Implementation Goals and Requirements Analysis

The program aims to implement a hotel management system.

First, the system users can be divided into two categories: Guest and Admin. The system mainly consists of two panels: Guest and Admin. The primary functions for the Guest include: viewing available rooms, booking rooms, checking their own reservation status, and canceling reservations. The main functions for the Admin include: adding rooms, viewing all rooms, deleting rooms, viewing all reservations, and canceling specific reservations. The system architecture diagram is shown in Figure 1.

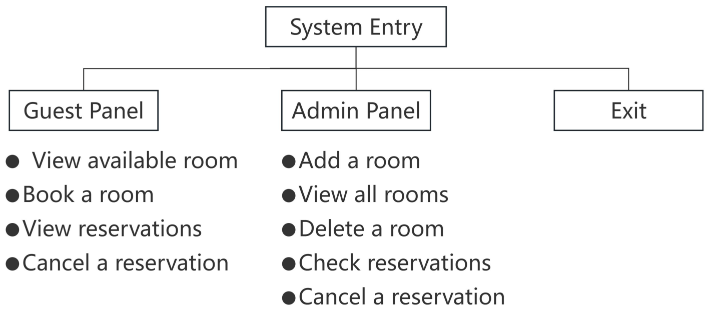


Figure 1 System Architecture Diagram

Additionally, the hotel room types are primarily divided into three categories: Standard, Suite, and Deluxe. For each room type, there are three options: Single, Double, and Suite, with varying prices. The system architecture diagram is shown in Figure 2.

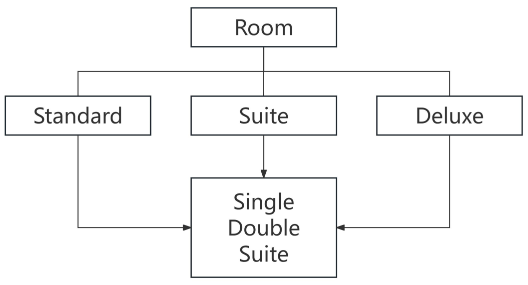


Figure 2 System Architecture Diagram

Code Structure

This program declares four classes to achieve the goals, as shown in Figure 3. The following is an explanation of each class in turn.

```
public class Hotel
public class Room
public class Reservation
public class HotelReservationUI extends Application
```

Figure 3 Class Declaration

1. Hotel

(1) Purpose: As the core of the hotel management system, the Hotel class is responsible for managing room and reservation information. It provides a set of methods to implement various functionalities such as adding, searching, and deleting rooms, as well as making and canceling reservations.

(2) Code Structure

① Attribute Design:

```java
private List<Room> rooms;
private List<Reservation> reservations;
```

Figure 4 Member Variable Declaration

Create instances of the Room and Reservation classes to retrieve hotel room information and reservation details, as shown in Figure 4.

② Method：

```java
public void addRoom(Room room);
public Room findRoom(int roomNumber);
public List<Room> getAllRooms();
public List<Room> getAvailableRooms();
public boolean removeRoom(int roomNumber);
public void addReservation(Reservation reservation);
public void removeReservation(Reservation reservation);
public boolean cancelReservationByRoomNumber(int roomNumber)
public List<Reservation> getAllReservations();
public Reservation findReservationByRoom(int roomNumber);
```

Figure 5 Method Declaration

Method design is shown in Figure 5, primarily focusing on the management of room and reservation information.

For room management, the methods cover adding, searching, retrieving all rooms, retrieving available rooms, and deleting rooms.

For reservation management, the methods cover adding, deleting, canceling reservations, and searching reservations.

2. Room

(1) Purpose: Similar to a data container, it represents a room in the hotel and encapsulates basic room information such as room number, type, price, and availability.

(2) Code structure

① Attribute design:

```java
private int roomNumber;
private String category;
private double price;
private boolean available;
```

Figure 6 Member Variable Declaration

Member variables are defined using the private access modifier to prevent external programs from modifying the key information of the hotel room. The member variables are responsible for storing critical information about the hotel room, such as room number, room type, room price, and availability, as shown in Figure 6.

② Method：

```
public Room(int roomNumber, String category, double basePrice);
private void setPrice(double basePrice);
public int getRoomNumber();
public String getCategory();
public double getPrice();
public boolean isAvailable();
public void setAvailable(boolean available);
```

Figure 7 Method Declaration

The method design is shown in Figure 7. It is mainly responsible for retrieving and modifying the information stored in the member variables, such as changing the room price, querying room type, checking room availability, etc.

