Table of Contents

Table of figures

Table of tables

1.0 Analysis and Design of Object- Oriented System Structure

Based on the case study of "Giant Forest Inn Hotel by Robert Stumpf", the problem specification of the case study is analyzed to gather crucial requirements for the improvement of the system. From the requirements gathered, a new custom hotel reservation and guest management system is designed.

The requirements gathered from the case study of "Giant Forest Inn Hotel by Robert Stumpf" are:

1) Guest can make a reservation on the web interface or through walk-in.
2) An email confirmation will be sent to the guest after the reservation is done via the web interface to inform the reservation status.
3) Walk in reservations are only allowed if the requested rooms are available.
4) Guests are required to sign a check in form and pay a deposit to complete the check in process.
5) Guest can note if they want a room that has a specific celebrity name.
6) The hotel provides 130 luxury rooms:
    a) 1 Queen bed - 50 rooms
    b) 2 Queen beds - 50 rooms
    c) Two Room - 15 rooms
    d) Three Room - 5 rooms
    e) Bridal - 5 rooms
7) The hotel provides 25 cottage rooms:
    a) Two Room - 10 rooms
    b) Three Room - 12 rooms
    c) Four Room - 3 rooms
8) The hotel provides luxury room and cottage room with patio and forest view.
9) The hotel provides 100 hotel rooms and 25 secluded cottages with garage
10) The hotel is able to provide amenities such as tennis court, sauna baths, stables with covered horseback trail, boat rentals, one ball room, five dining room, indoor and outdoor swimming pool.

11) The hotel rooms charges are as follows:

a) If guests stay for less than two months, renting price will be charged based on the selected room per night.
b) If guest stay for more than two months, a leasing price will be charged based on the selected room per month.

12) All amenities, services and food orders enjoyed by guests are accumulated and charged based on the guest's name to be included in the final bill that will be presented a night before checking out.
13) Should there be any problem on the bill calculation, guests can request the bookkeeping department to recheck the bill and make necessary adjustments.
14) Payments can be done by cash, cheque or credit card.
15) The hotel must be able to track the condition of the room based on the following statuses:
    a) Ready
    b) Occupied
    c) Under repair
    d) Not ready
16) The hotel must be able to track the housekeeping status of the rooms.

From the requirements gathered above, the following diagrams are designed for the improved system.

1.1 State Diagram

Figure 1 to Figure 10 displays the process of the improved system using state diagram.

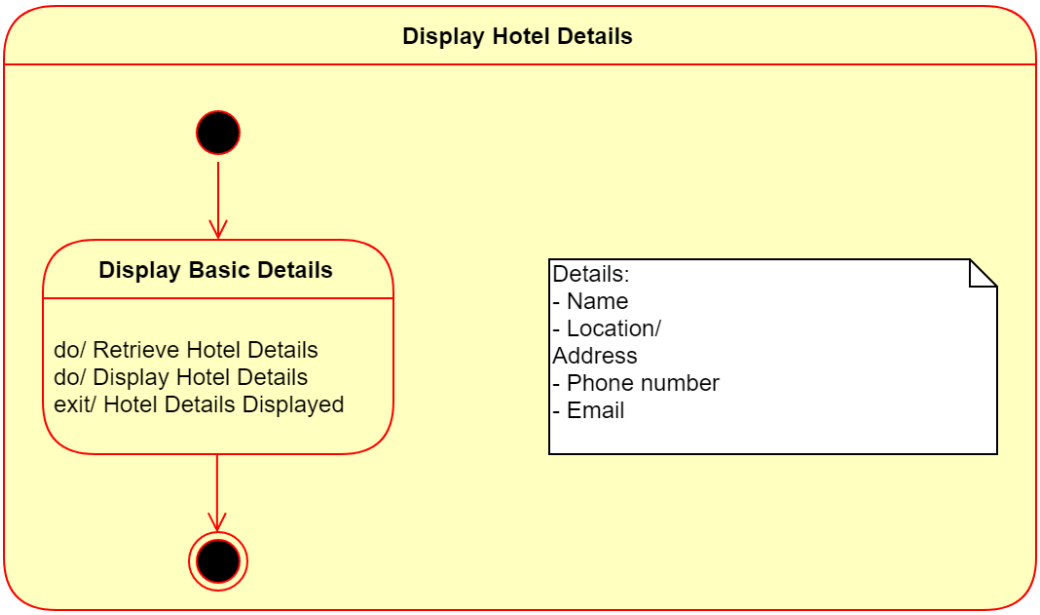

Figure 1: State Diagram – Display Hotel Details

Figure 1 describes the process of displaying the hotels basic details that includes the hotel name, location, phone number and email.

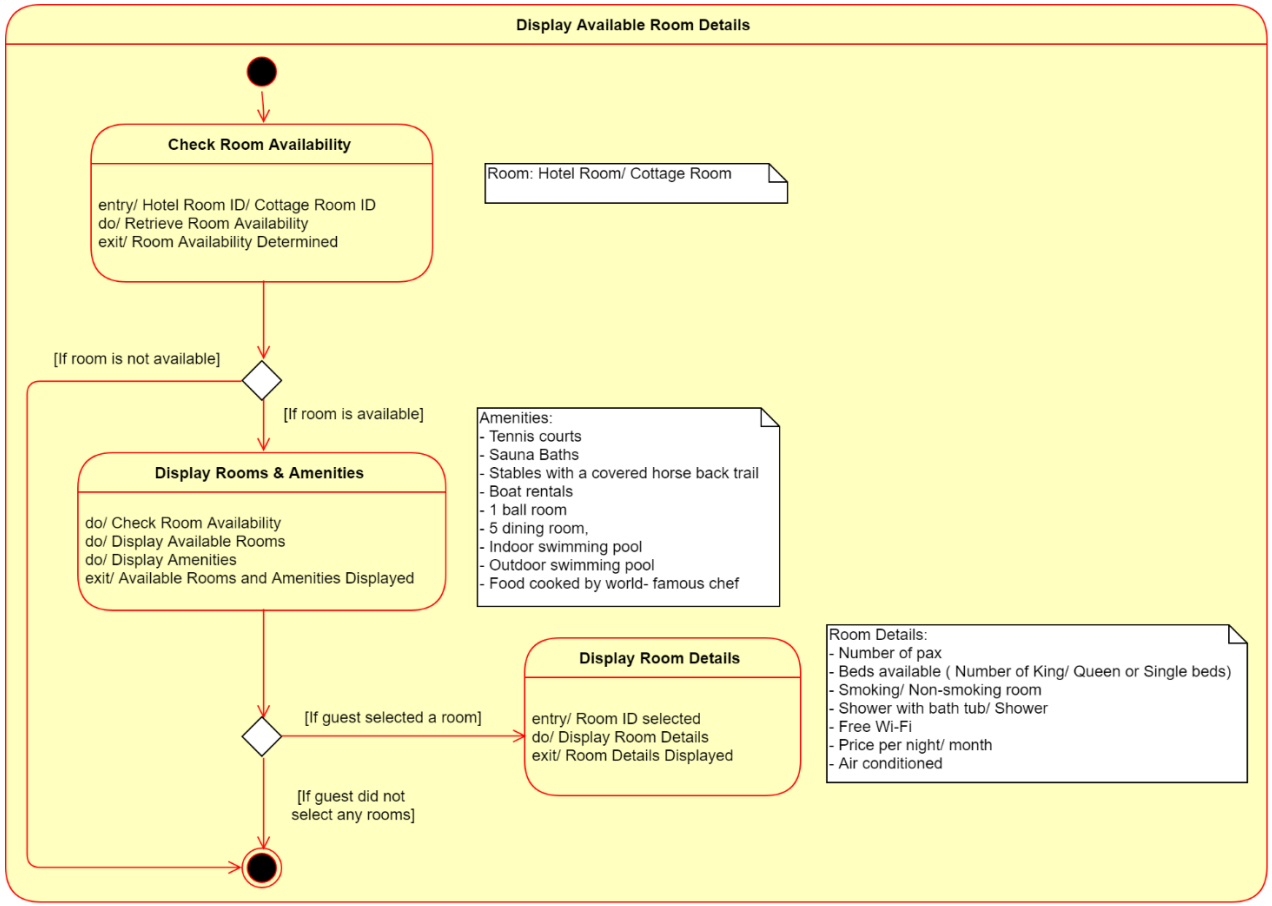

Figure 2: State Diagram – Display Availability Details

Figure 2 shows the flow of checking the available rooms on the system to be displayed to the guests. The system will display the available rooms along with the amenities that can be used by the guests of the room and also the details of the room.

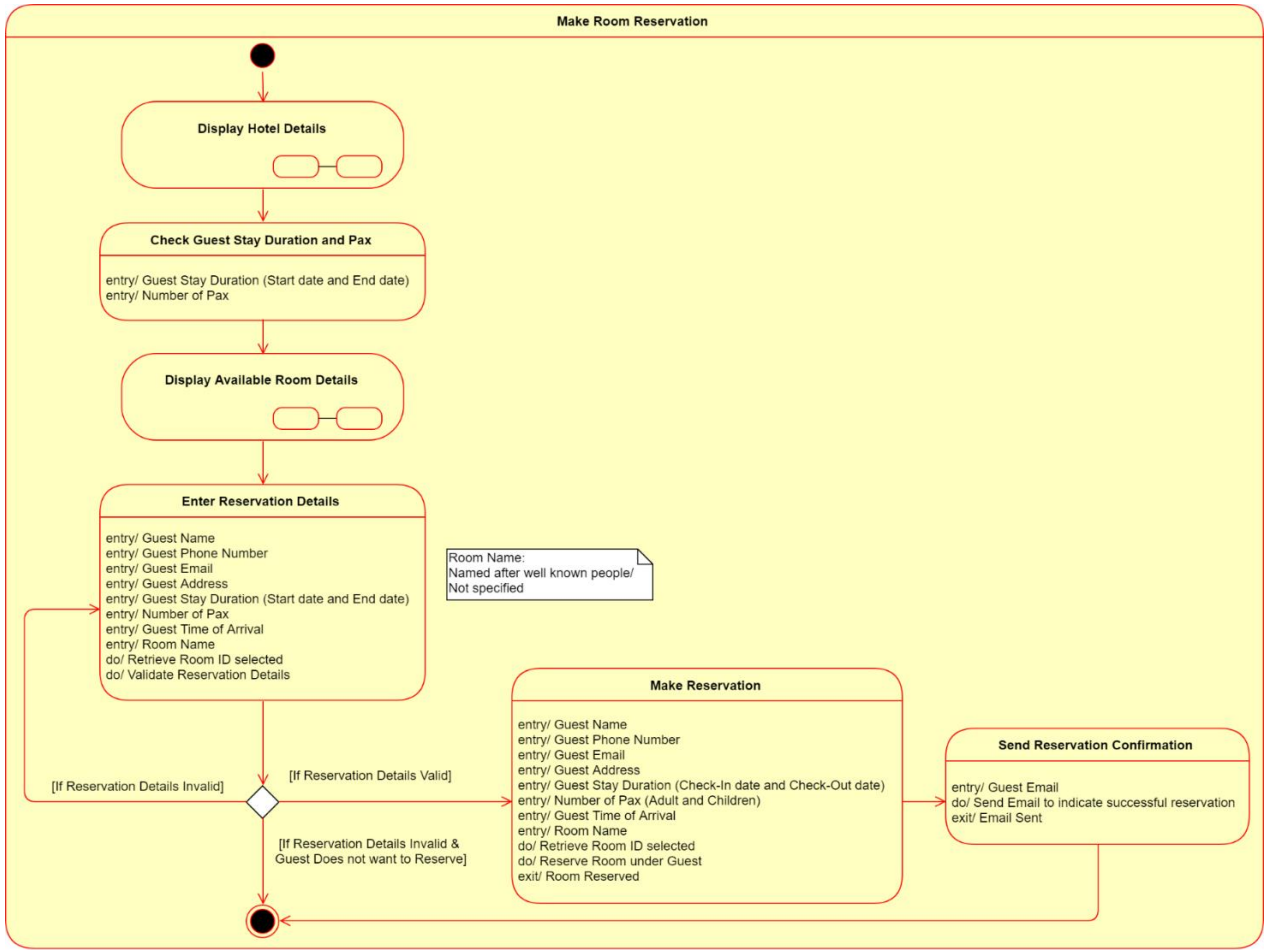

Figure 3: State Diagram – Make Room Reservation

Figure 3 shows the flow of reservation done via the web interface of the system. Guests will choose the room that they would like to stay from the interface and make the reservation by providing details such as the number of guests staying, the check in date, check out date and so on. Once the reservation is made, an email will be sent to the guest as confirmation in the reservation.

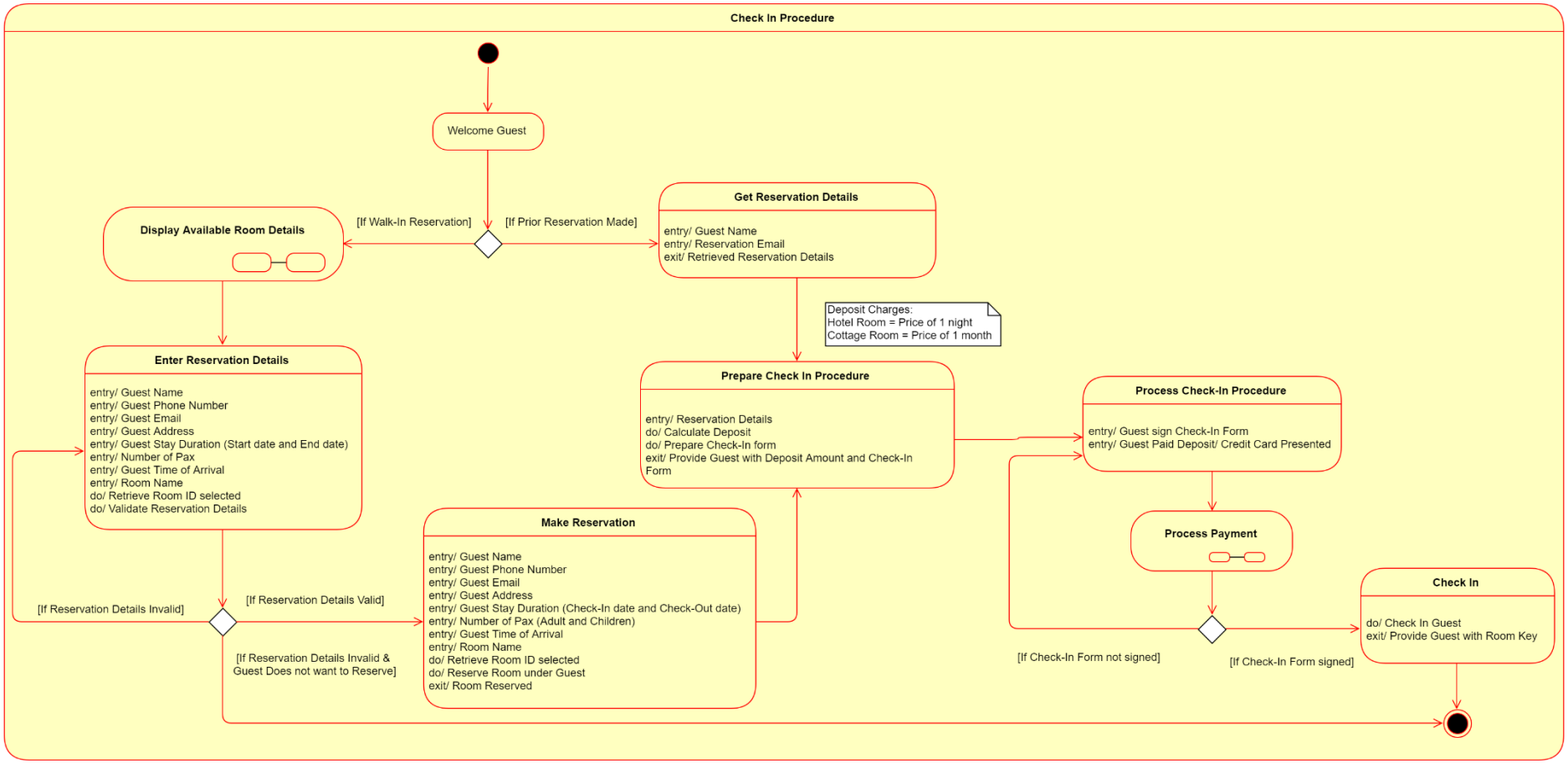

Figure 4: State Diagram – Check In Procedure

Figure 4 shows the check in procedure in the hotel. Guests are required to pay a deposit and sign the check in form to complete the check in process. For walk-in guests, the availability of desired room is checked before placing a quick reservation.

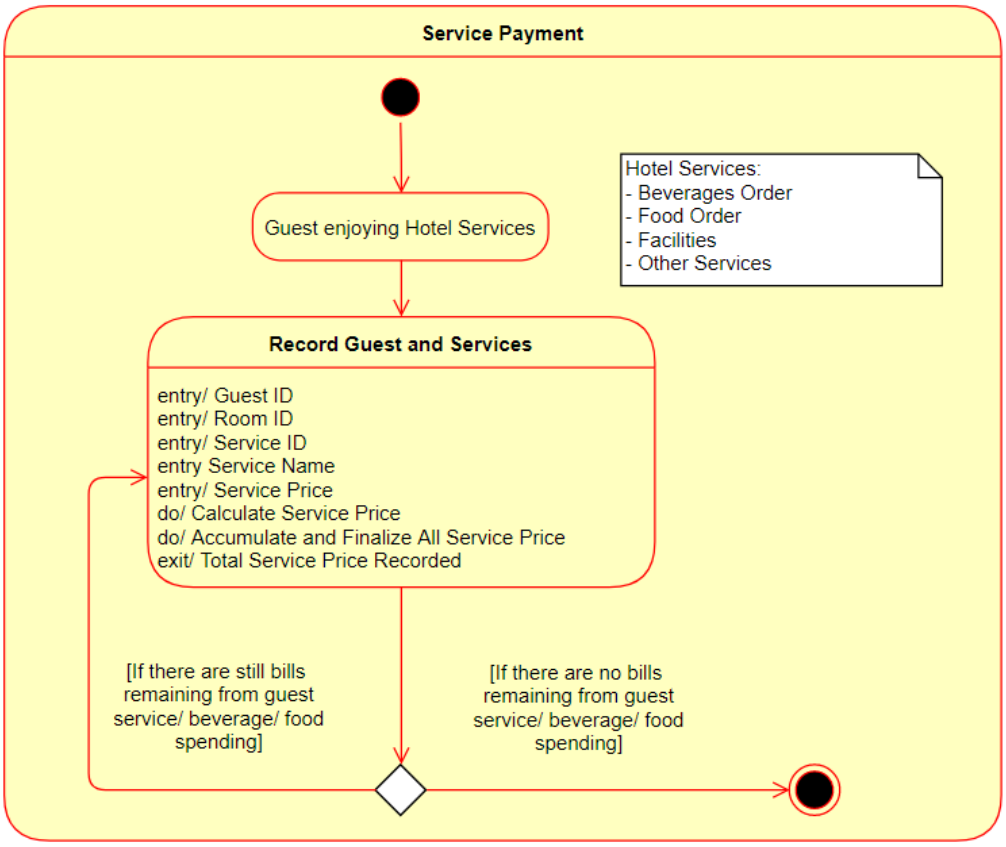

Figure 5: State Diagram – Service Payment

Figure 5 shows the flow of guests using the services provided by the hotel. Guests can use the hotel amenities, food services and other services available and the bill that is incurred will be included in the total payment at the end of the guest's stay.

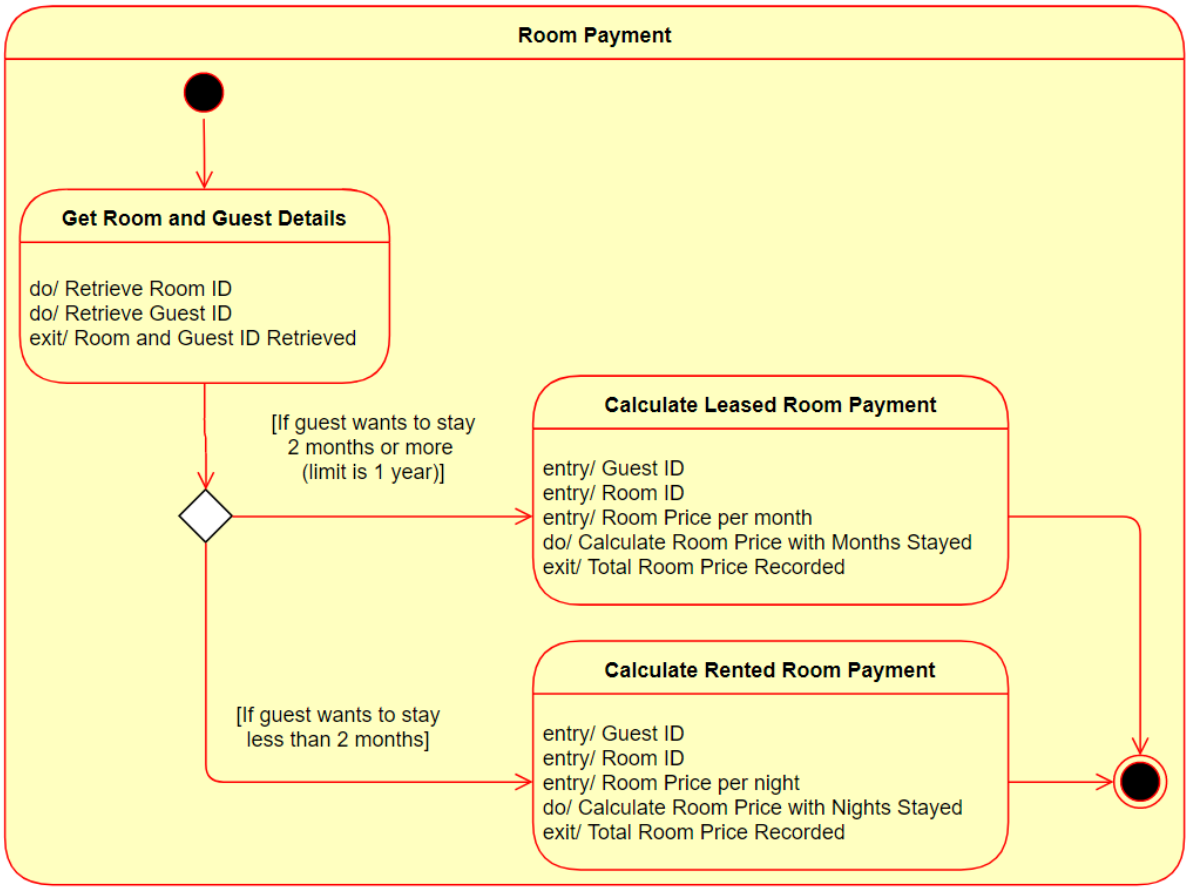

Figure 6: State Diagram – Room Payment

Figure 6 shows the calculation for room payments. If guests stay more than two months, then the room payment will be calculated by month. If guests wish to stay less than two months, then the room payment will be calculated by night.

