.out.println("Exiting...");
                    break;
            }
        } while (choice != 0);
    }
}
```

Features

- Room Booking: Customers can book rooms based on type and availability.
- Payment Integration: Supports multiple payment methods like Bkash, Nagad, and Card.
- Customer Records: Maintains detailed booking history.

Conclusion

The Hotel Management System is a robust application demonstrating the advantages of the MVC architecture. It is modular, easy to maintain, and scalable for future enhancements.