3. Reservation

(1) Purpose: A data container used to represent a reservation, encapsulating information such as the number of days booked, total price, and other related details.

(2) Code structure

① Attribute design:

```
private Room room;
private int days;
private double totalCost;
```

Figure 8 Member Variable Declaration

Member variables are defined using the private access modifier to prevent external programs from modifying the key information of the hotel room. The Room class instances are created to retrieve room information. The member variables are responsible for storing reservation information, as shown in Figure 8.

② Method：

```
private void calculateTotalCost();
public Room getRoom();
public int getDays();
public double getTotalCost();
```

Figure 9 Method Declaration

The method design is shown in Figure 9. It is primarily responsible for retrieving and modifying the information stored in the member variables, such as calculating the price, retrieving the number of days, etc.

4. HotelReservationUI

(1) Purpose: The user interface layer of the hotel reservation system, responsible for interacting with users, including displaying menus, handling user inputs, and displaying information. It manages hotel rooms and reservation information by calling methods from the Hotel class.

(2) Code structure

① Attribute design:

```
private Hotel hotel;
private Stage primaryStage;
private Reservation currentReservation;
```

Figure 10 Member Variable Declaration

As shown in Figure 10, all member variables are defined using the private access modifier to prevent external programs from modifying the key information of the hotel rooms. The Hotel class instance is created to retrieve room information and reservation details. An instance of the Stage class is created using JavaFX application design to represent the main window of the interactive interface. An instance of the Reservation class is created to retrieve reservation information.

② Method：

```
//Initialize
public void start(Stage primaryStage);
private void initializeRooms();
private VBox createMainMenu();
private void openGuestPanel();
private void openAdminPanel();
//Guest Functions
private void showAvailableRooms();
private void bookRoom();
private void viewReservations();
private void cancelReservation();
//Admin Functions
private void addRoom();
private void showAllRooms();
private void deleteRoom();
private void viewAllReservations();
private void cancelAnyReservation();
private void showAlert(String title, String content);
```

Figure 11 Method Declaration

As shown in Figure 11, multiple methods are defined to build the user interaction interface, which are mainly divided into initialization interface, guest interface, and admin interface methods.

The initialization interface method initializes the window by calling information from the Hotel instance, using a VBox layout manager to set the colors, fonts, etc., for the interaction panel, and links the main window to the guest or admin interface. When the user clicks on different interfaces, the program automatically performs the corresponding actions.

The guest interface method mainly handles the functions of viewing, booking, and managing room reservations. It retrieves information by calling instances of Hotel, Room, and Reservation, and creates pop-up windows to display the information to the user.

The admin interface method deals with functions such as adding, viewing, and deleting rooms, as well as viewing and canceling reservations. It creates Room instances and adds them to the room list, and also supports modifying or querying room information.

5. Summary

In conclusion, my program follows the MVC (Model-View-Controller) architecture pattern, separating the data model, user interface, and business logic to ensure the program is simple and easy to modify, as shown in Figure 12.

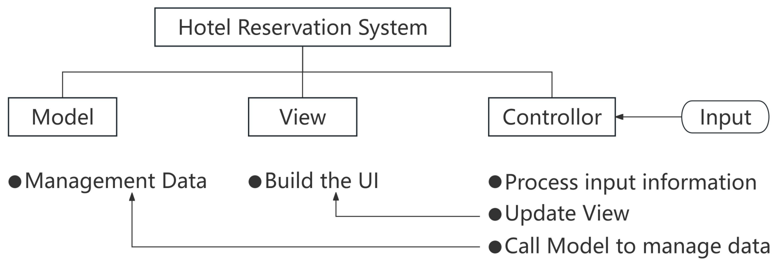


Figure 12 MVC Architecture Diagram

(1) Model:

① Responsible for managing hotel room information and reservation data.

② Includes the Hotel, Room, and Reservation classes.

(2) View:

① Responsible for building the user interaction and graphical interface using JavaFX.

② Includes the construction of the main menu, Guest Panel, and Admin Panel.

(3) Controller:

① Responsible for calling the Model to process input data and update the View layer.

② This program is managed by the HotelReservationUI class.

Algorithm Implementation

• Room Class

1. Room Initialization (Implemented via Constructor)

The room's basic information, such as room number, type, price is initialized using the Java constructor. The implementation flow is shown in Figure 13.