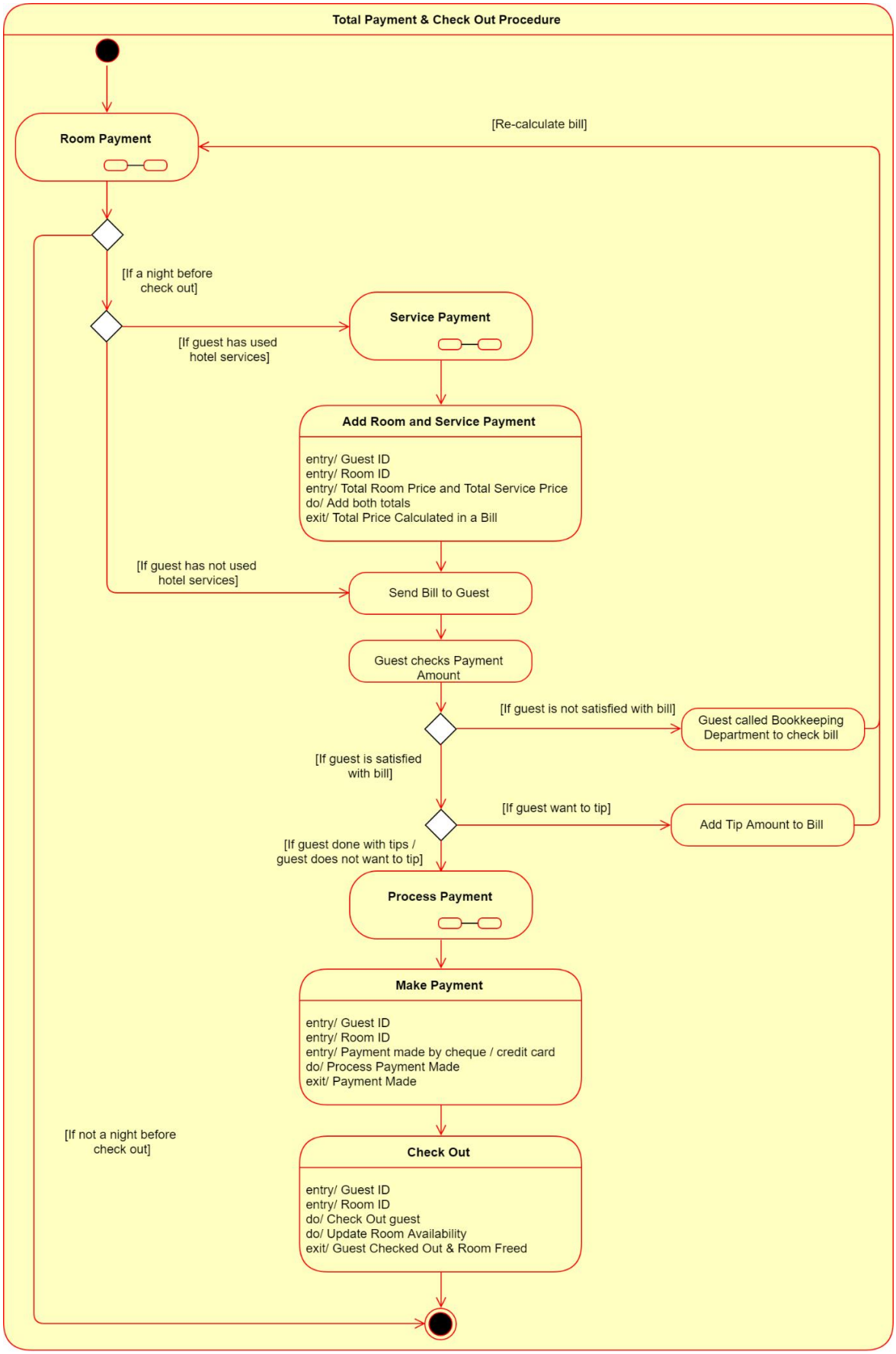

Figure 7: State Diagram – Total Payment & Check Out Procedure

Figure 7 shows the bill payment done by guests after accumulating the services and the room payment that has incurred throughout the guest's stay and the payment validation process. Once the payment is done, the guests can then proceed to check out and the room occupancy will be available.

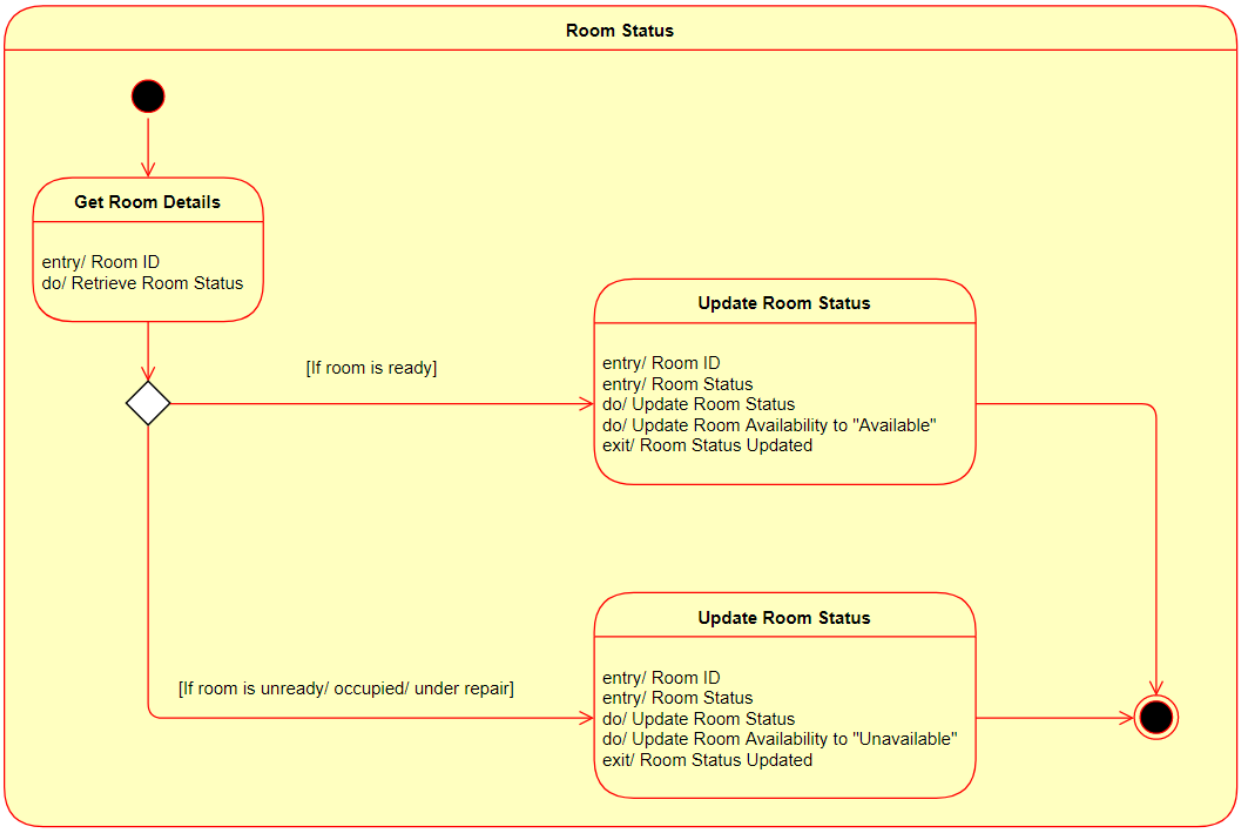

Figure 8: State Diagram – Room Status

Figure 8 shows the room status of the hotel rooms and cottage rooms to indicate their current condition so that their availability can be determined.

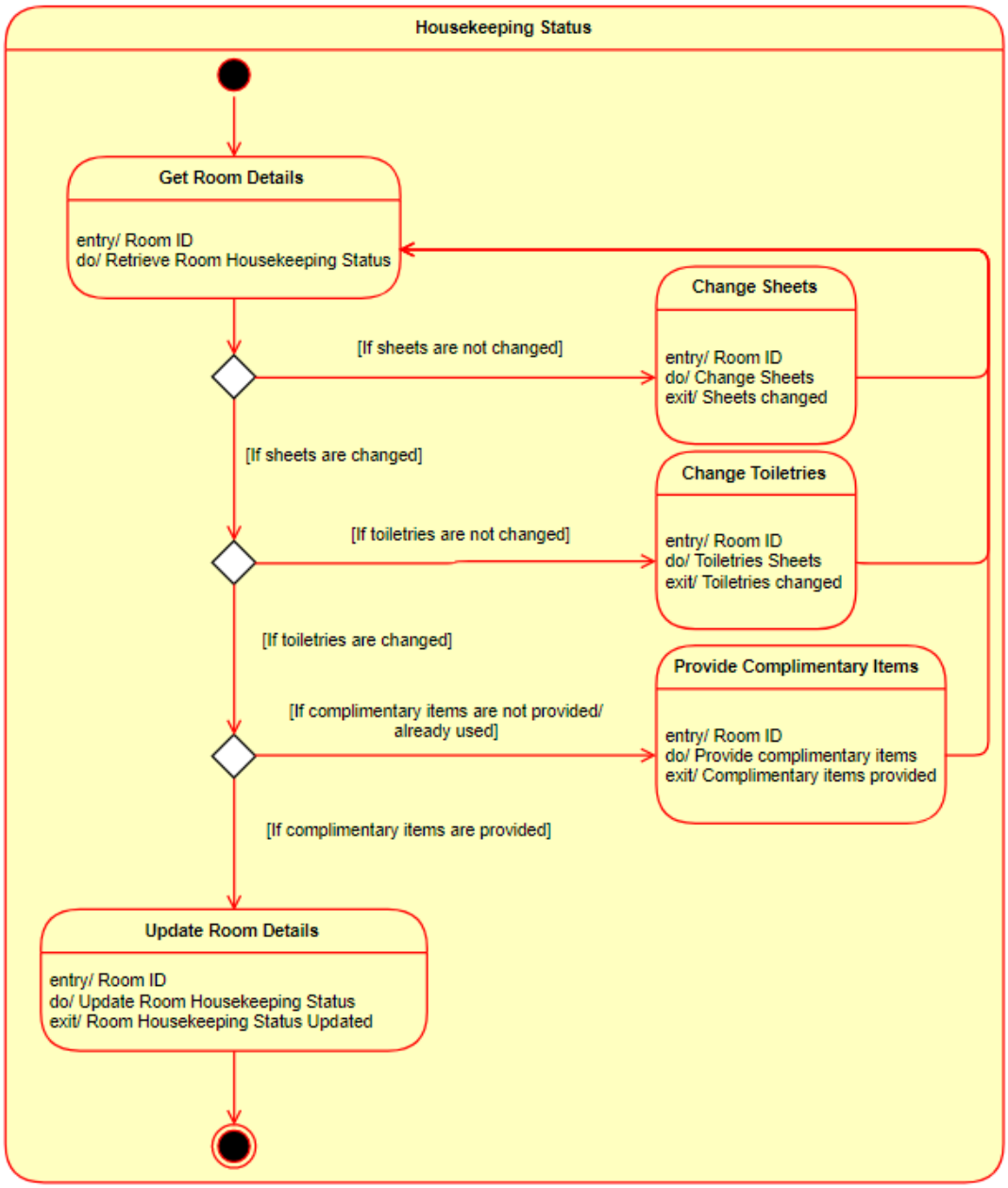

Figure 9: State Diagram – Housekeeping Status

Figure 9 shows the housekeeping status of the hotel rooms and cottage rooms to keep track of the housekeeping tasks that has been done for each room. These tasks can include changing the bed sheets, changing the blanket sheets, refilling toiletries, refilling complimentary items and so on.

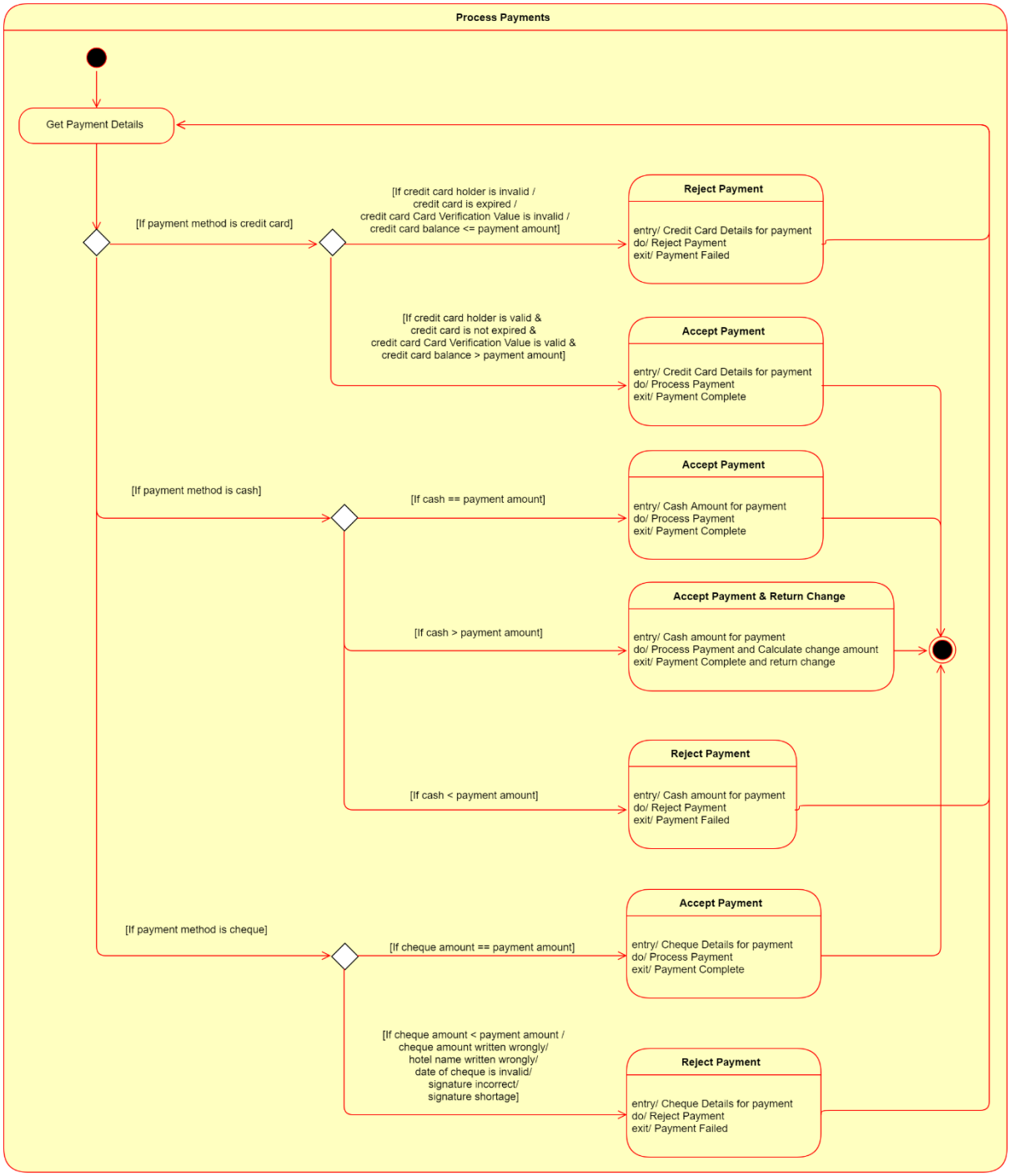

Figure 10: State Diagram – Process Payments

Figure 10 shows validation steps for the different payment methods that can be done in the system. Guests can pay with cash, cheque or credit card. For cheque, the cheque holder name, bank name and signature are checked. For the credit card, the card holder name, bank name, card balance and card expiry are considered.

1.2 Use Case Diagram

The use case diagram created in Figure 11 and Figure 12 based on the system requirements will explain the relationship of each actor with the system in this section.

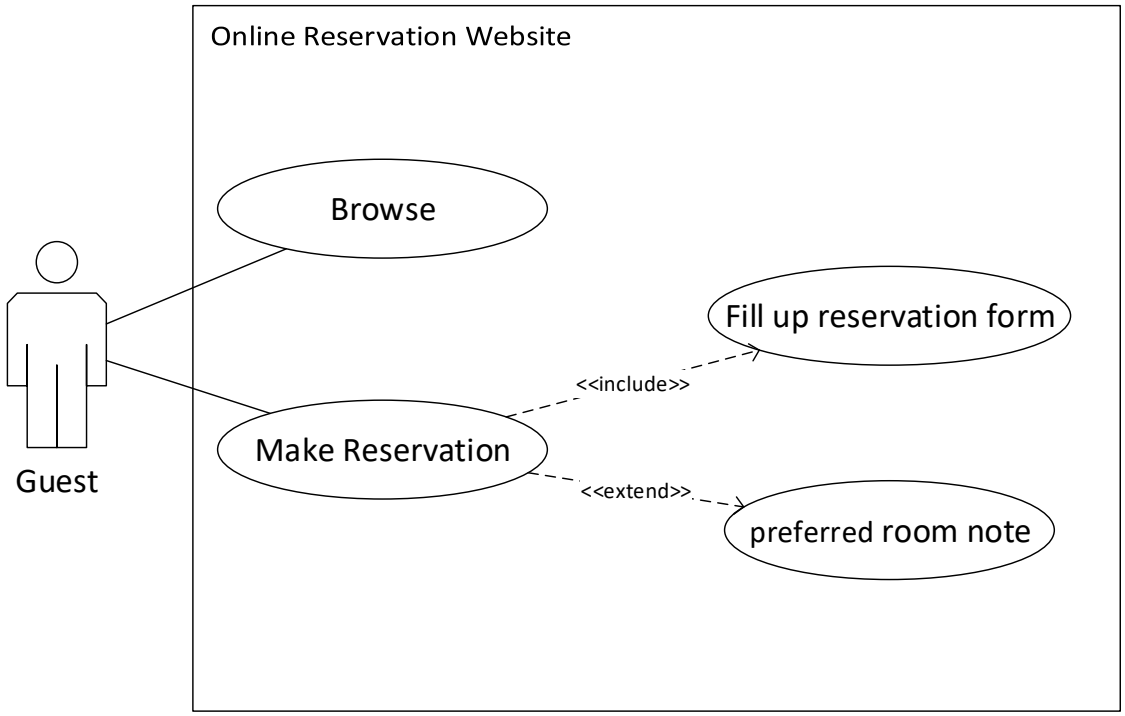

Figure 11: Show the use case of the guest on the online website of the system

Based on Figure 11 above, guest can visit the online website to browse for information regarding Giant Forest Inn and make a reservation. The guest will have to first fill up a reservation form by inputting their basic information such as name, contact information and address. Hence, input the room, room view and also an optional note. The optional note is usually for special request such as the request for a specific celebrity named room.

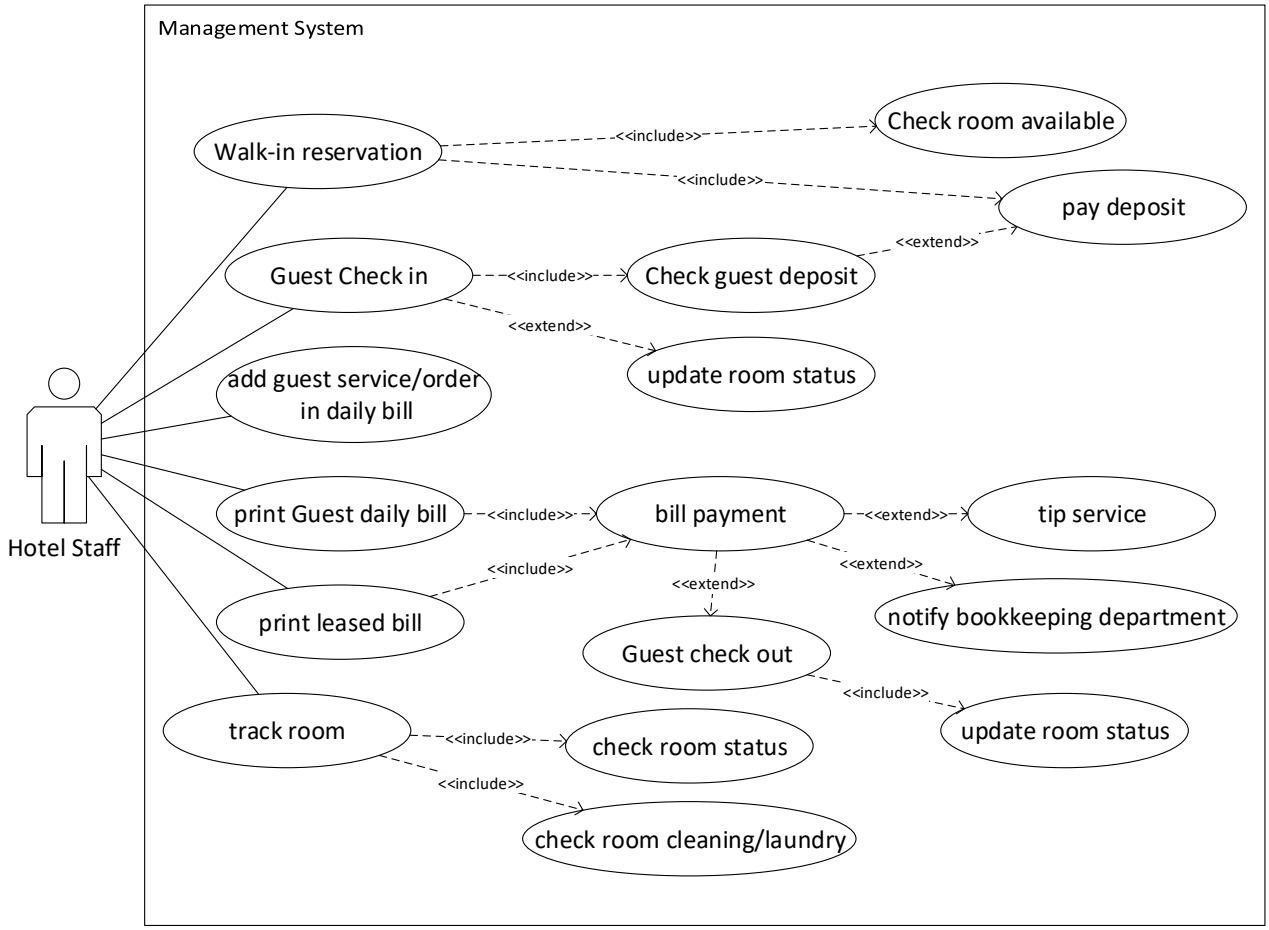

Figure 12: Show the use case of the actors in the hotel management system

Based on Figure 12 above, the use case of the hotel staff in the hotel management system such as making a walk-in reservation, doing a guest check in, adding guest service/order in daily bill, printing guest daily bill and leased bill, track each room regarding its status and cleaning, and etc.

1.3 Class Diagram

The class diagram created for the system will be used to show the relationship between the classes. It will also show the relationship between the attributes and the operations present in each class. The class diagrams will be displayed in Appendix A: Class Diagrams.

2.0 System Development and Testing

2.1 Front-End Development

The front-end development of the improved system is produced using Vue.js as front-end and Node.js Express as middleware. The middleware is used to allow communication between the Vue.js and the MySQL database. The front-end is the web interface made for guests to browse and make reservation. The pages of the website are as shown:

1) Home page of Giant Forest Inn:

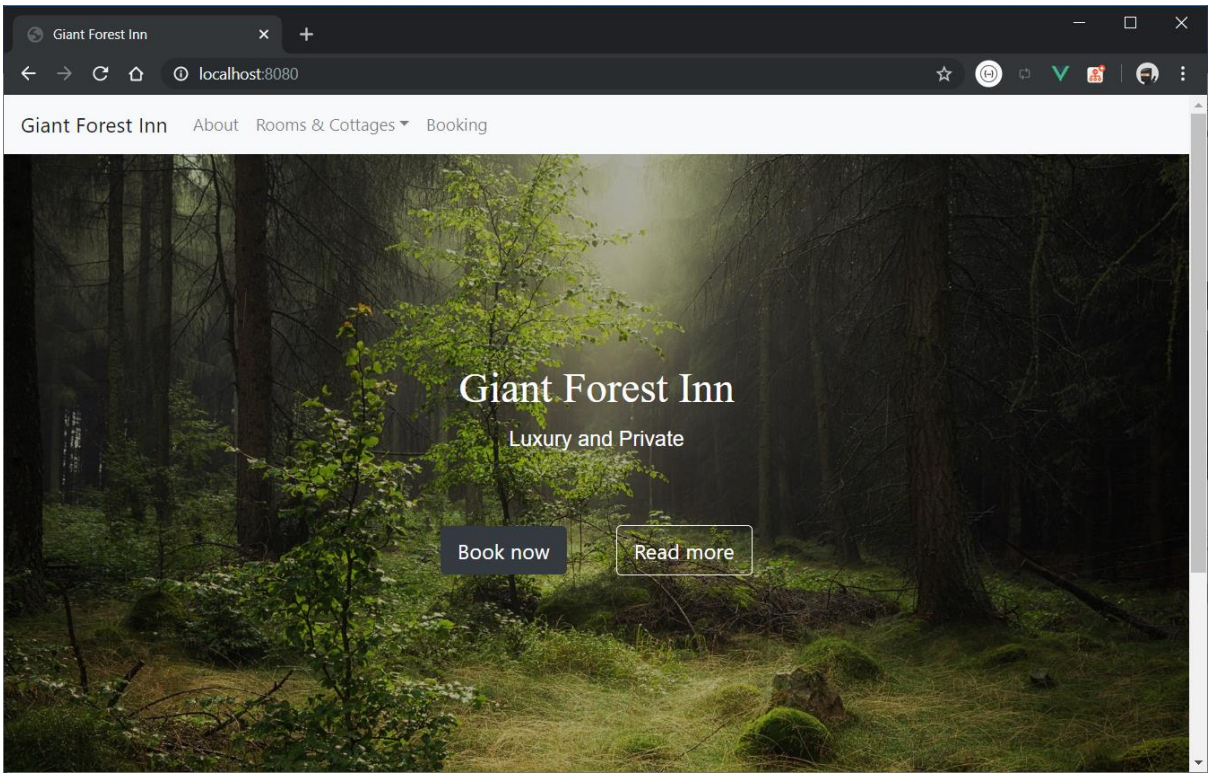

Figure 13: The home page of the website

2) About page:

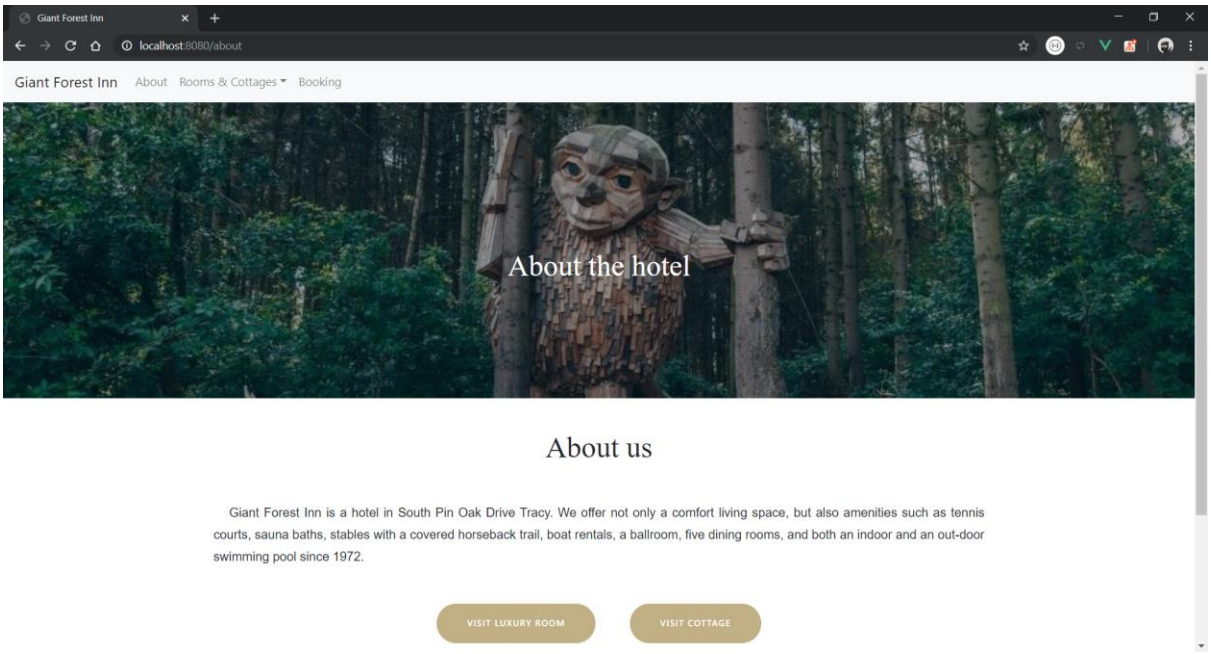

Figure 14: The about us page of the website

3) Room page:

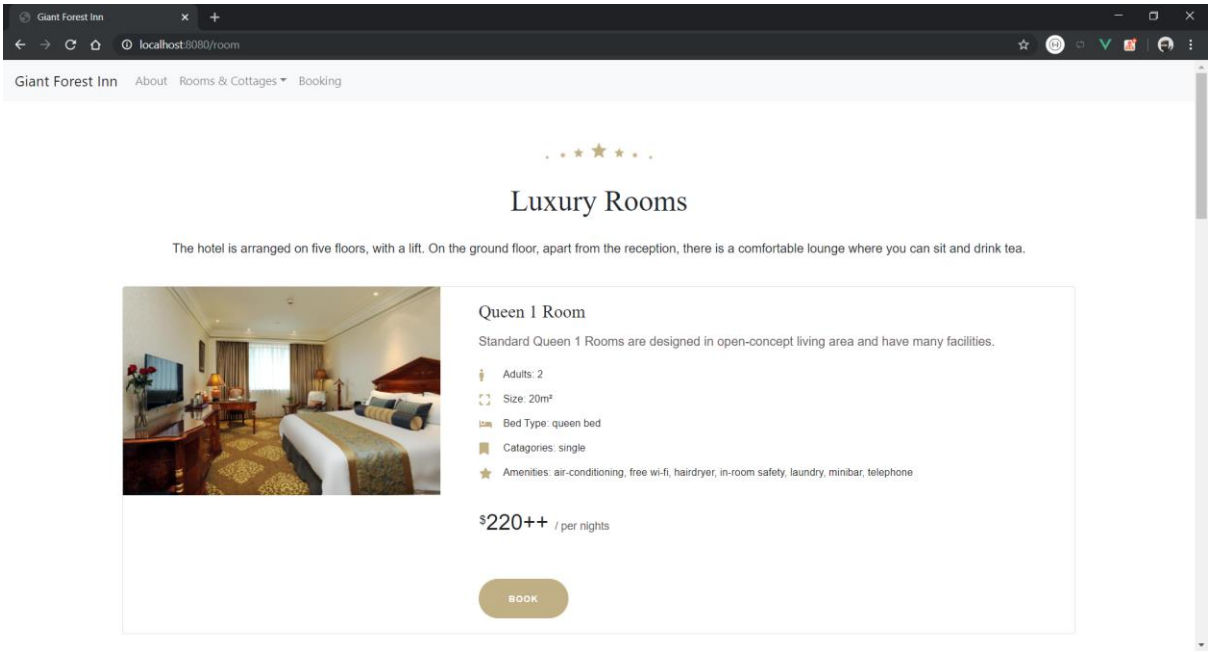

Figure 15: The room page of the website

4) Cottage page:

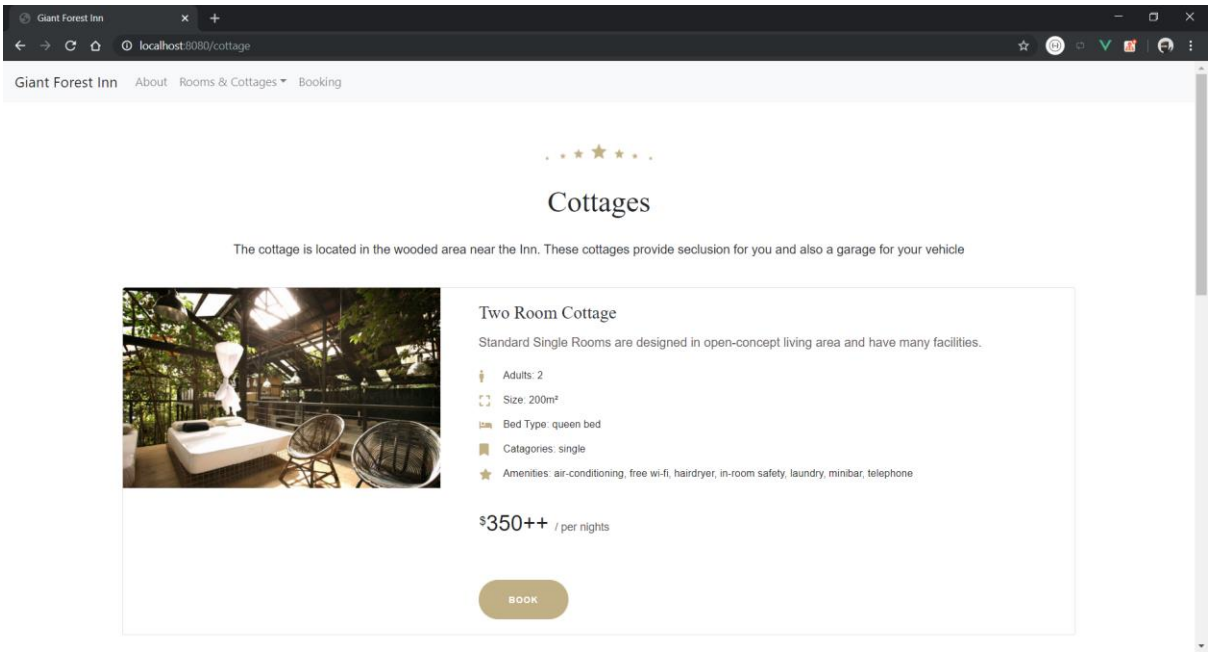

Figure 16: The cottage page of the website

5) Booking page:

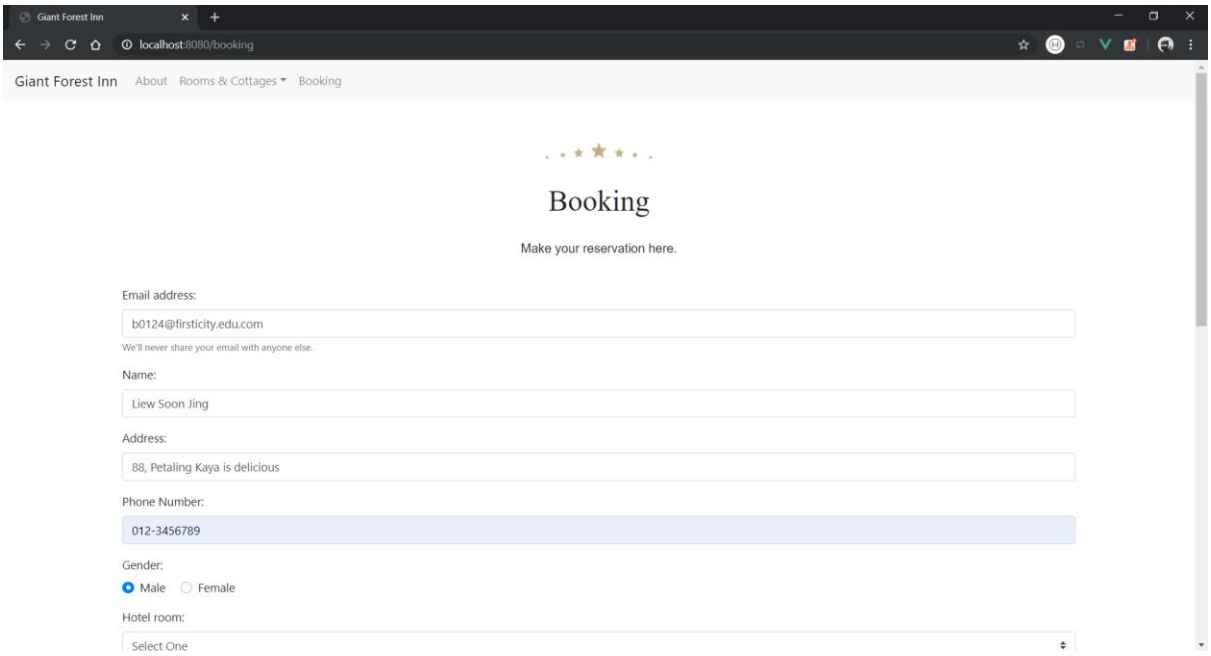

Figure 17: The booking (reservation) page of the website

The test plan of the system is shown in Appendix B: Test Plan in the section B.1 Front-End Testing for the testing and evaluations done on the performance of the system.

2.2 Back-End Development

The back-end development of the improved system is produced using Java. The connection from the front-end to the back-end system is linked with a shared database connection to sync the data in both systems. The system has a total of 21 tables as shown in Figure 18 that keeps track of all the transactions in the system.

Figure 18: Database of the System

After the database has been created, the system is then developed with Java programming language. The interface of the system that has been created as shown from Figure 19 to Figure 28.

1) Reservation Display:

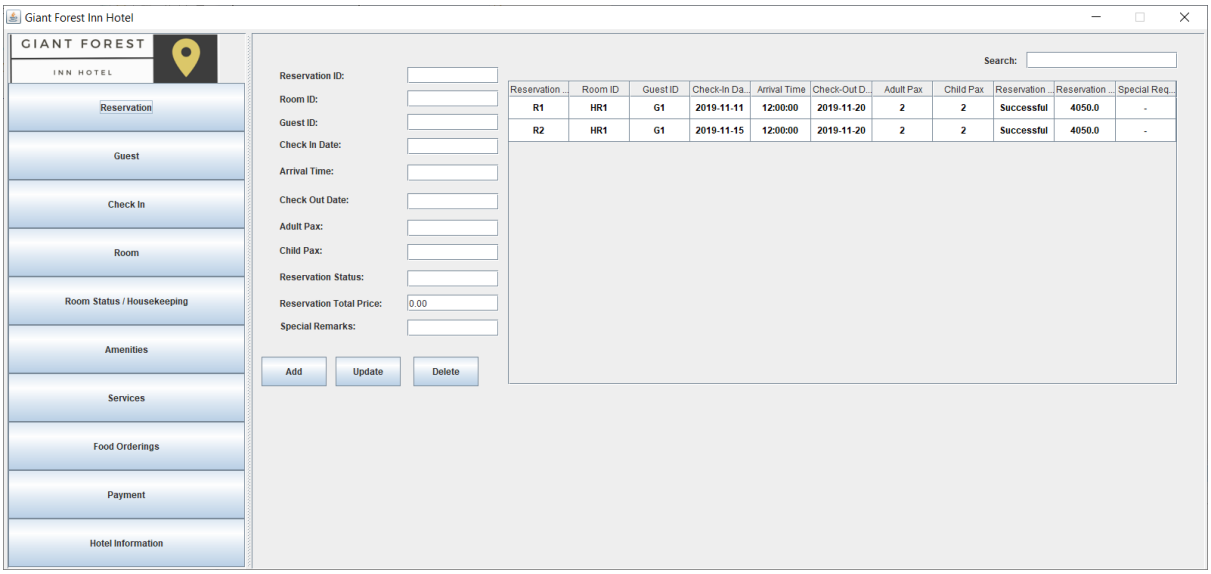

Figure 19: Reservation Interface for Backend System

2) Guest Display

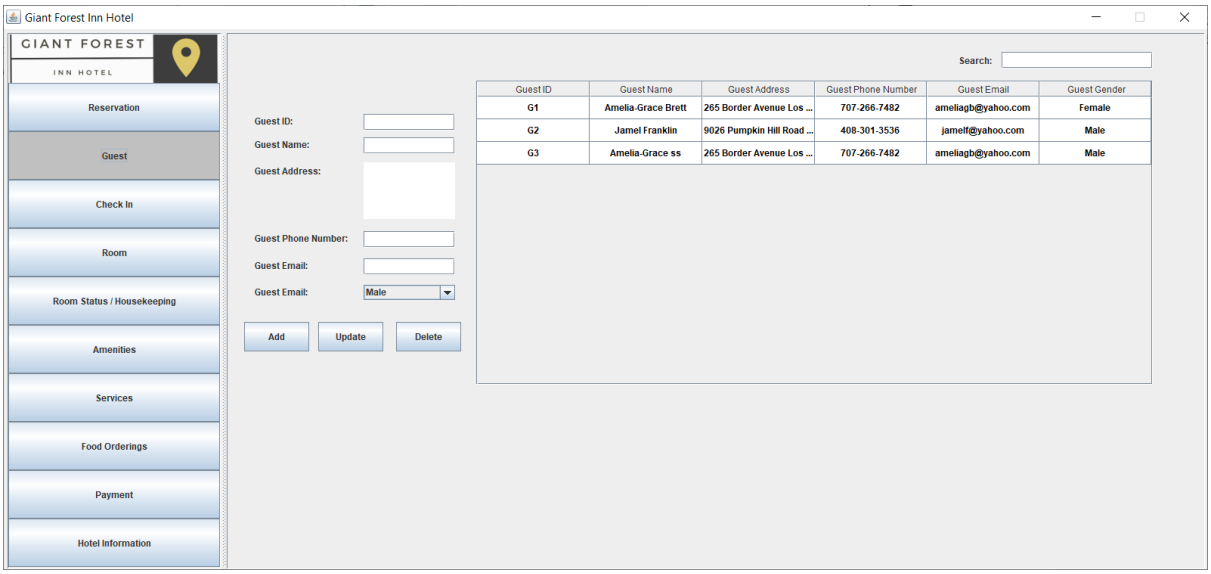

Figure 20: Guest Interface for Backend System

3) Check In Display

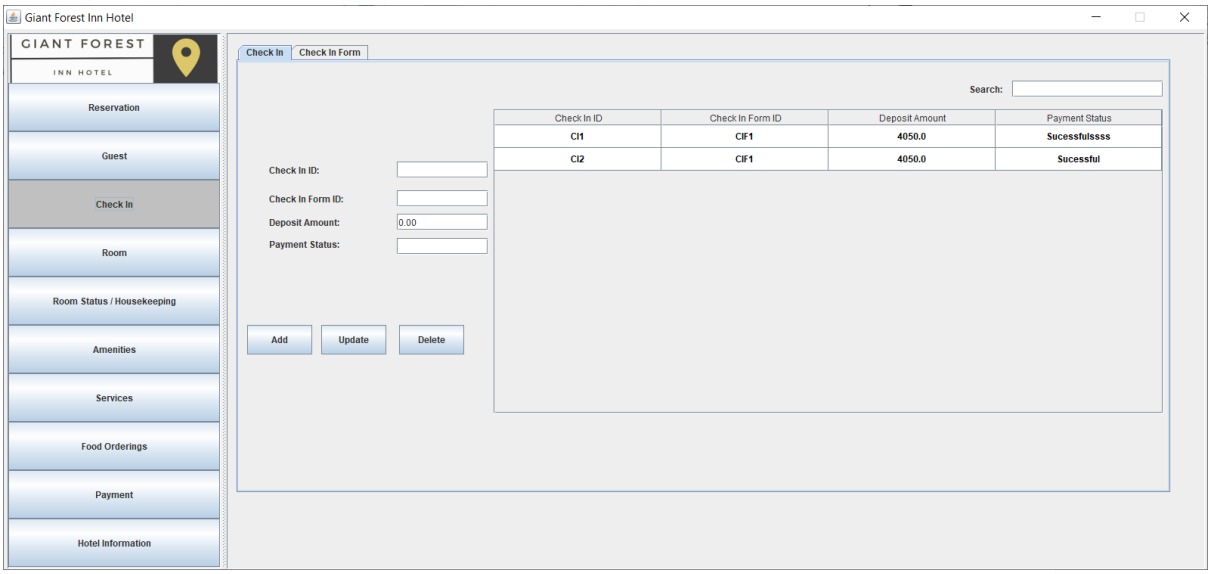

Figure 21: Check In Interface for Backend System

4) Room Display

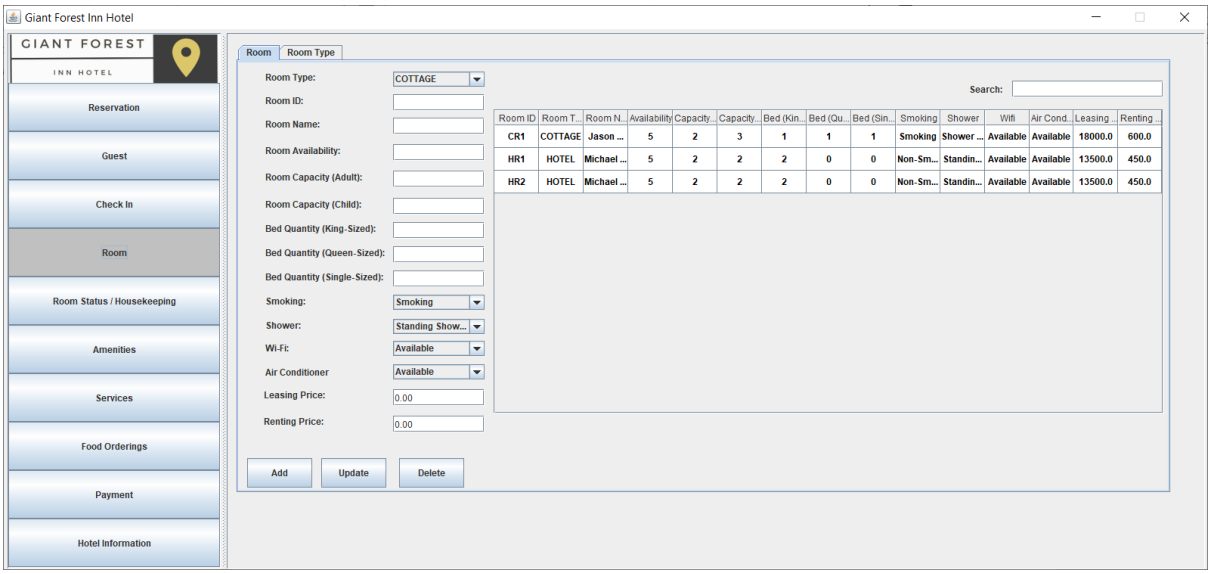

Figure 22: Room Interface for Backend System

5) Room Status Display

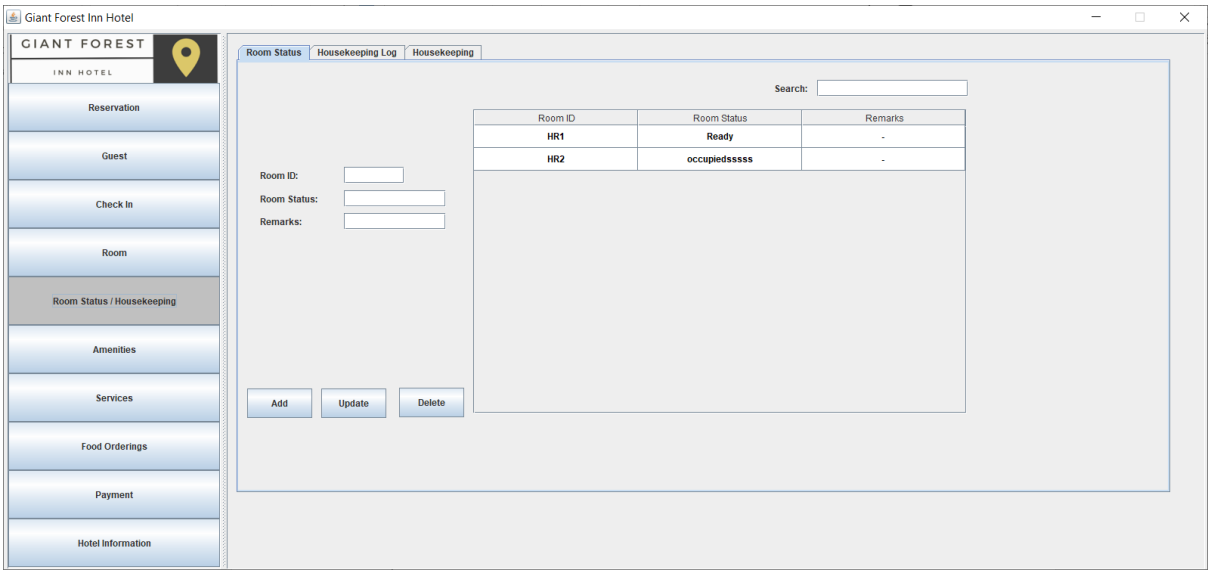

Figure 23: Room Status Interface for Backend System

6) Amenities Display

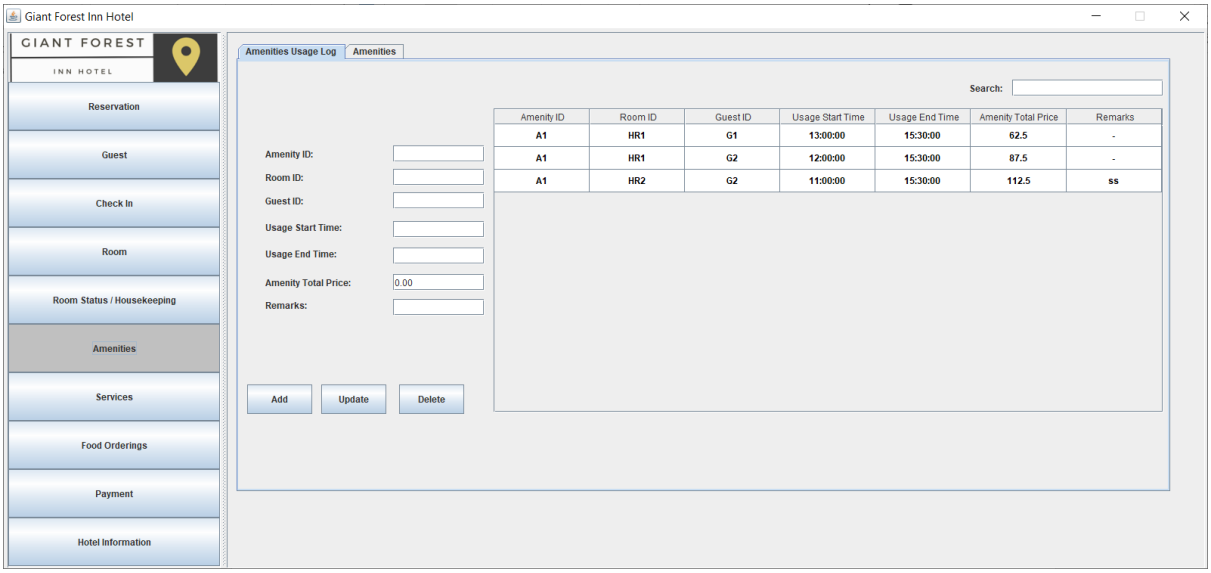

Figure 24: Amenities Interface for Backend System

7) Services Display

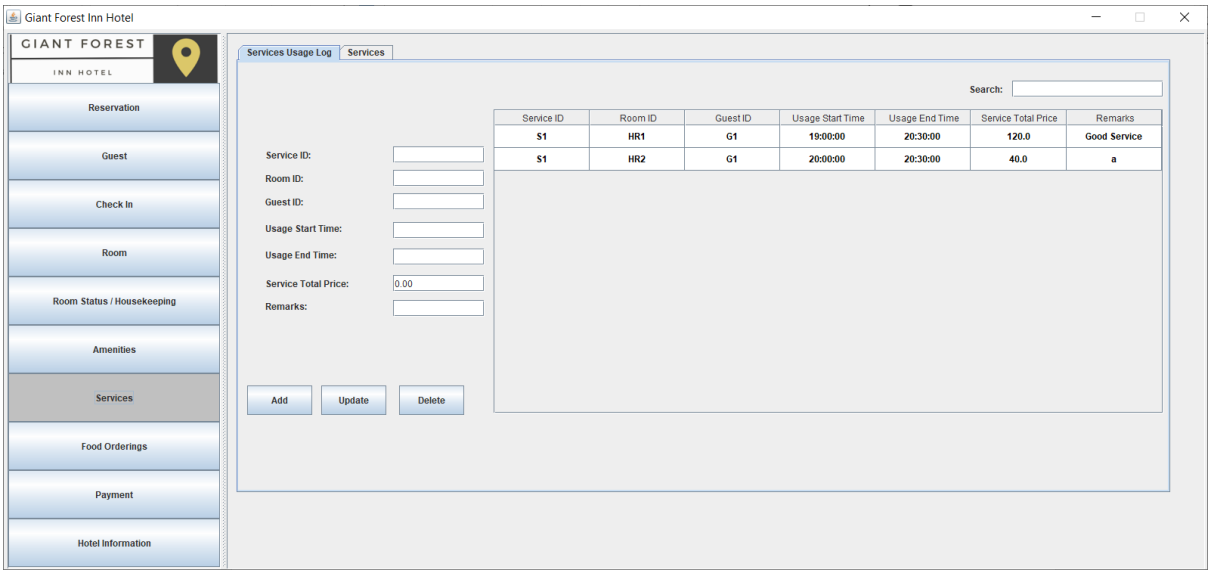

Figure 25: Services Interface for Backend System

8) Food Ordering Display

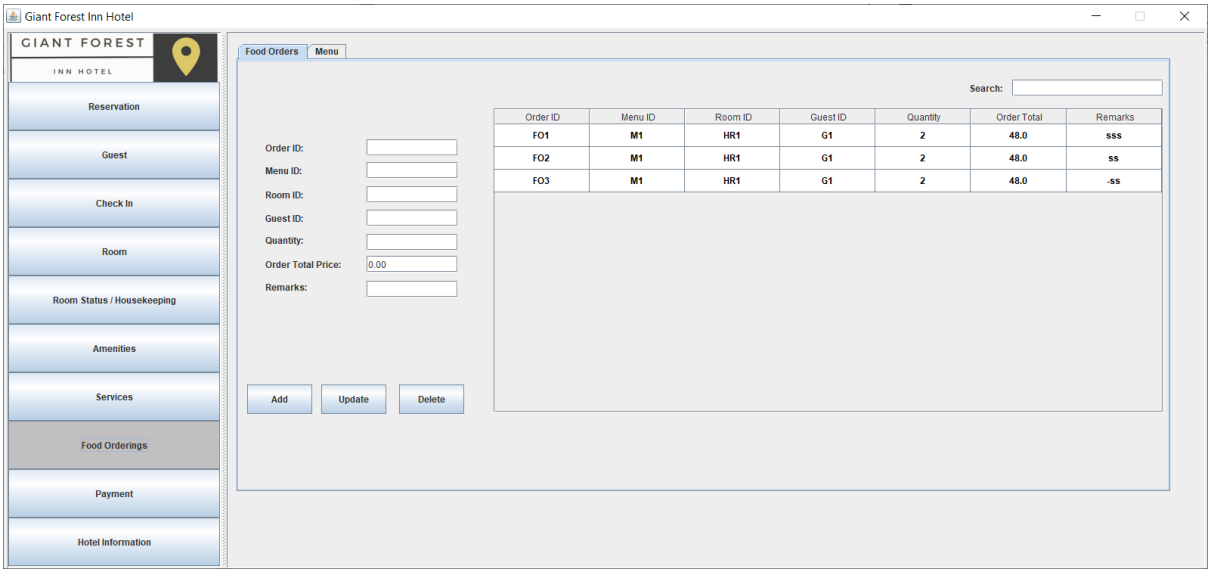

Figure 26: Food Ordering Interface for Backend System

9) Payment Display

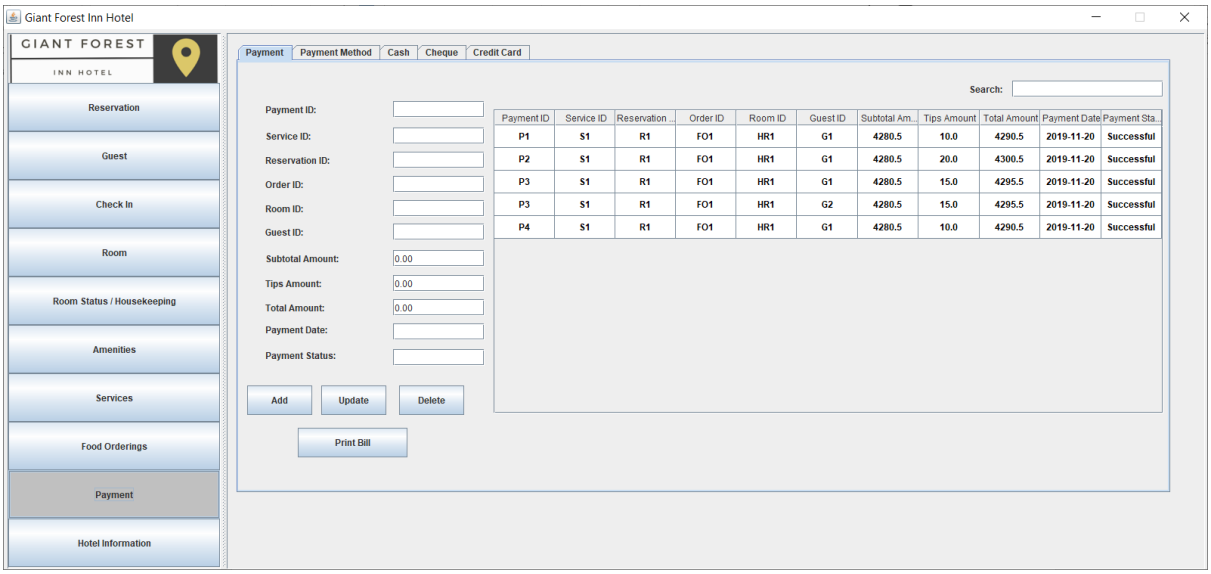

Figure 27: Payment Interface for Backend System

10) Hotel Information Display

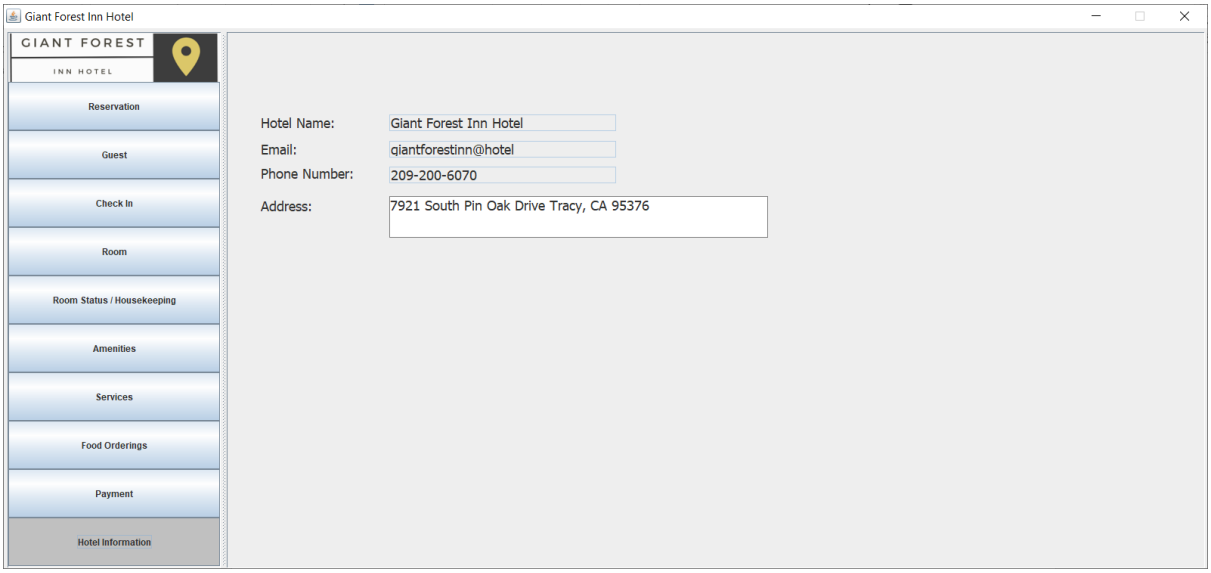

Figure 28: Hotel Information Interface for Backend System

For the development of the backend system, the codes are develop using the object-oriented concept. The application of the concepts will be shown with supporting screenshots of the codes.

All the variables are encapsulated as shown in Figure 29 and they are accessed using a getter and setter for each variable created so that there is no direct modification done to the actual variable. This is where encapsulation is important as it helps create programs that are easier to maintain and debug as they are used to create abstract datatypes that access the variables only through their external interface [1]. Figure 29 below shows the encapsulation concept implemented in the system.

```
// variables for hotel
private String hotelName, hotelAddress, hotelPhoneNumber, hotelEmail;

public String getHotelName() {
    return hotelName;
}
public void setHotelName(String hotelName) {
    this.hotelName = hotelName;
}
public String getHotelAddress() {
    return hotelAddress;
}
public void setHotelAddress(String hotelAddress) {
    this.hotelAddress = hotelAddress;
}
public String getHotelPhoneNumber() {
    return hotelPhoneNumber;
}
public void setHotelPhoneNumber(String hotelPhoneNumber) {
    this.hotelPhoneNumber = hotelPhoneNumber;
}
public String getHotelEmail() {
    return hotelEmail;
}
public void setHotelEmail(String hotelEmail) {
    this.hotelEmail = hotelEmail;
}
```

Figure 29: Encapsulation in System Development

As there are a number of tables and variables that are present in the system, instead of creating a new method to call for each set of variables or table, the same method can be used repeatedly by changing its operation. This concept is also known as polymorphism, where is refers to the ability of a variable to be bound to entities of multiple types and hiding the behavior of the variable in behind a uniform interface [2]. Figure 30 shows the polymorphism application in the system.

```
//hotel Constructor
public CRUD_1(String hotelName, String hotelAddress, String hotelPhoneNumber, String hotelEmail){
    this.hotelName = hotelName;
    this.hotelAddress = hotelAddress;
    this.hotelPhoneNumber = hotelPhoneNumber;
    this.hotelEmail = hotelEmail;
}


//roomType Constructor
public CRUD_1(String roomTypeID, String roomTypeName) {
    this.roomTypeID = roomTypeID;
    this.roomTypeName = roomTypeName;
}
```

Figure 30: Polymorphism in System Development

The system also includes inheritance object-oriented concept by creating a superclass and allowing subclasses to utilize the methods created in the superclass. This is done so that instead of redefining the attributes and methods that are already defined in another class, a subclass can inherit these properties and only implementing the properties it requires, which will result in the reduction of code redundancy [3]. Figure 31 and Figure 32 shows the inheritance concept used in the system.

```
public class HotelDetails {
    private ArrayList<CRUD_1> hotelDetails;

    //Constructor that contains the ArrayList that is going to be manipulated
    public HotelDetails(ArrayList<CRUD_1> hotelDetails) {
        hotelDetails=new ArrayList<>();
    }
```

Figure 31: Inheritance in System Development (Superclass)

```
public class Room extends HotelDetails {

    public Room(ArrayList<CRUD_1> room) {
        super(room);
    }
```

Figure 32: Inheritance in System Development (Subclass)

The test plan of the system shown in Appendix B: Test Plan in the section B.2 Back-End Testing for the testing and evaluations done on the performance of the system.

Reference

[1] M. Skoglund, “Practical Use of Encapsulation in Object-Oriented Programming,” in *Proceedings of the International Conference on Software Engineering Research and Practise*, 2003.

[2] N. Milojković, A. Caracciolo, M. F. Lungu, O. Nierstrasz, D. Röthlisberger, and R. Robbes, “Polymorphism in the Spotlight: Studying Its Prevalence in Java and Smalltalk,” in *IEEE International Conference on Program Comprehension*, 2015.

[3] J. A. L. Dallal, “The impact of inheritance on the internal quality attributes of Java classes,” *Kuwait J. Sci. Eng.*, 2012.

Appendix A: Class Diagram

A.1 Overview of Class Diagram Relationships

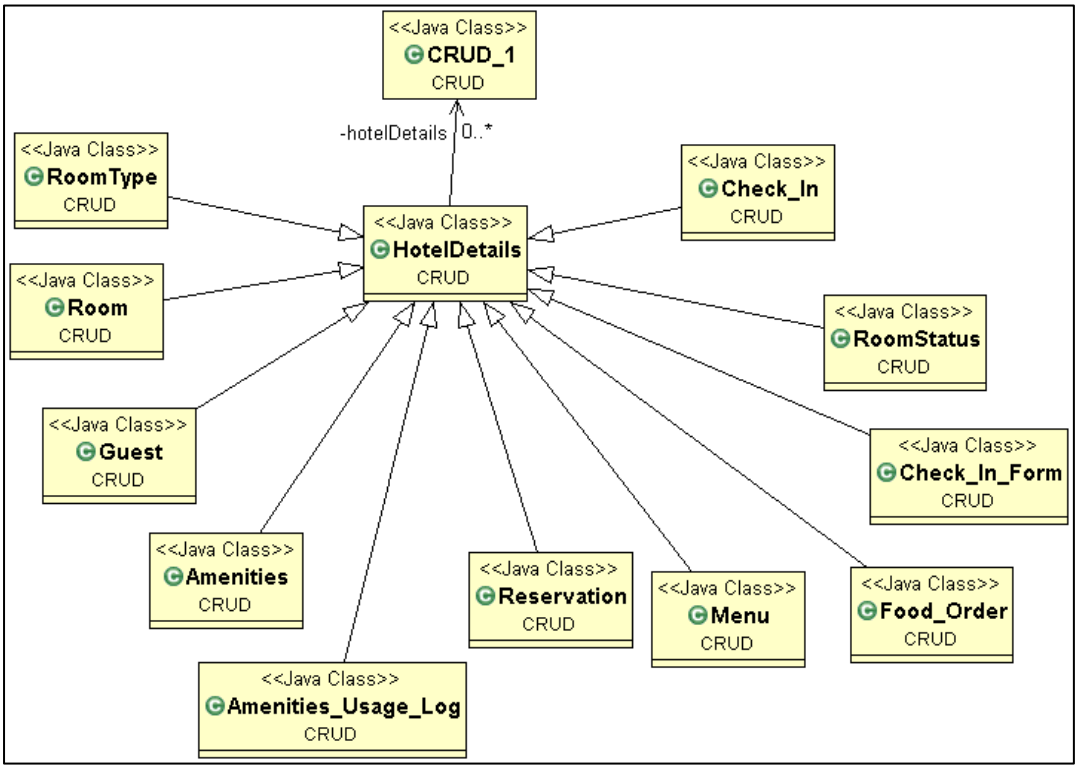

Figure 33:Class Diagram - Inheritance Relationship between classes that handles Create, Retrieve, Update and Delete (1)

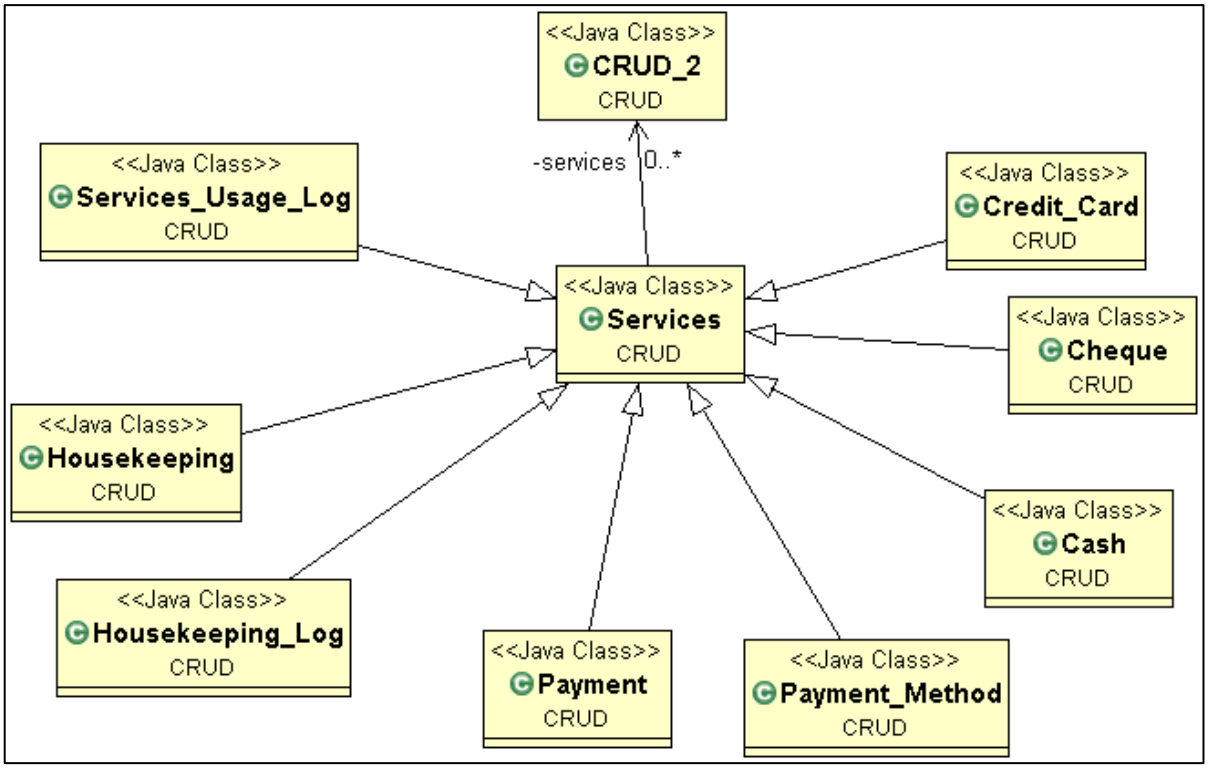

Figure 34:Class Diagram - Inheritance Relationship between classes that handles Create, Retrieve, Update and Delete (2)

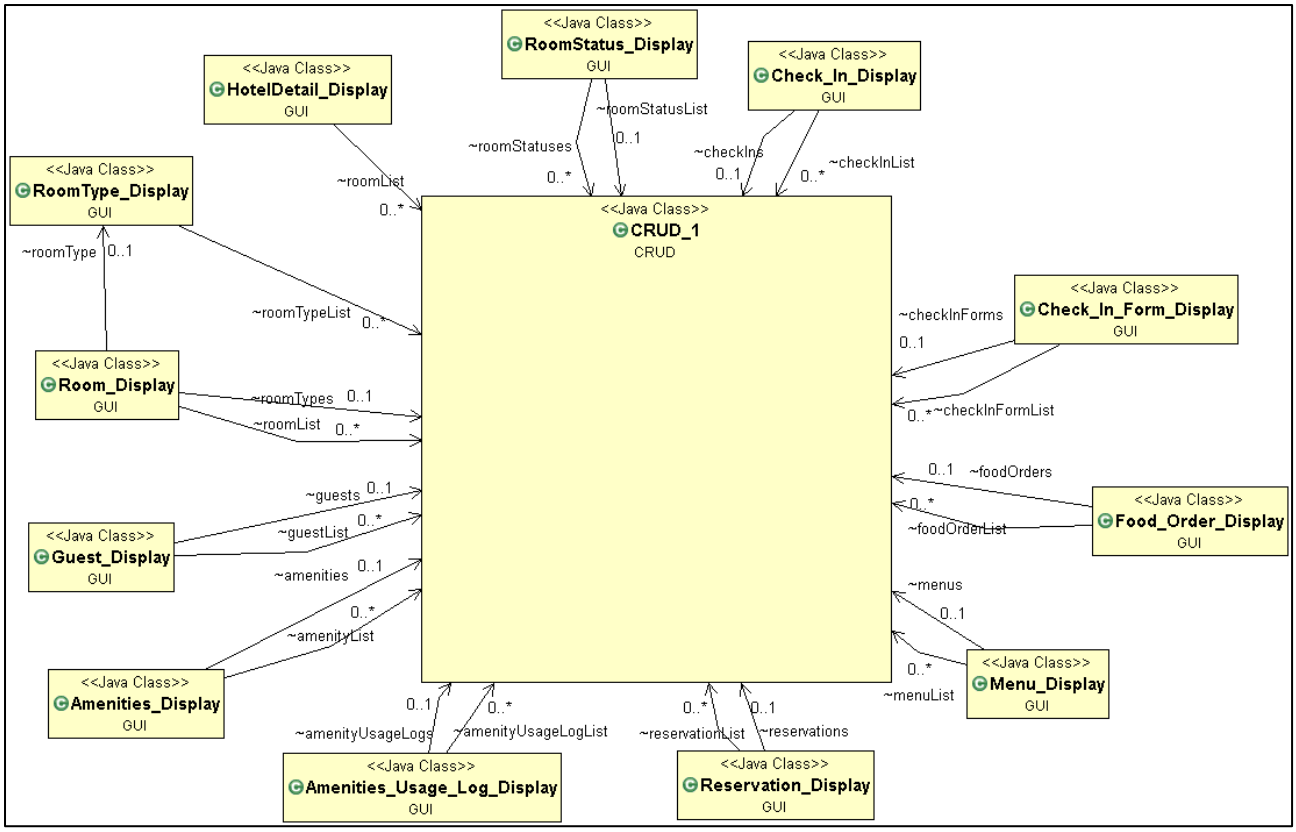

Figure 35:Class Diagram - Inheritance Relationship between classes that handles Display (1)