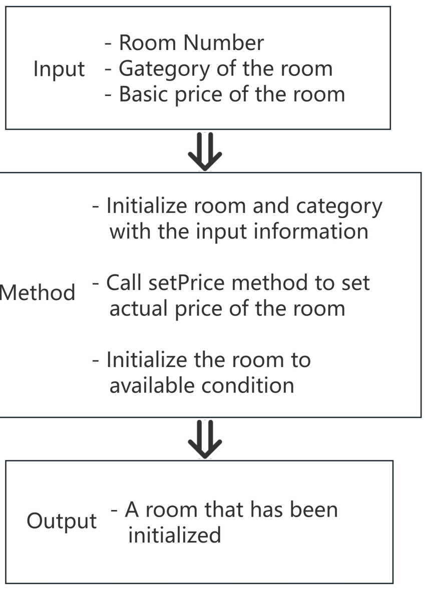

Input
- Room Number
- Gategory of the room
- Basic price of the room
Method
- Initialize room and category with the input information
- Call setPrice method to set actual price of the room
- Initialize the room to available condition
Output
- A room that has been initialized

Figure 13 Implementation Flowchart

2. Set Room Price

In this system, hotel rooms are divided into three categories: Standard, Suite, and Deluxe. For each room category, there are three options: Single, Double, and Suite. The price calculation is based on the three options of the Standard room. The prices for the three types of Standard rooms are shown in Table 1.

Single	500 RMB /night
Double	800 RMB /night
Suite	1200 RMB /night

Table 1 Standard Room Price

The price correspondence is as follows:

$$\mathrm{SuiteRoom} = 1.5 \times \mathrm{Standard}Room$$
$$DeluxRoom = 3 \times \mathrm{Standard}Room$$

In conclusion, the algorithm for setting room prices is shown in the implementation flow in Figure 14.

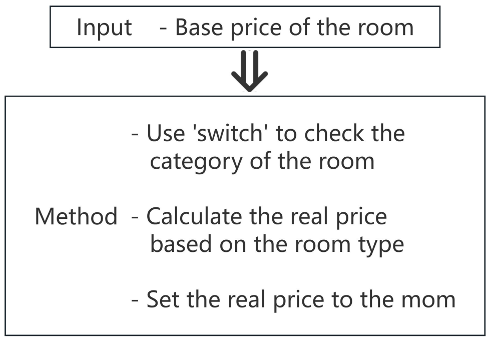
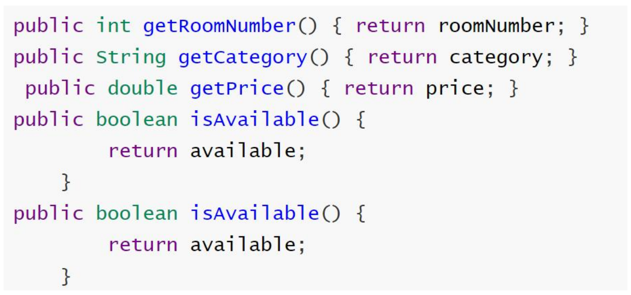

Input
- Base price of the room
Method
- Use 'switch' to check the category of the room
- Calculate the real price based on the room type
- Set the real price to the mom

Figure 14 Implementation Flowchart

3. Get Room Information



```
public int getRoomNumber() { return roomNumber; }
public String getCategory() { return category; }
 public double getPrice() { return price; }
public boolean isAvailable() {
        return available;
    }
public boolean isAvailable() {
        return available;
    }
```

Figure 15 Method Declaration

The method is shown in Figure 15. By accessing the member variables of the Room class, the corresponding values can be returned.

• Reservation Class

1. Reservation Information Initialization (Implemented via Constructor)

The room reservation information, including the reserved room details and price calculation, is initialized using the Java constructor. The implementation flow is shown in Figure 16.

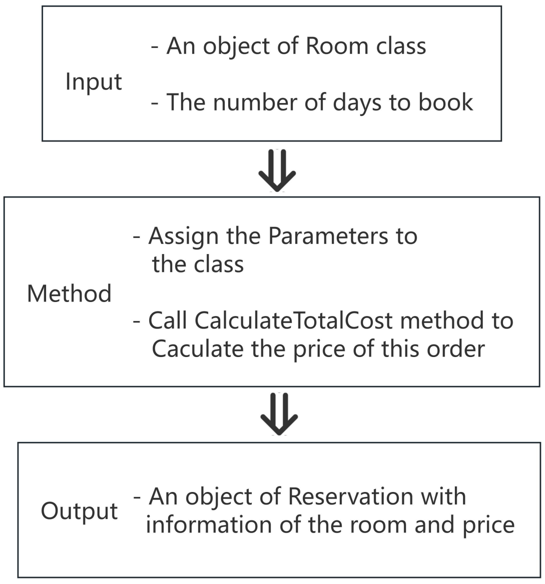

Input
- An object of Room class
- The number of days to book
Method
- Assign the Parameters to the class
- Call CalculateTotalCost method to Caculate the price of this order
Output
- An object of Reservation with information of the room and price

Figure 16 Implementation Flowchart

2. Total Cost Calculation Algorithm

By using the member variables to get the number of reserved days and basic room information, the base reservation cost is calculated using the following formula:

$$\text{total}Cost = day \times price$$

Then, further checks are made to apply a 20% discount if the number of reservation days exceeds 7, resulting in the final total cost. The discount calculation is as follows:

$$\text{total}Cost = \text{total}Cost \times (1 - 20\%)$$

The implementation flow is shown in Figure 17.

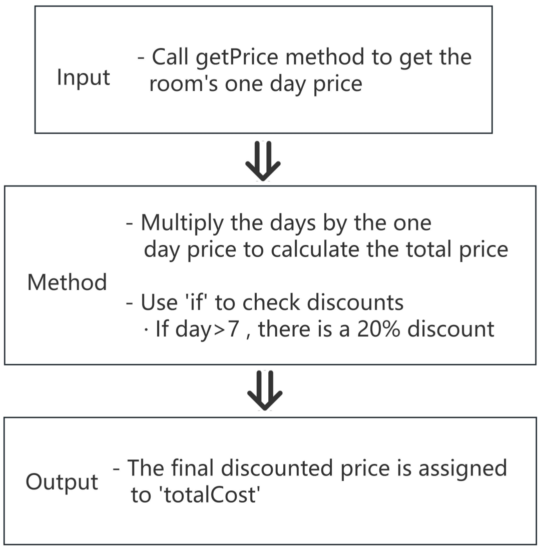

Input
- Call getPrice method to get the room's one day price
Method
- Multiply the days by the one day price to calculate the total price
- Use 'if' to check discounts
· If day>7 , there is a 20% discount
Output
- The final discounted price is assigned to 'totalCost'

Figure 17 Implementation Flowchart

3. Reservation Information Retrieval Algorithm

```
public Room getRoom() { return room; }
public int getDays() { return days; }
public double getTotalCost() { return totalCost; }
```

Figure 18 Method Declaration

The method is shown in Figure 18. By accessing the member variables of the Reservation class, the corresponding values can be returned.

• Hotel Class

1. Reservation Information Initialization (Implemented via Constructor)

Use a Java constructor to initialize the lists for storing room information and reservation information as empty. The implementation flow is as shown in Figure 19.

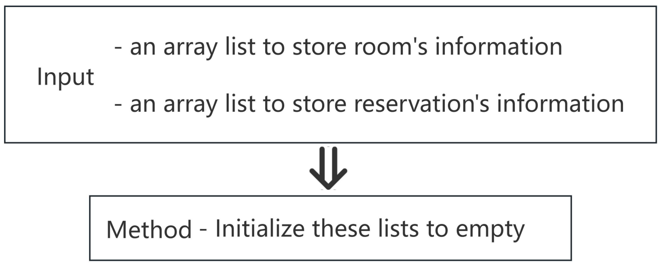

Input
- an array list to store room's information
- an array list to store reservation's information
Method - Initialize these lists to empty

Figure 19 Implementation Flowchart

2. Room Management Algorithm

(1) Add Room

By using list operations, a Room object is added to the room list. The implementation flow is shown in Figure 20.

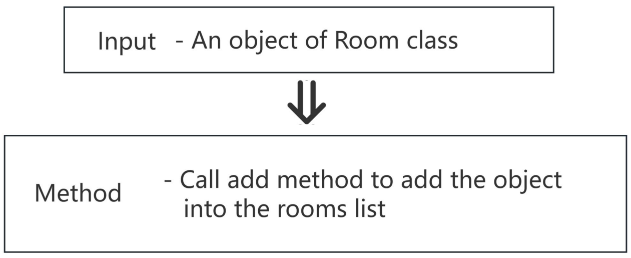

Input - An object of Room class
Method - Call add method to add the object into the rooms list

Figure 20 Implementation Flowchart

(2) Delete Room

By traversing the Room list using the room number, when the corresponding room is found, list operations are used to delete the room and return true. If the room is not found or the deletion is unsuccessful, false is returned. The implementation flow is shown in Figure 21.