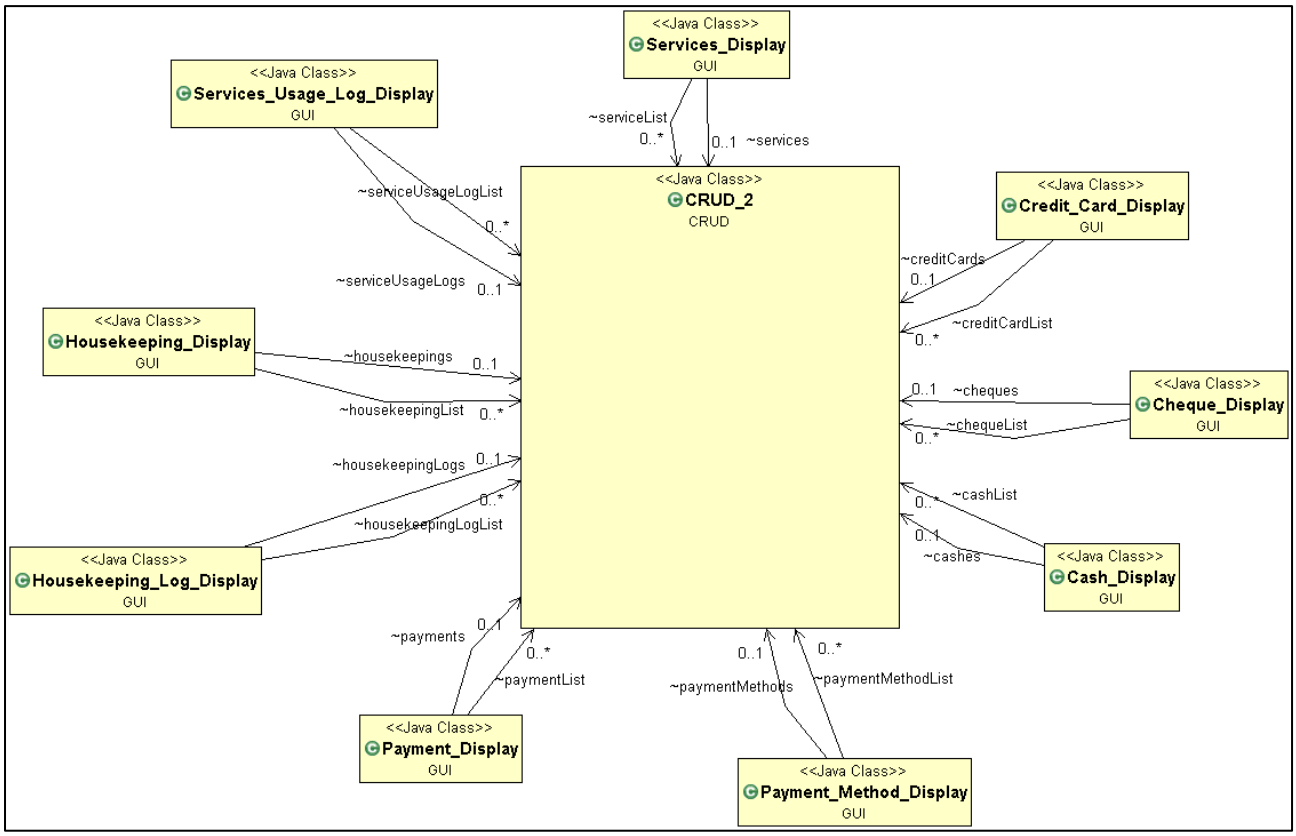

Figure 36:Class Diagram - Inheritance Relationship between classes that handles Display (2)

A.2 Detailed of Class Diagram Relationships

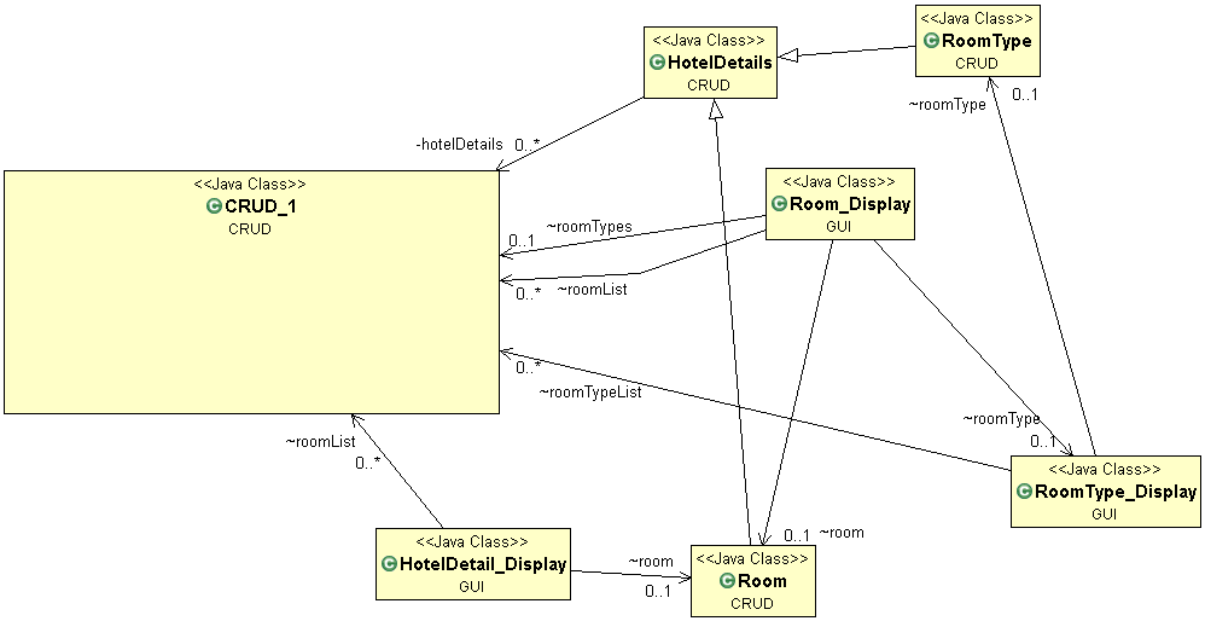

Figure 37:Class Diagram - Relationship between Hotel Detail, Room & Room Type classes

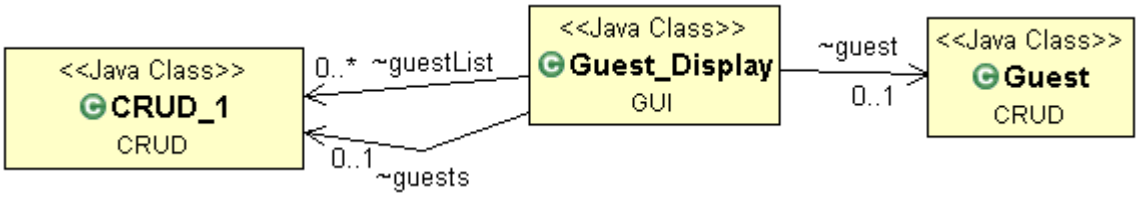

Figure 38:Class Diagram - Relationship between Guest classes

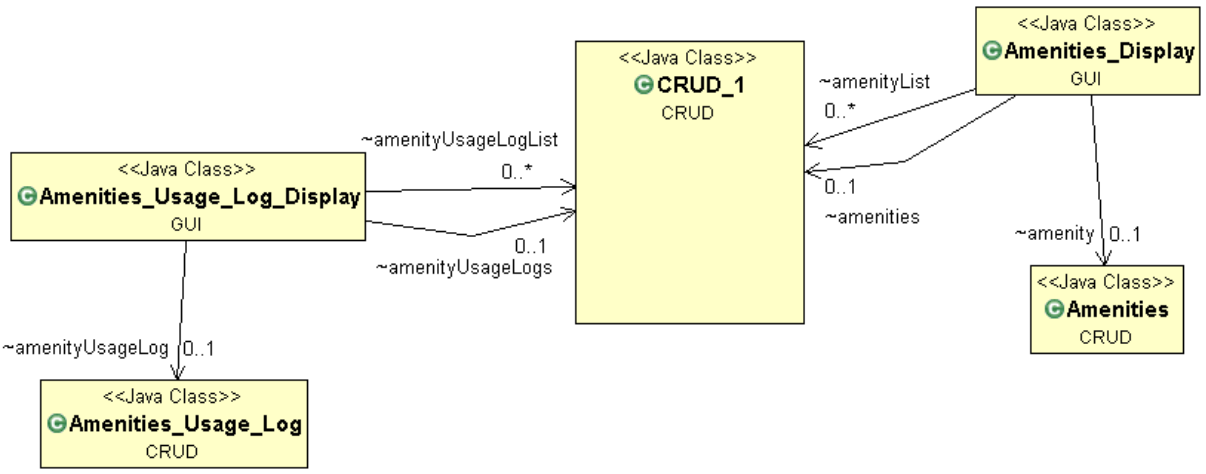

Figure 39:Class Diagram - Relationship between Amenities and Amenities Usage Log classes

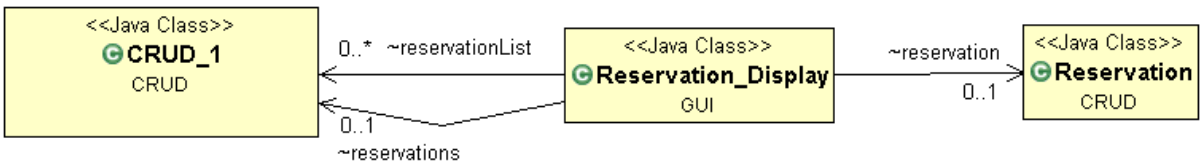

Figure 40:Class Diagram - Relationship between Reservation classes

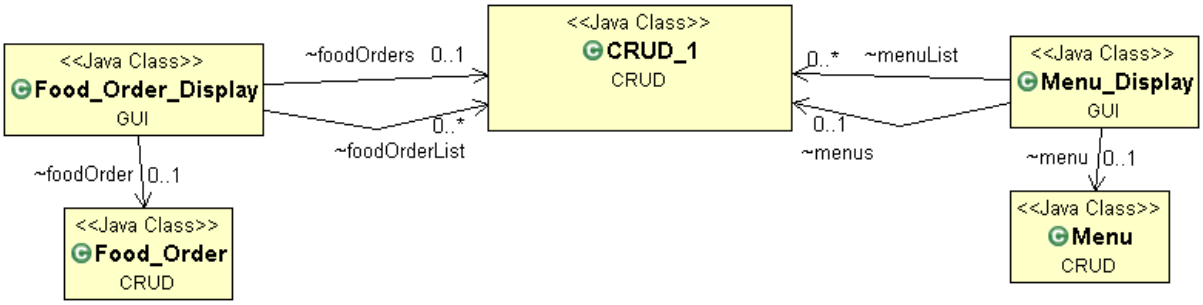

Figure 41:Class Diagram - Relationship between Menu and Food Order classes

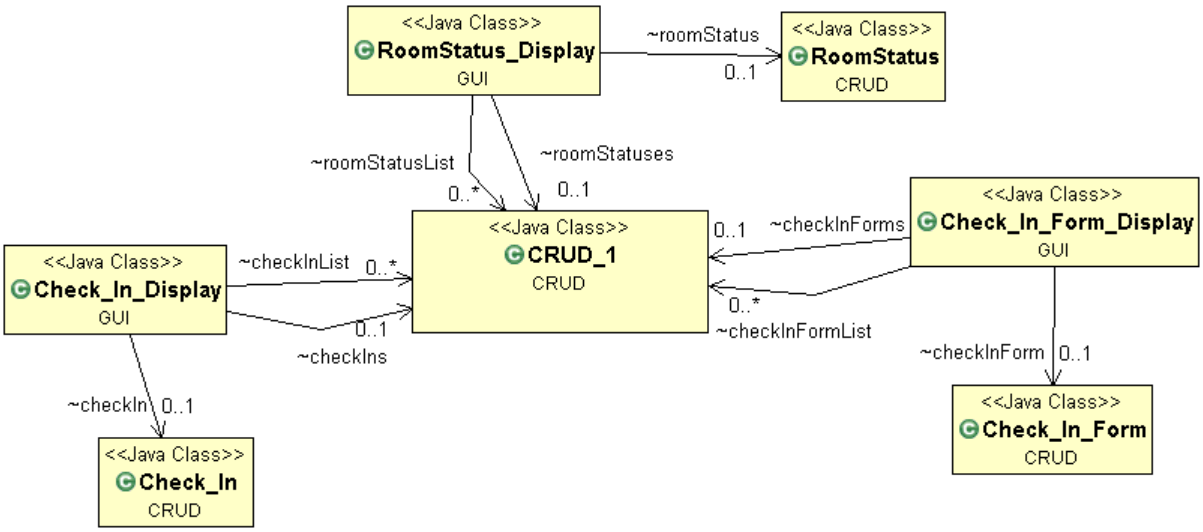

Figure 42:Class Diagram - Relationship between Room Status, Check In and Check In Form classes

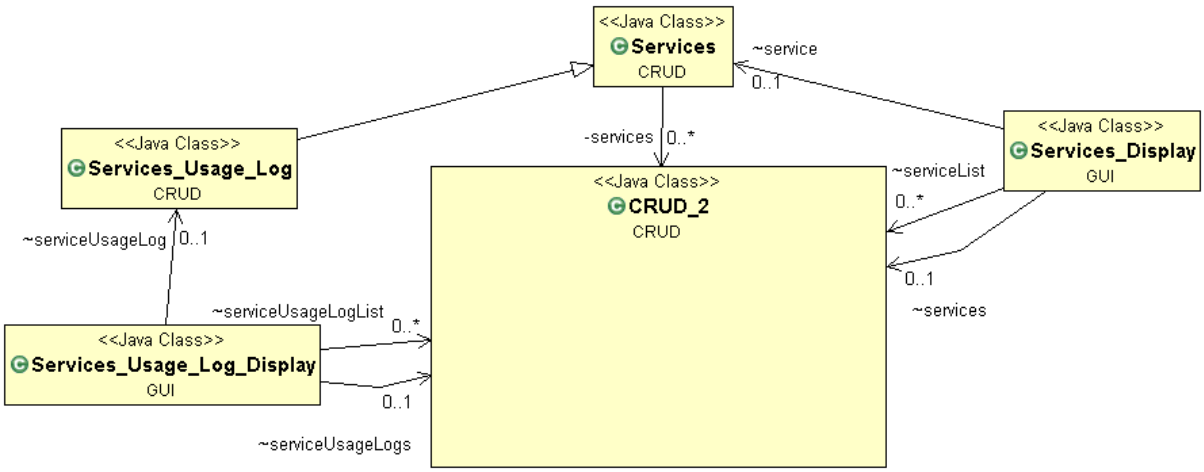

Figure 43:Class Diagram - Relationship between Services and Services Usage Log classes

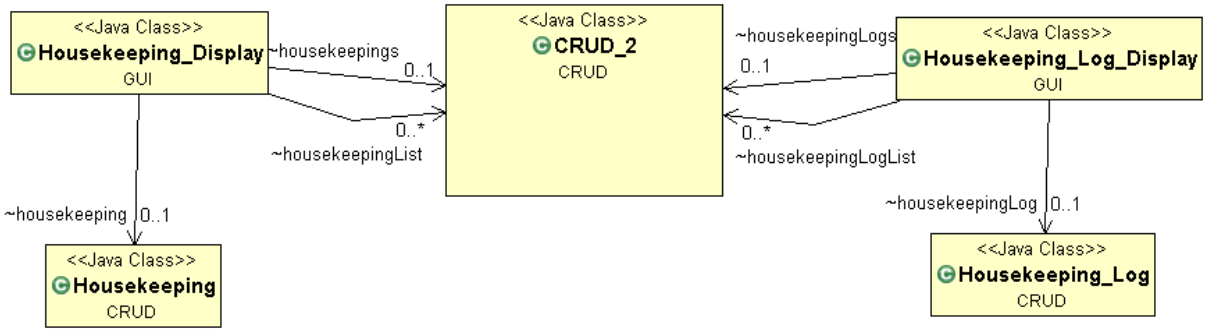

Figure 44:Class Diagram - Relationship between Housekeeping and Housekeeping classes

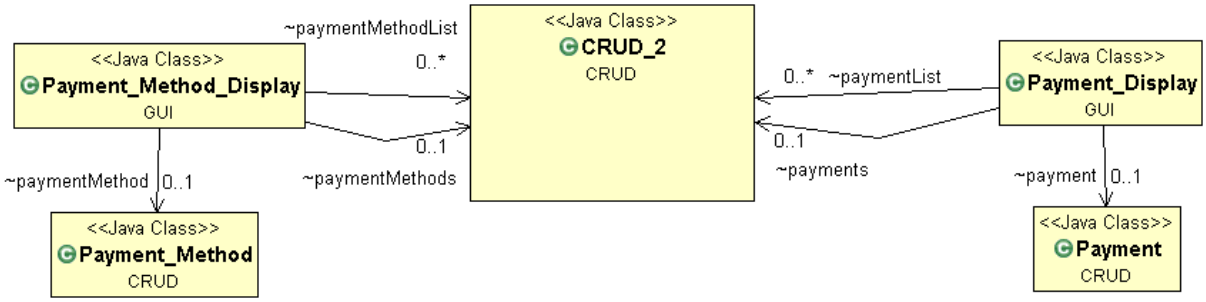

Figure 45:Class Diagram - Relationship between Payment and Payment Method classes

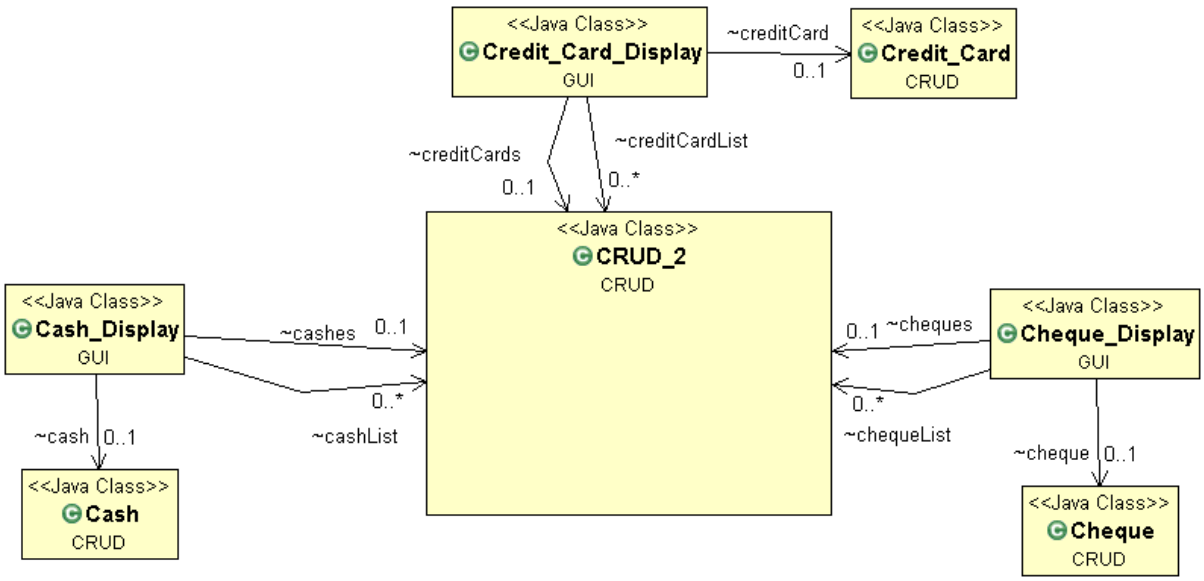

Figure 46:Class Diagram - Relationship between Cash, Cheque and Credit Card classes

A.3 Attributes and Operations of Individual Class Diagrams

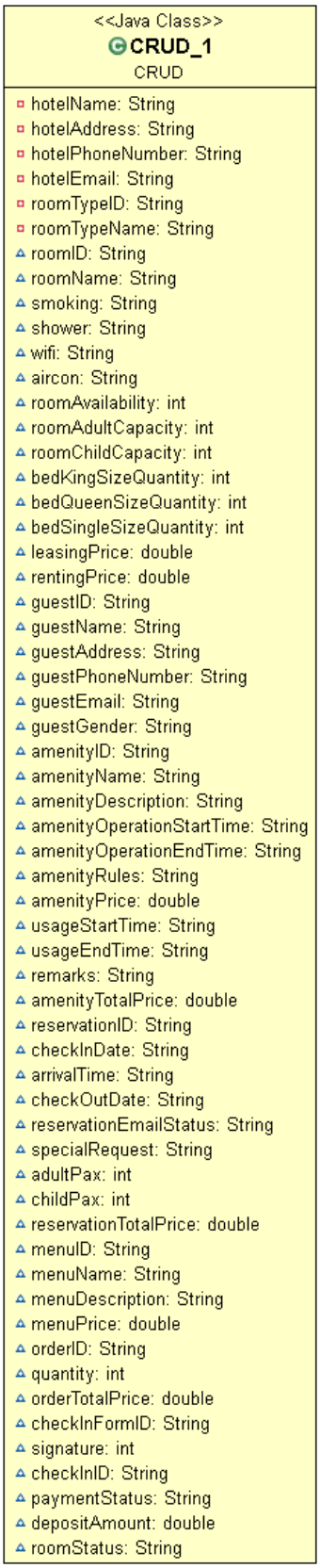

Figure 47:Class Diagram – CRUD1 Attributes

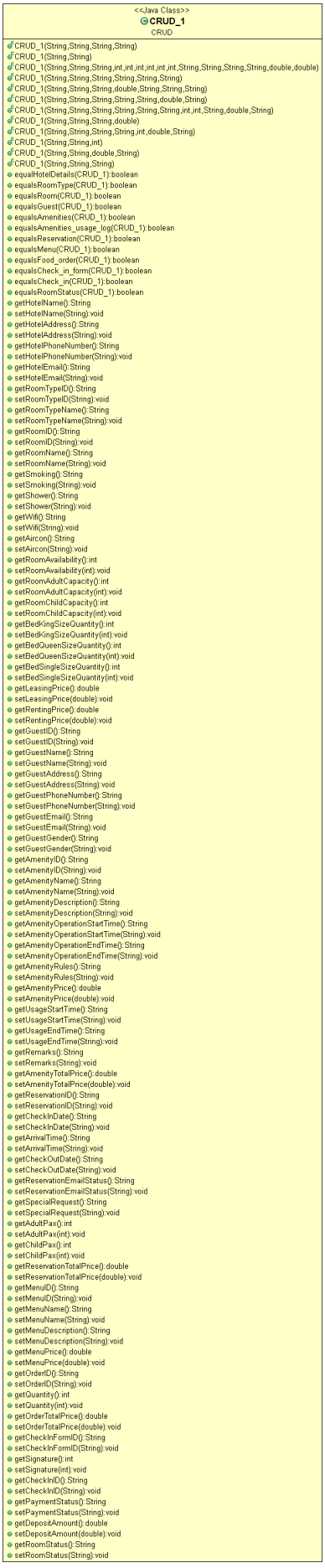

Figure 48:Class Diagram – CRUD1 Operations

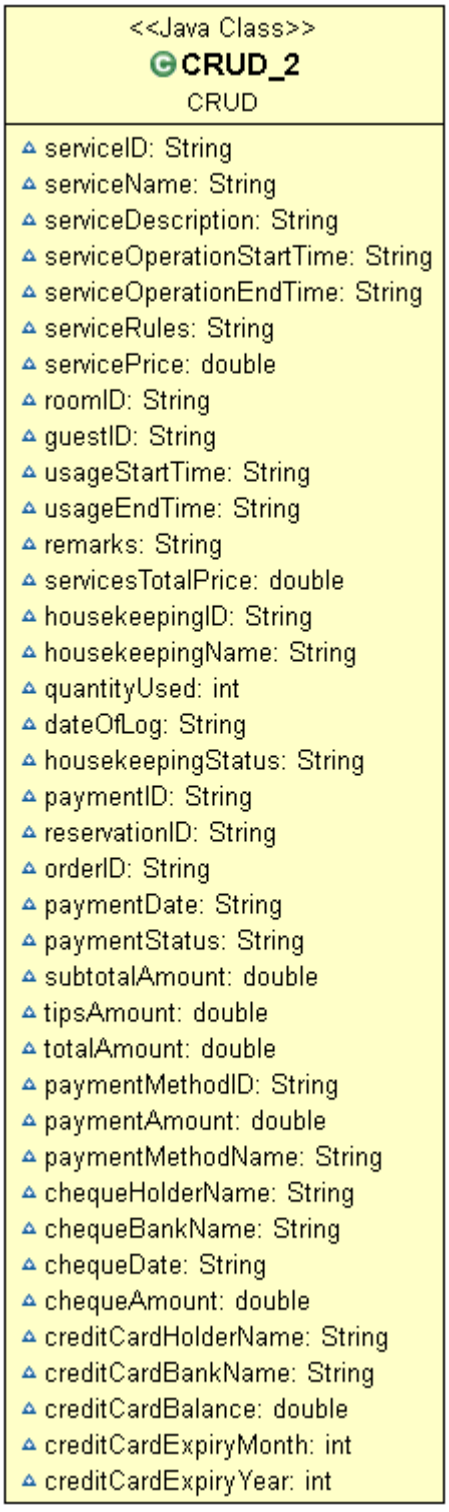

Figure 49:Class Diagram – CRUD2 Attributes

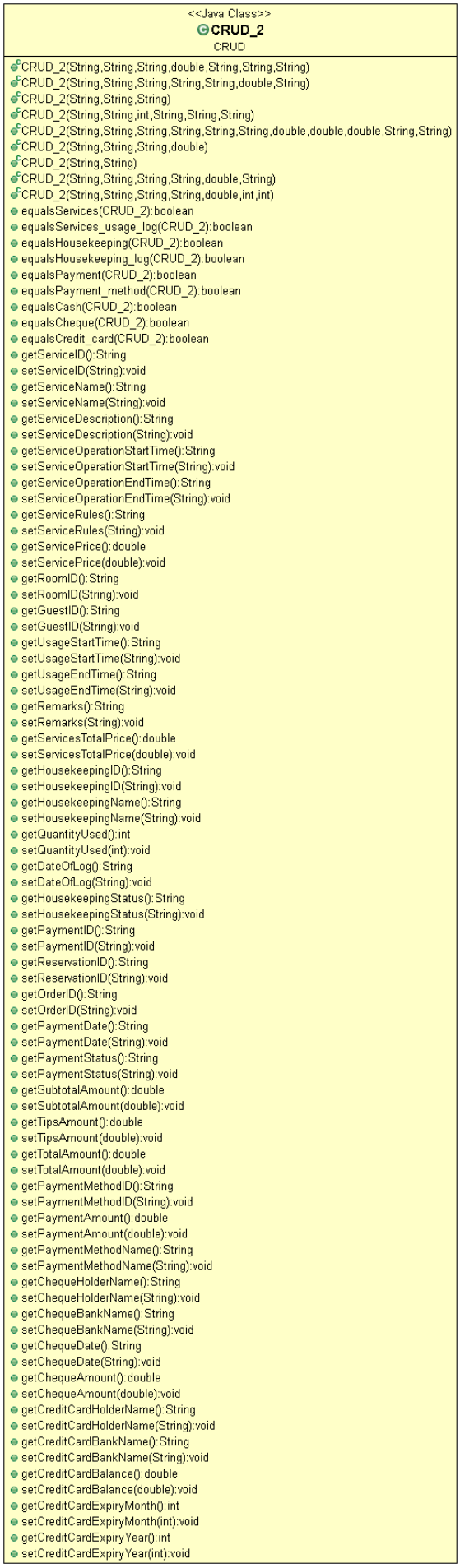

Figure 50:Class Diagram – CRUD2 Operations

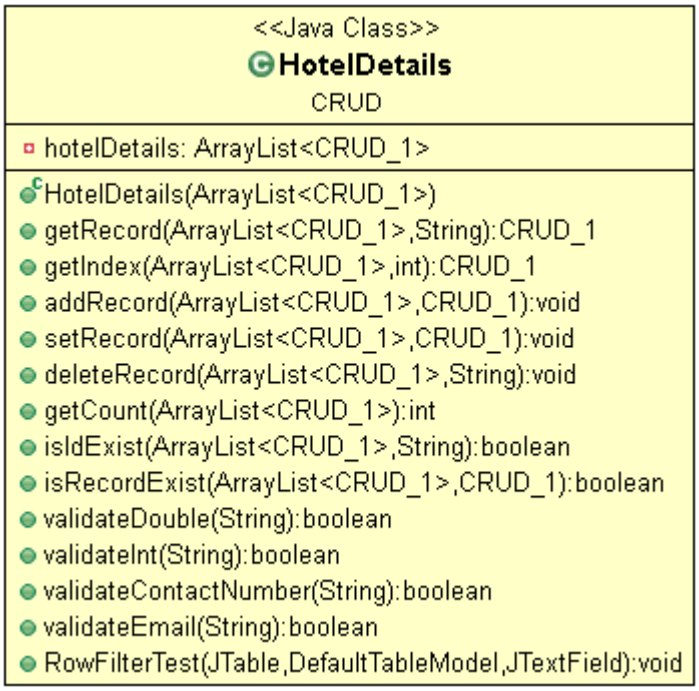

Figure 51:Class Diagram – Hotel Details

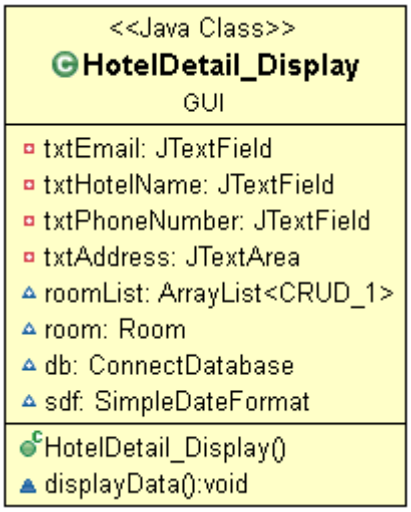

Figure 52:Class Diagram – Hotel Details Display

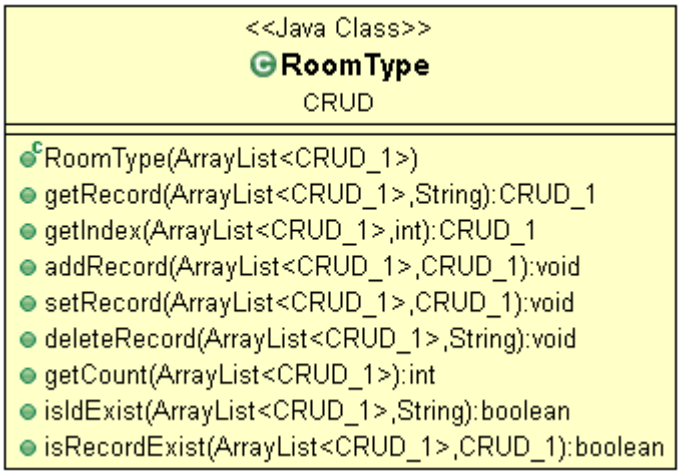

Figure 53:Class Diagram – Room Type

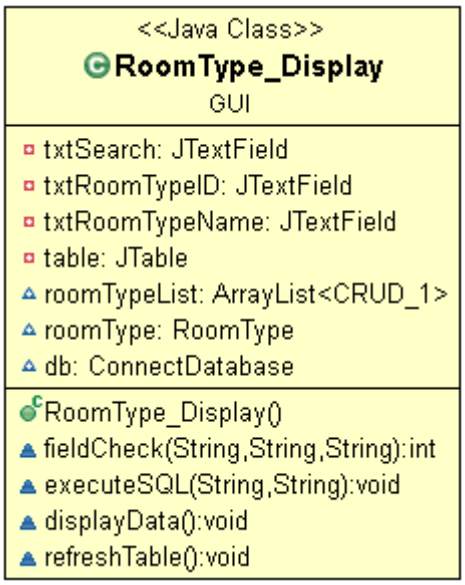

Figure 54:Class Diagram – Room Type Display

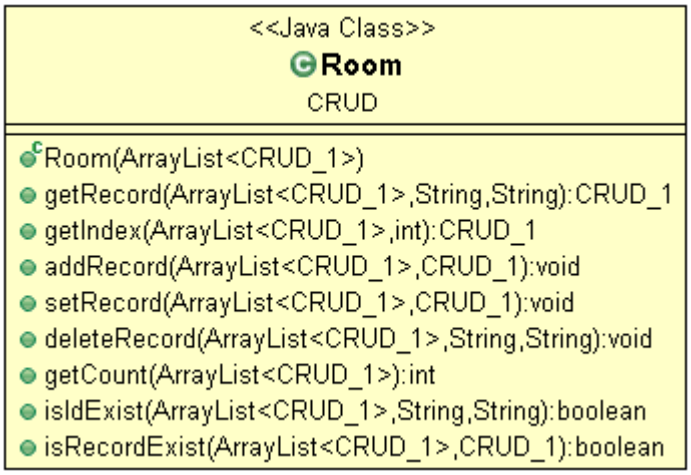

Figure 55:Class Diagram – Room

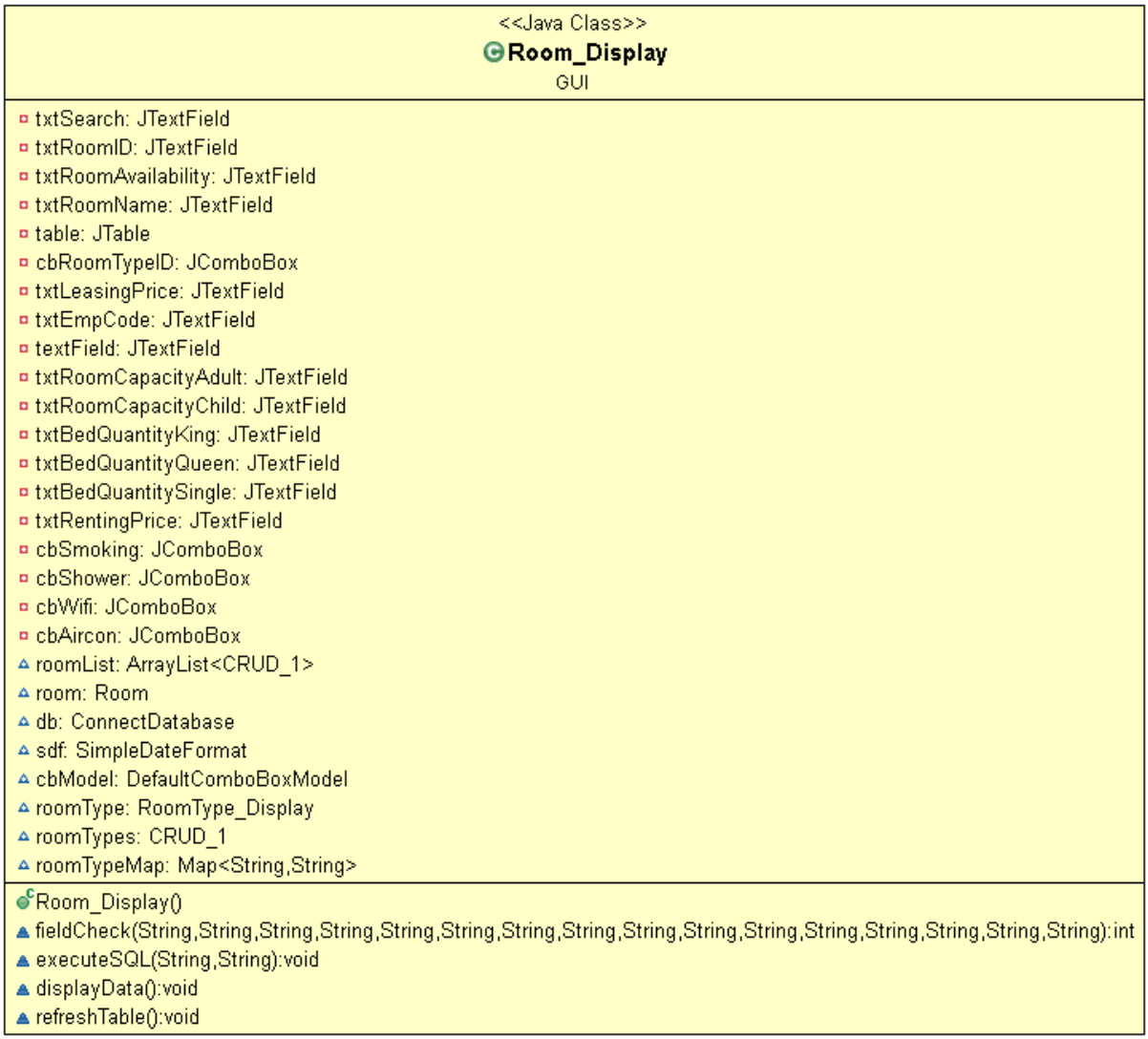

Figure 56:Class Diagram – Room Display

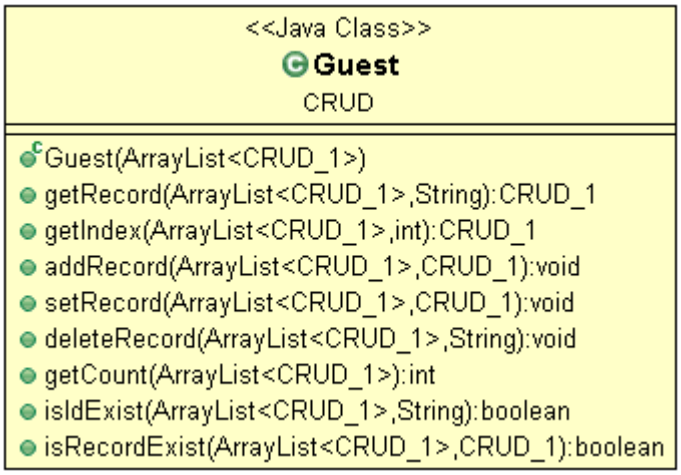

Figure 57:Class Diagram – Guest

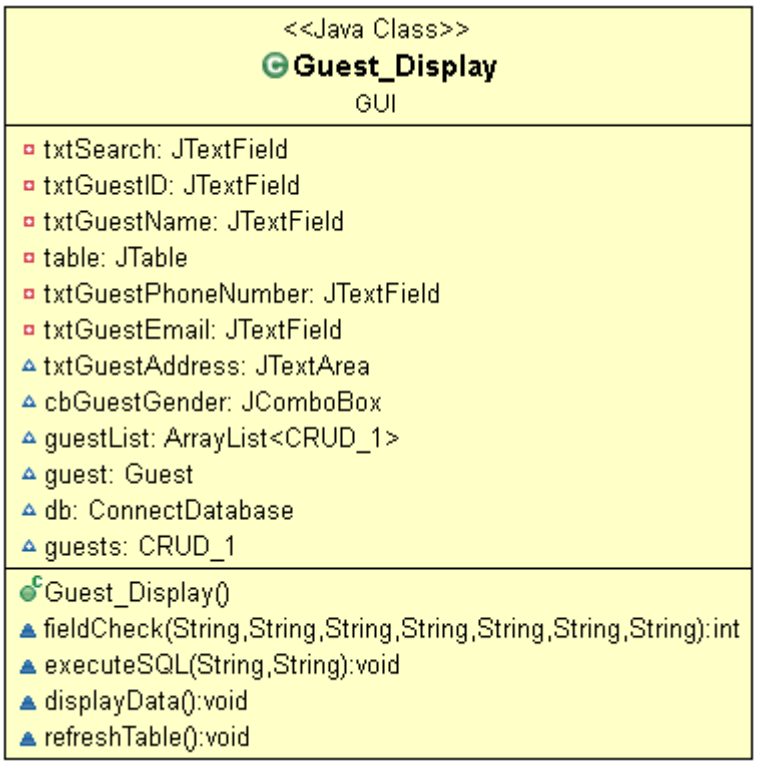

Figure 58:Class Diagram – Guest Display

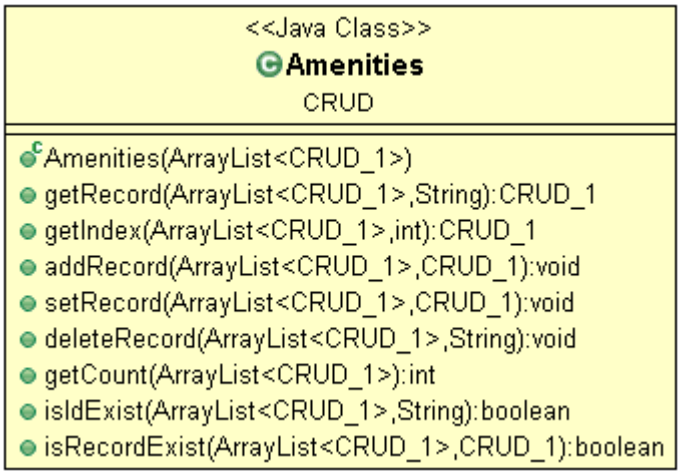

Figure 59:Class Diagram – Amenities

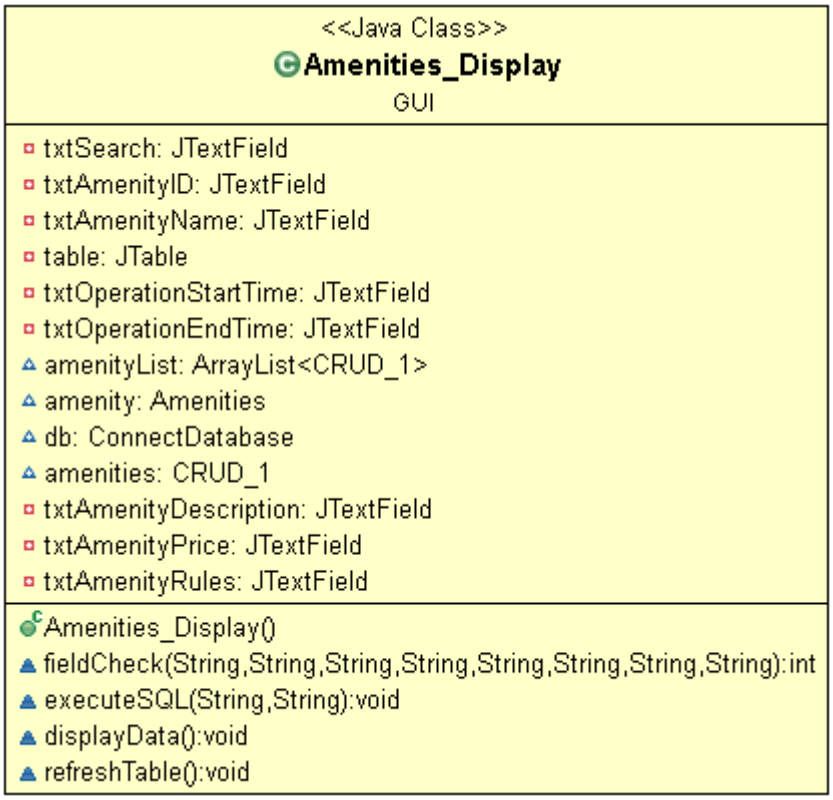

Figure 60:Class Diagram – Amenities Display

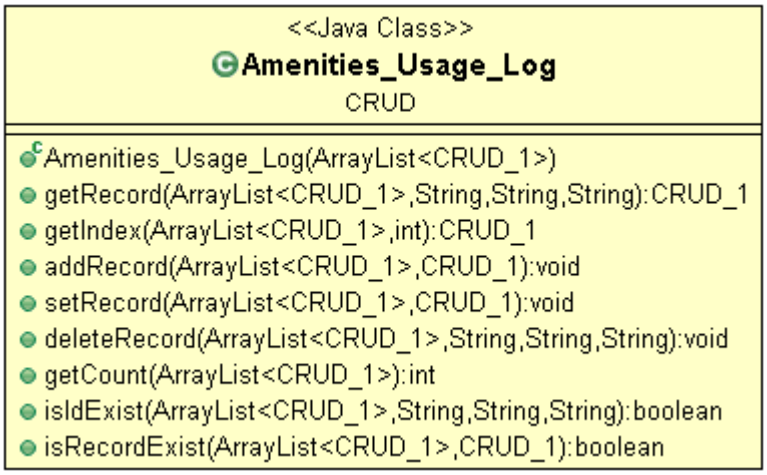

Figure 61:Class Diagram – Amenities Usage Log

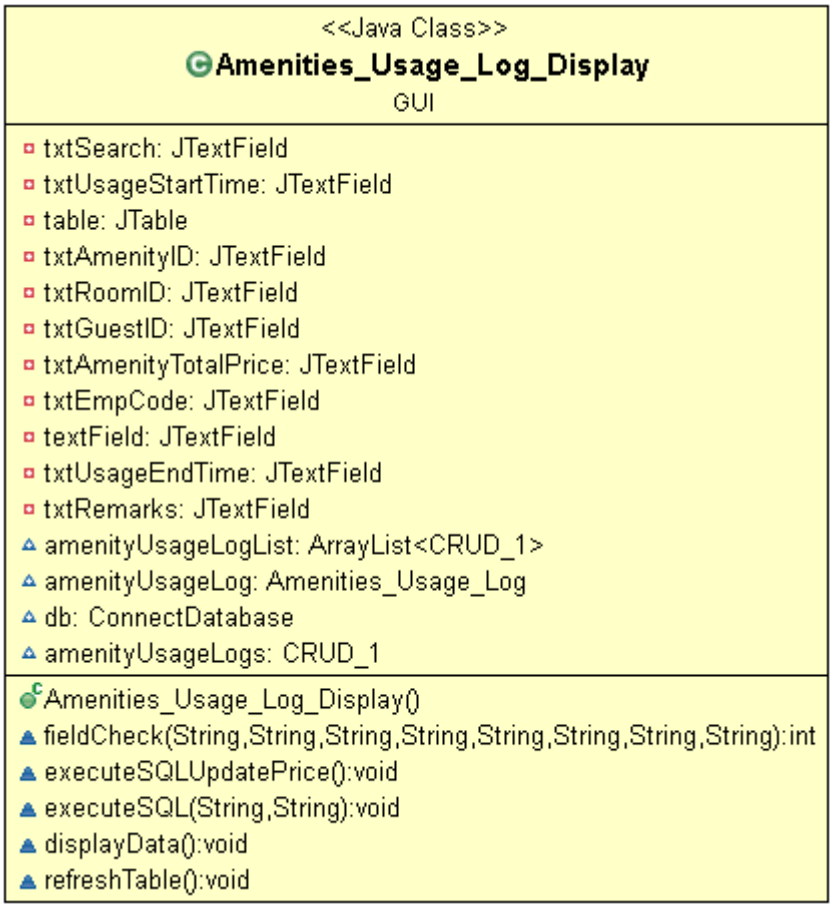

Figure 62:Class Diagram – Amenities Usage Log Display

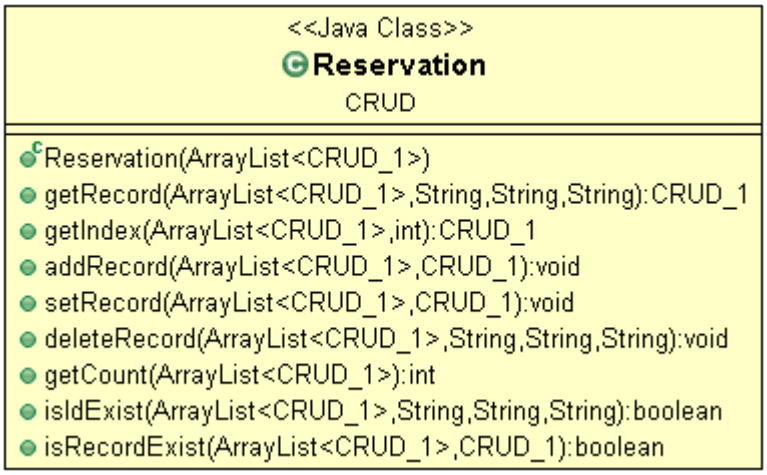

Figure 63:Class Diagram – Reservation