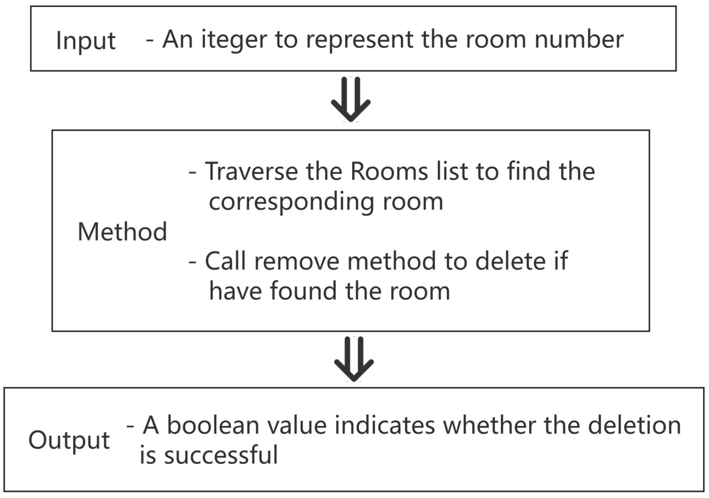

Input - An integer to represent the room number
Method - Traverse the Rooms list to find the corresponding room
- Call remove method to delete if have found the room
Output - A boolean value indicates whether the deletion is successful

Figure 21 Implementation Flowchart

(3) Retrieve Room Information

To retrieve information for a specific room, view all rooms, or get available rooms, simply traverse the Rooms list and return the corresponding information. The method is shown in Figure 22.

```
public Room findRoom(int roomNumber);
public List<Room> getAllRooms();
public List<Room> getAvailableRooms();
```

Figure 22 Method Declaration

3. Reservation Management Algorithm

(1) Add Reservation

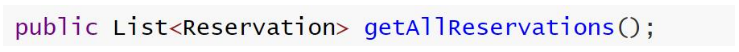
Similar to the "Add Room" algorithm in the Room Management method, use list operations to add a Reservation object to the Reservation list.

(2) Delete Reservation

Similar to the "Delete Room" algorithm in the Room Management method, use list operations to traverse the Reservation list, find the corresponding reservation, and remove the Reservation object from the list. Additionally, return a boolean value indicating the success of the deletion.

(3) Retrieve Reservation Information

The method for viewing reservation information is shown in Figure 23. Traverse the Reservation list and return the corresponding information.



```
public List<Reservation> getAllReservations();
```

Figure 23 Method Declaration

Inter-Class Interaction

The Hotel class, as the core of the hotel management system, is responsible for managing and operating the encapsulated information of Room and Reservation. It implements functionalities such as adding, searching, and deleting hotel rooms and reservation information. The HotelReservationUI class is responsible for interacting with the user and building the user interface. It provides user input to the Hotel class and calls its methods for data processing, updating the interaction page. The class relationships are shown in Figure 24.

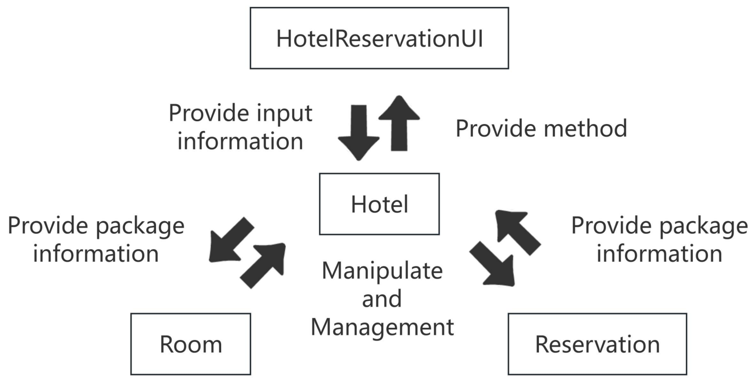

HotelReservationUI
Provide input information
Provide method
Hotel
Provide package information
Provide package information
Manipulate and Management
Room
Reservation

Figure 24 Class Relationship Diagram