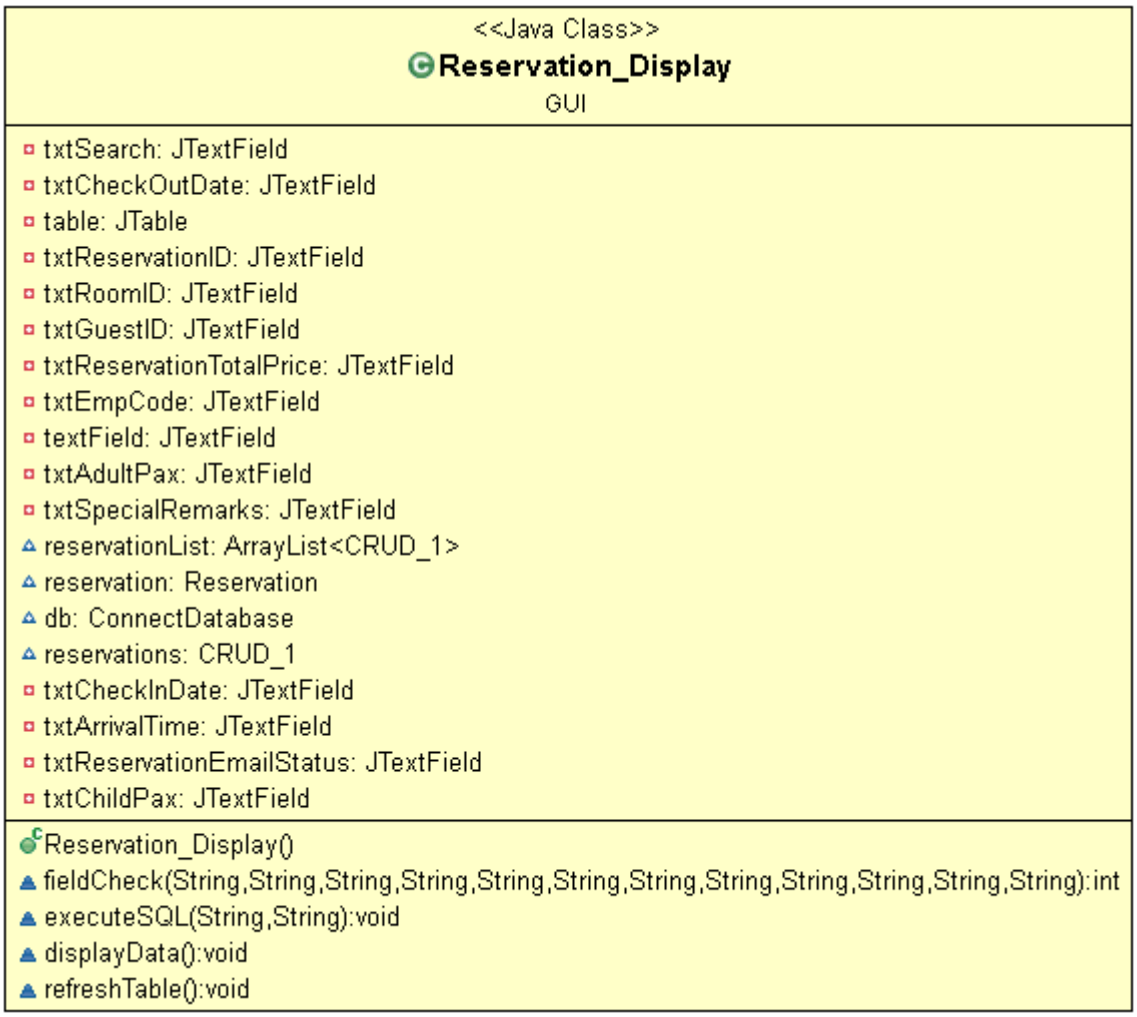

Figure 64:Class Diagram – Reservation Display

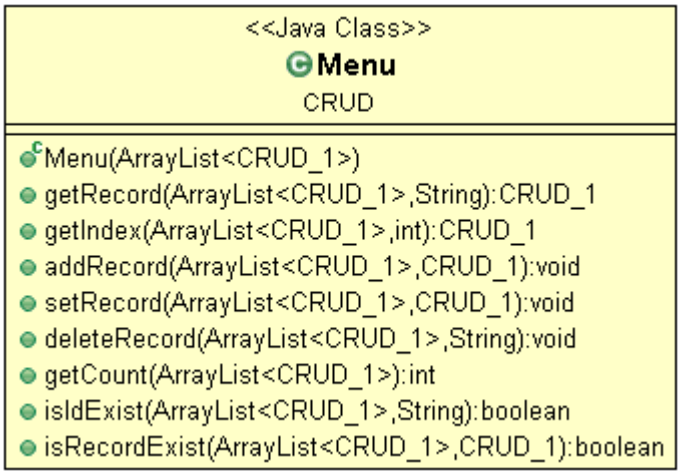

Figure 65:Class Diagram – Menu

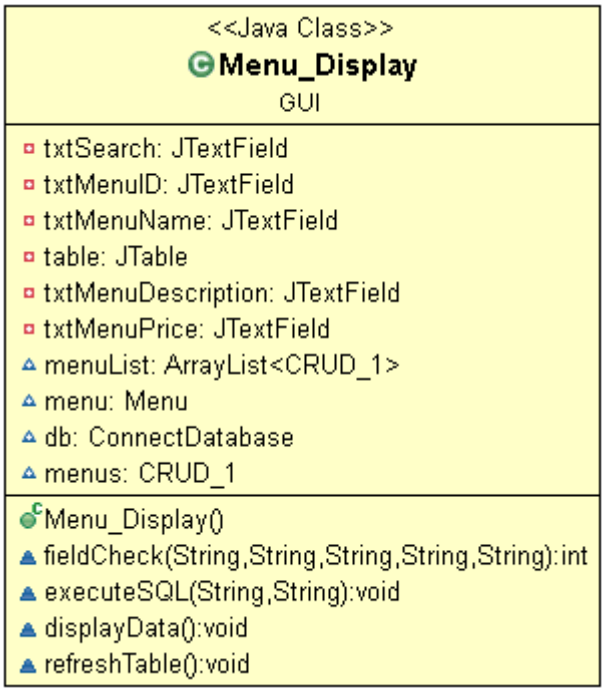

Figure 66:Class Diagram – Menu Display

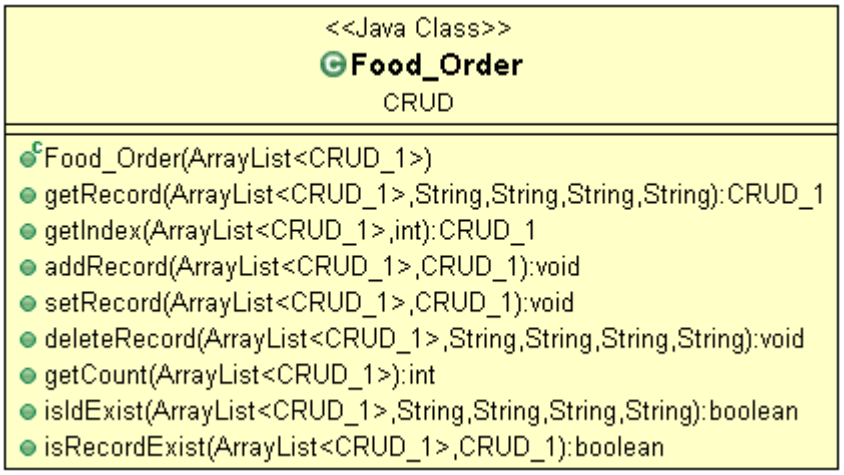

Figure 67:Class Diagram – Food Order

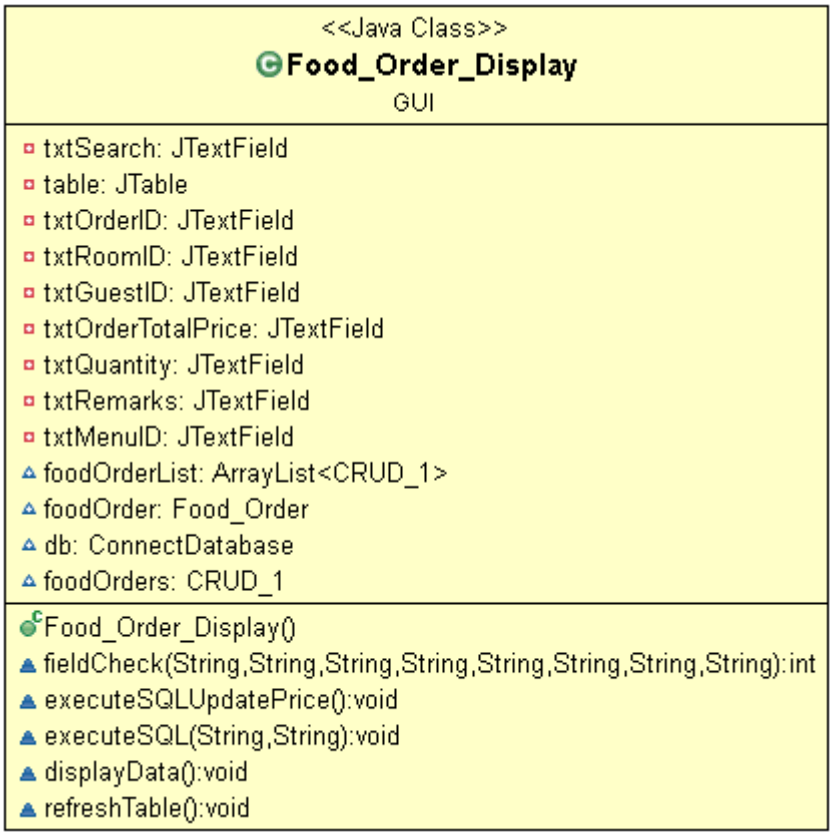

Figure 68:Class Diagram – Food Order Display

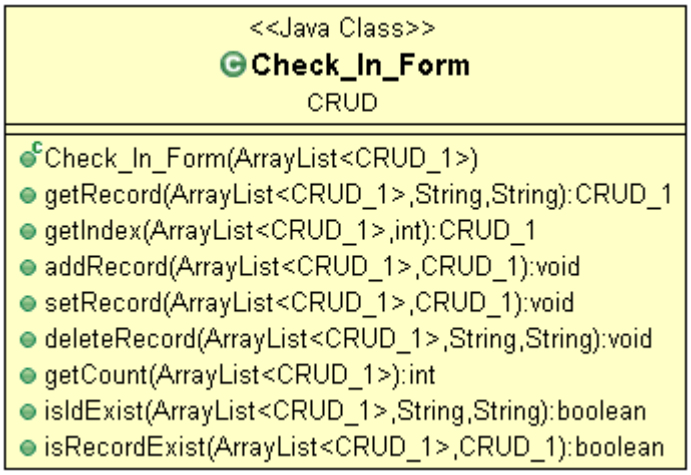

Figure 69:Class Diagram – Check in Form

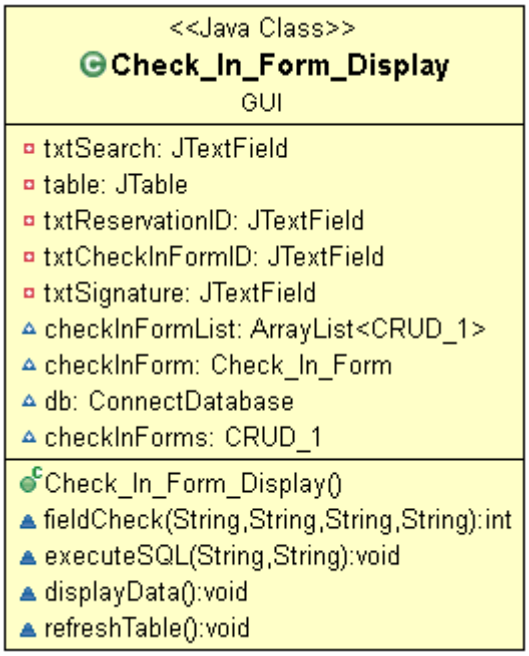

Figure 70:Class Diagram – Check in Form Display

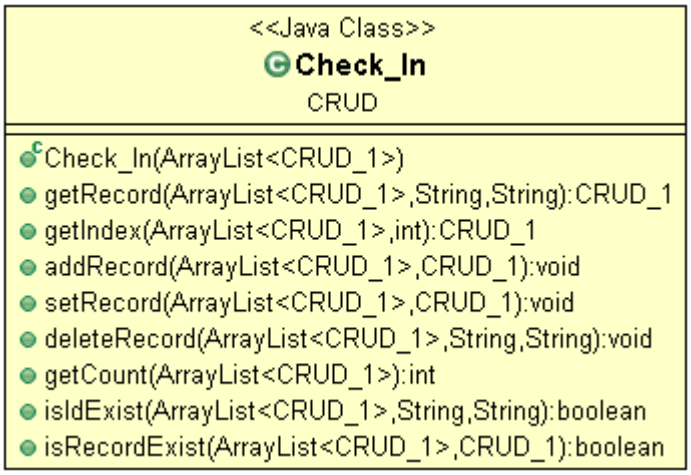

Figure 71:Class Diagram – Check in

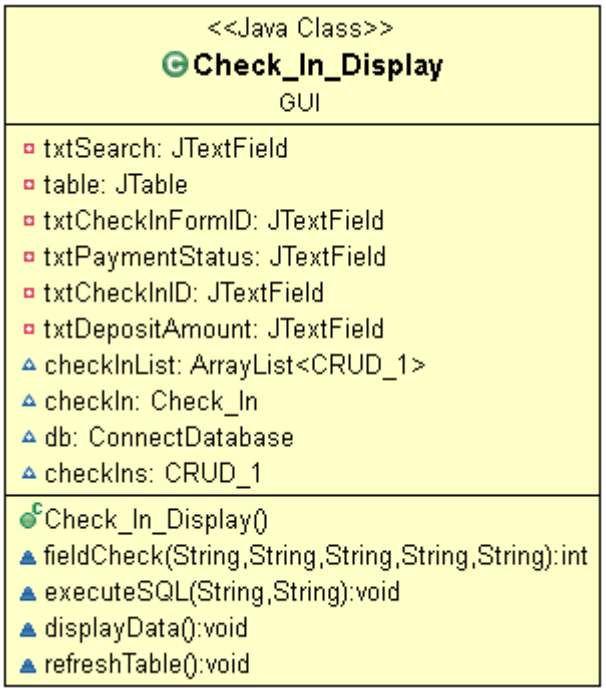

Figure 72:Class Diagram – Check in Display

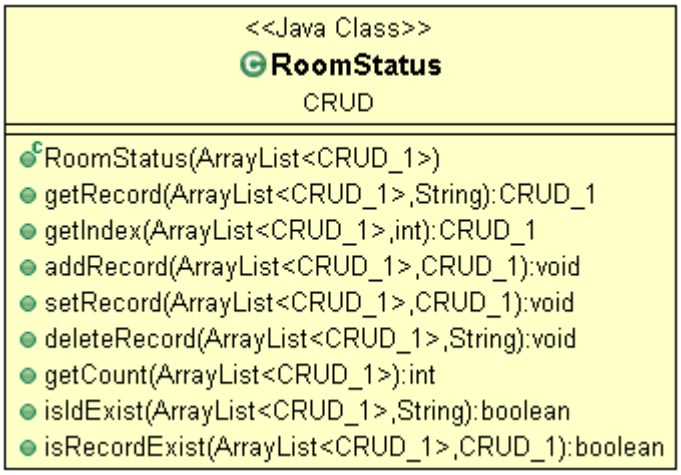

Figure 73:Class Diagram – Room Status

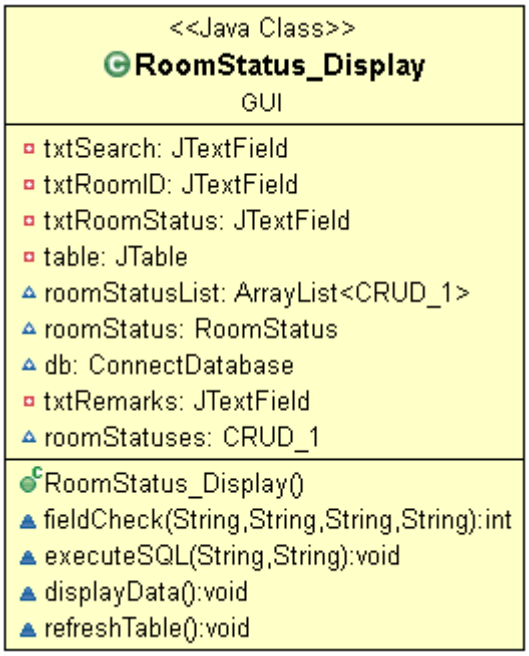

Figure 74:Class Diagram – Room Status Display

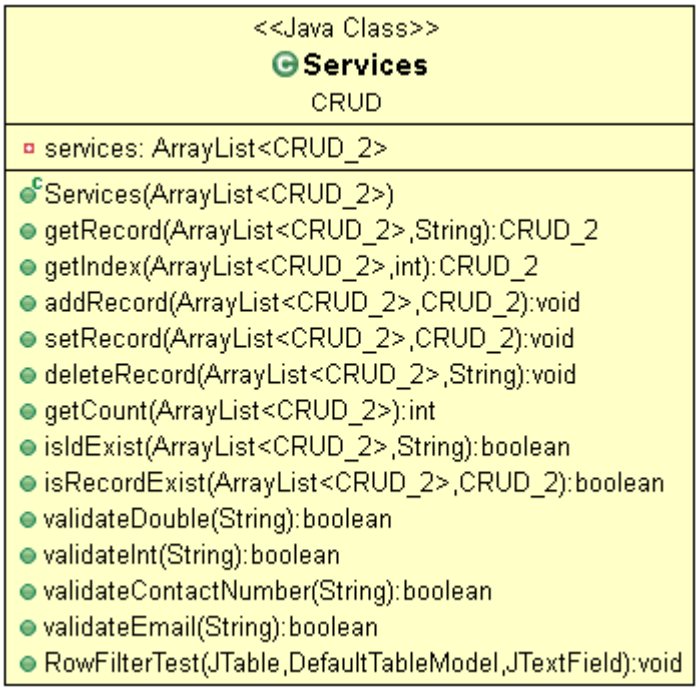

Figure 75:Class Diagram – Services

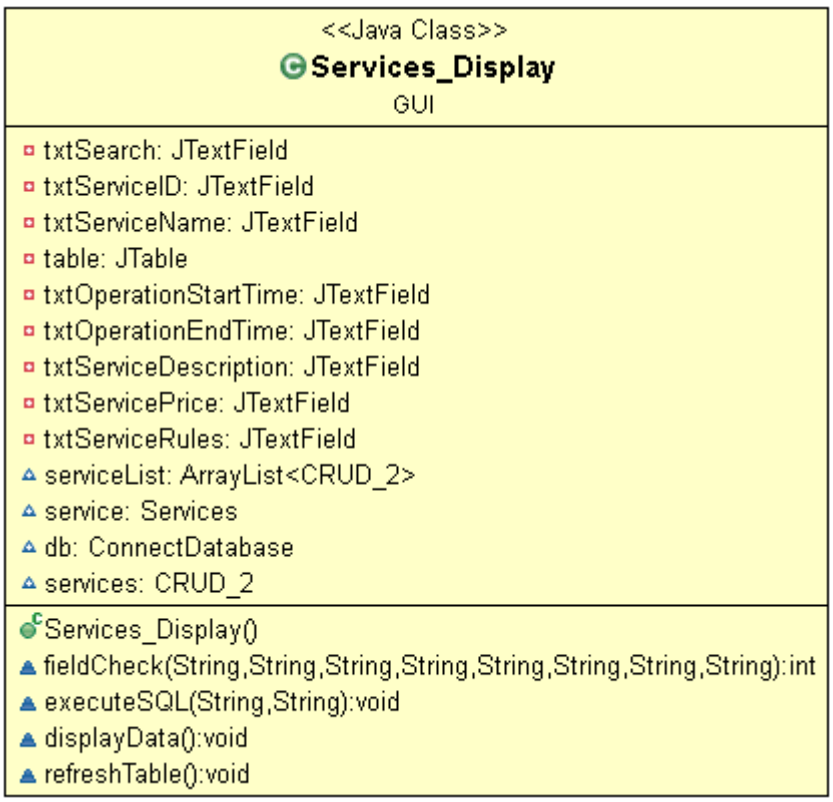

Figure 76:Class Diagram – Services Display

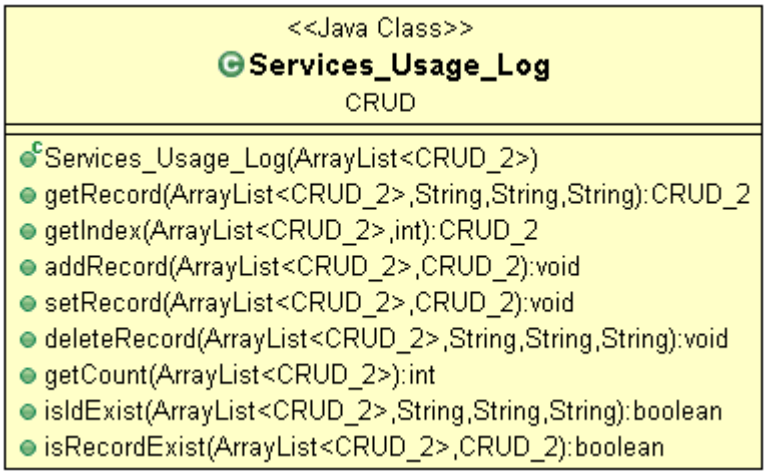

Figure 77:Class Diagram – Services Usage Log

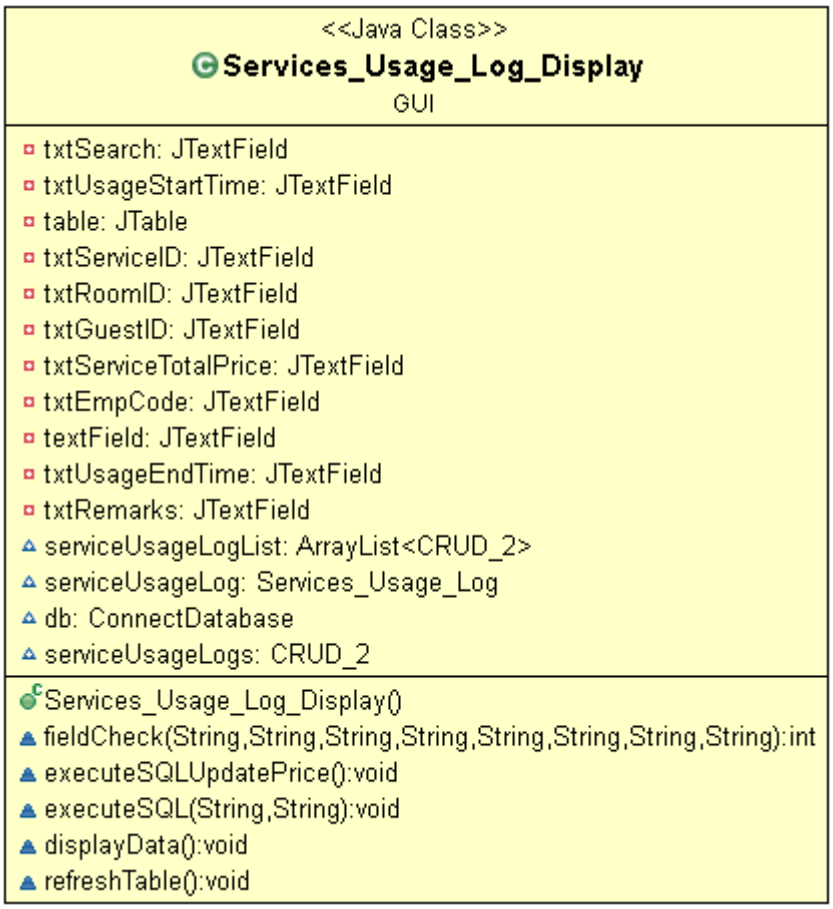

Figure 78:Class Diagram – Services Usage Log Display

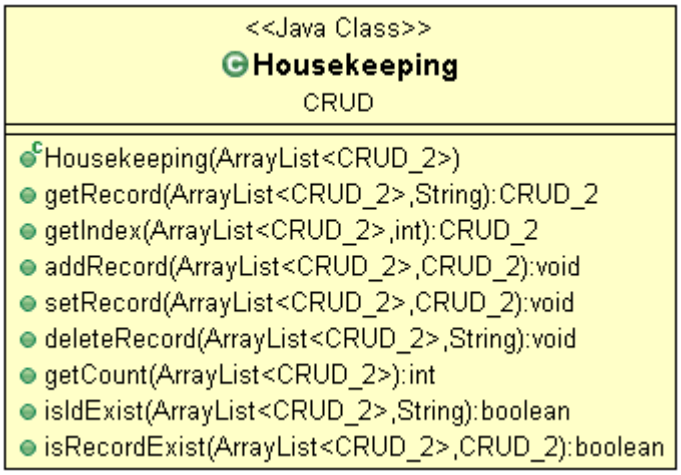

Figure 79:Class Diagram – Housekeeping

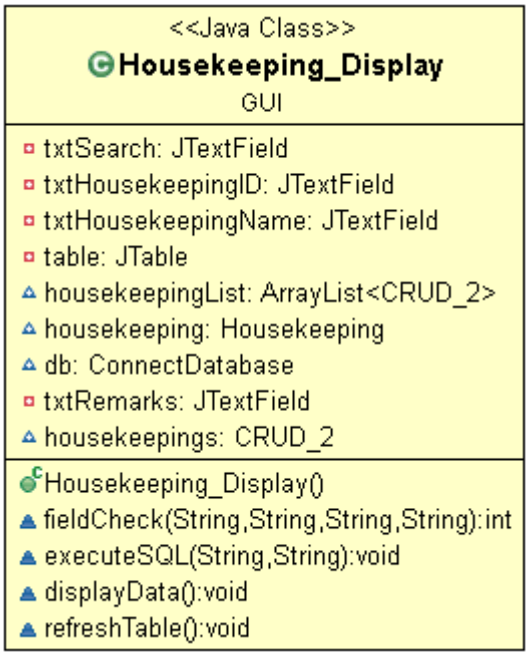

Figure 80:Class Diagram – Housekeeping Display

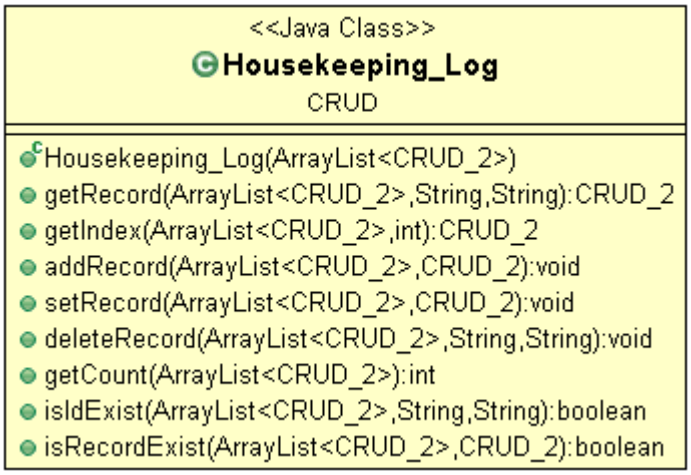

Figure 81:Class Diagram – Housekeeping Log

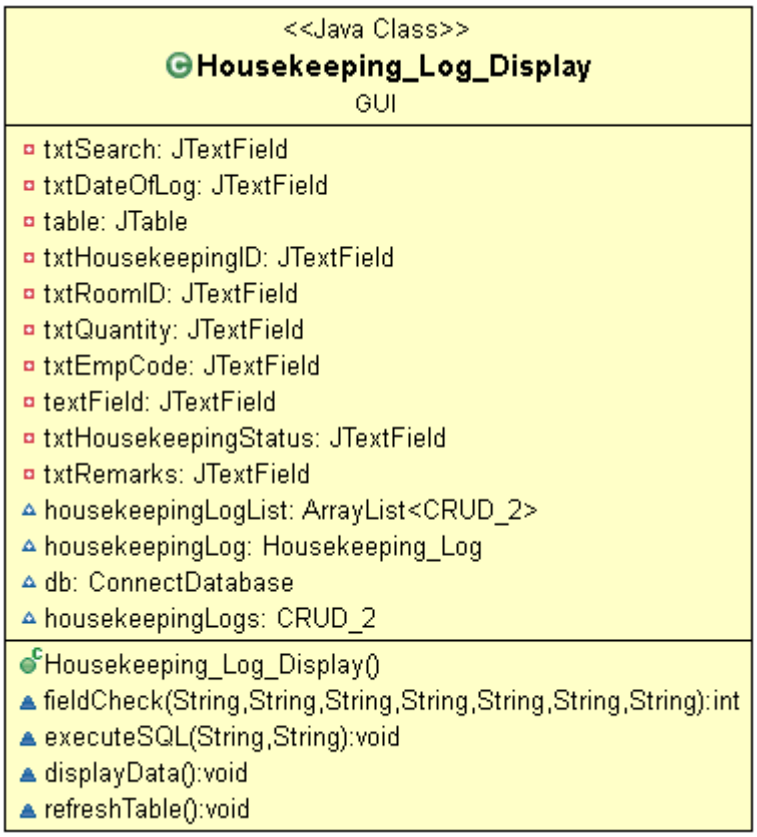

Figure 82:Class Diagram – Housekeeping Log Display

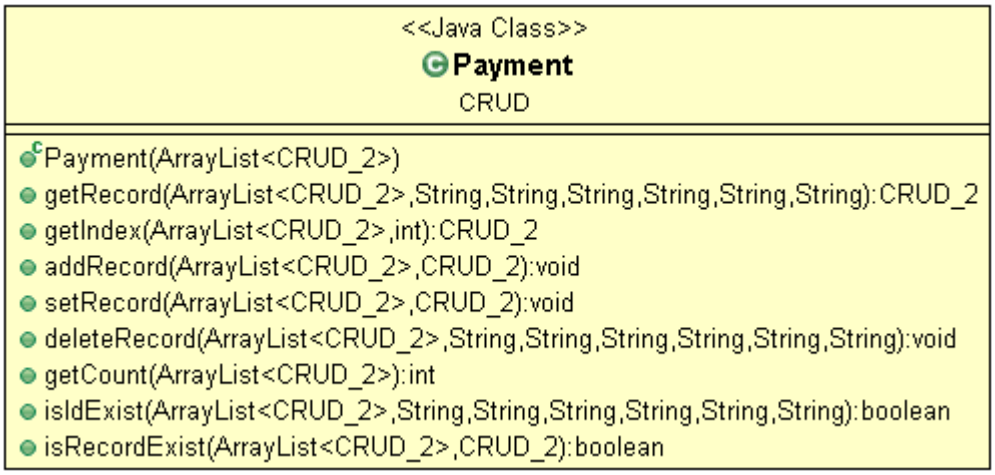

Figure 83:Class Diagram – Payment

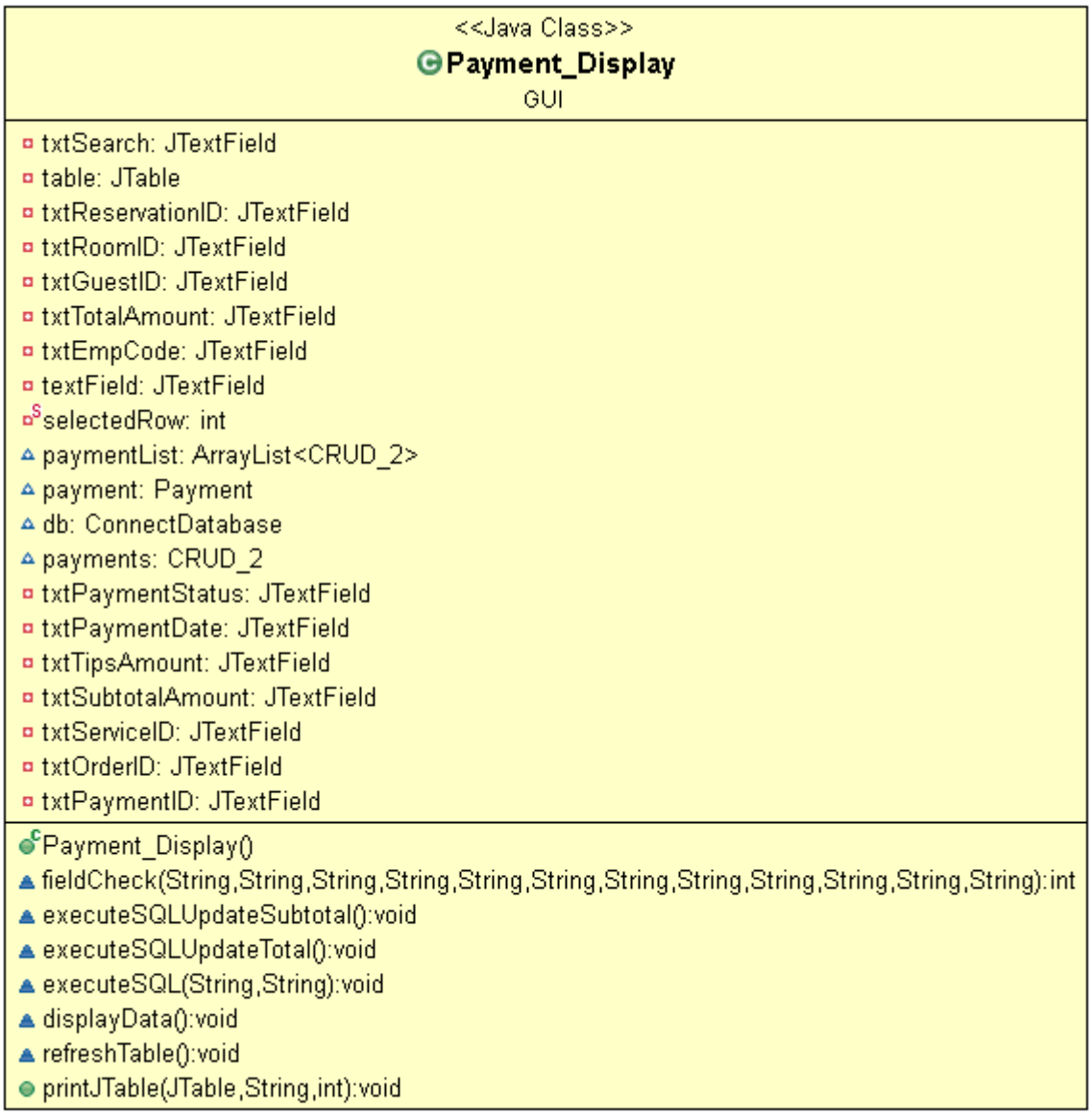

Figure 84:Class Diagram – Payment Display

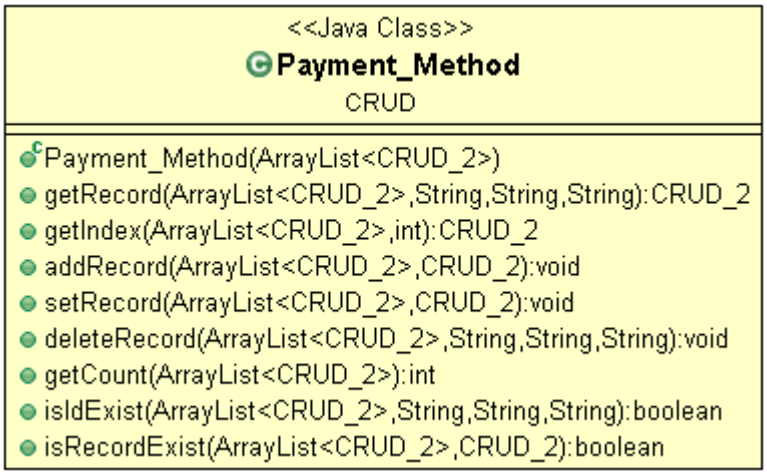

Figure 85:Class Diagram – Payment

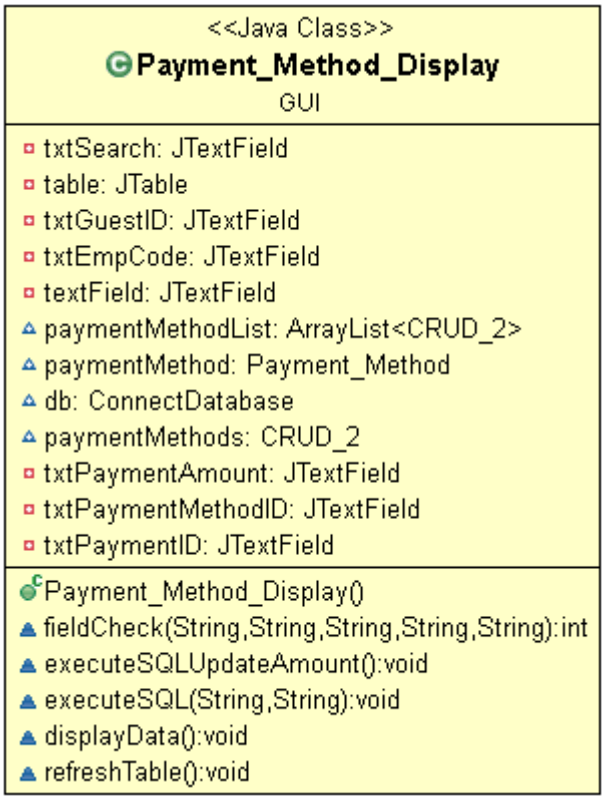

Figure 86:Class Diagram – Payment Display

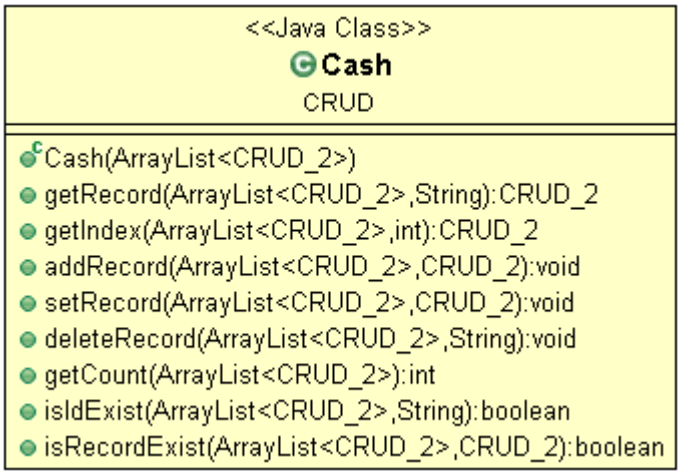

Figure 87:Class Diagram – Cash

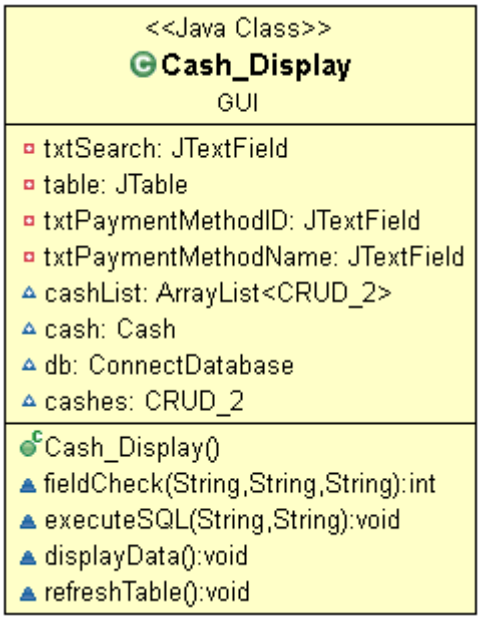

Figure 88:Class Diagram – Cash Display

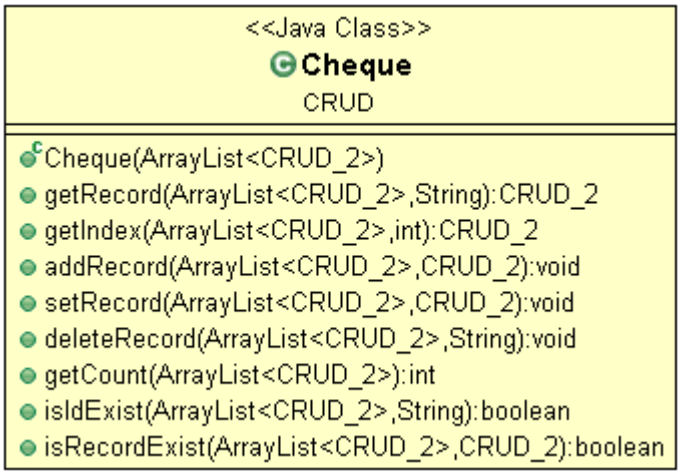

Figure 89:Class Diagram – Cheque

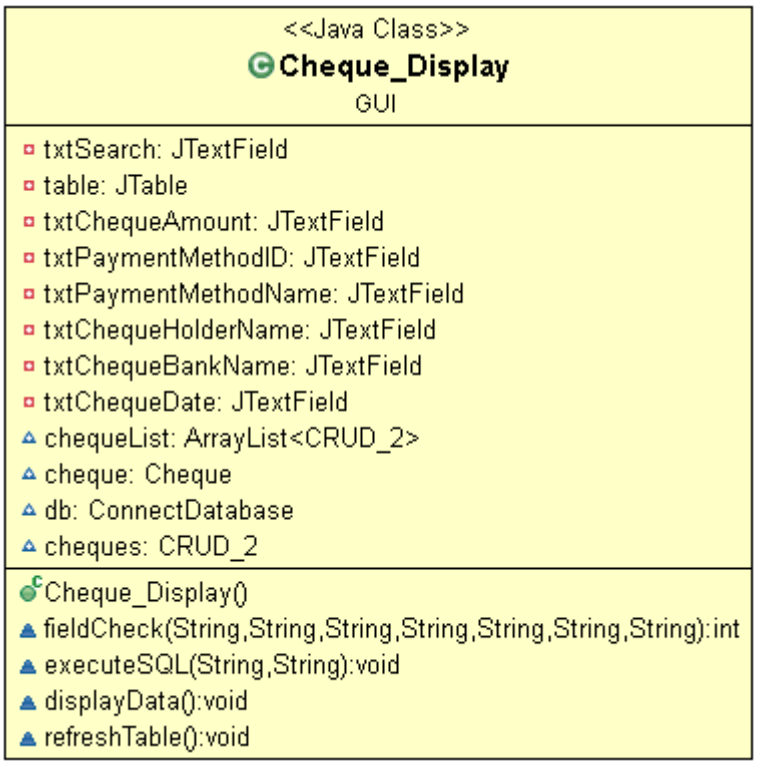

Figure 90:Class Diagram – Cheque Display

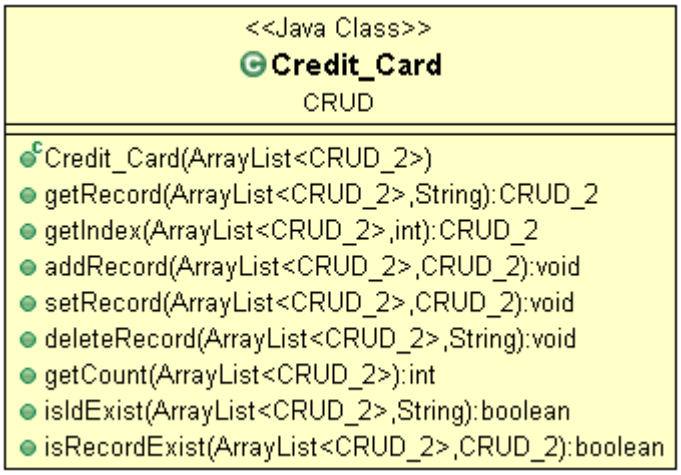

Figure 91:Class Diagram – Credit Card

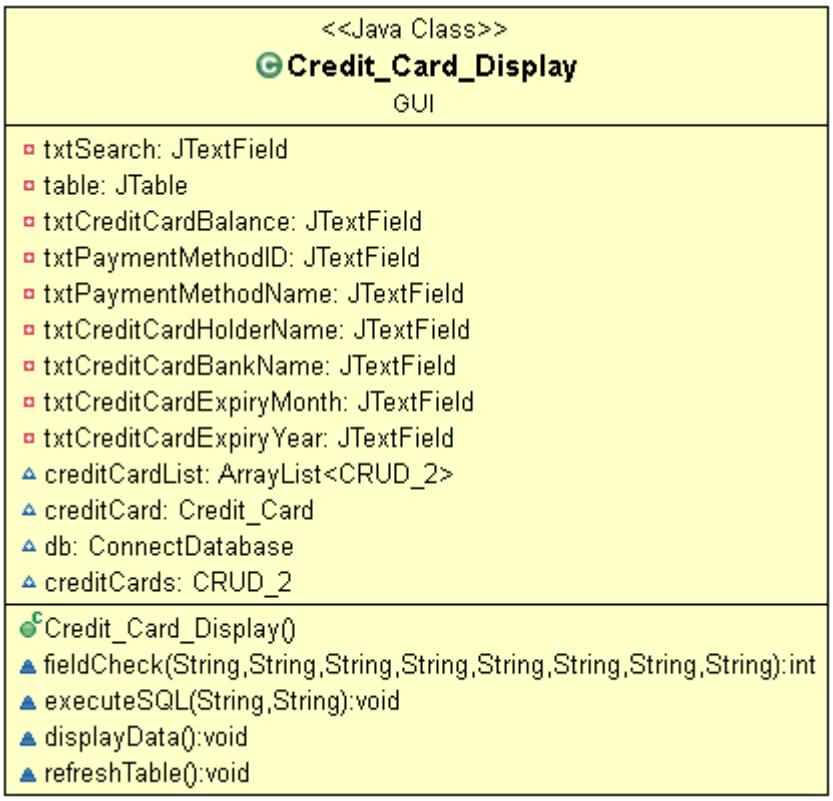

Figure 92:Class Diagram – Credit Card Display

Appendix B: Test Plan

B.1 Front-End Testing

Table 1: Test Plan for Front-End Testing

Test Case #	Test Case	Description	Steps	Expected Result	Actual Result	Status
1	Home page (home.vue)	Test the home page display	Browser link: "localhost:8080/"	Display the home page		Accepted
2	About us page (about.vue)	Test the about us page display	Browser link: "localhost:8080/about"	Display the about us page		Accepted
3	Room page (room.vue)	Test the room page display	Browser link: "localhost:8080/room"	Display the room page		Accepted
		Test the "Book" button	1) Browser link: "localhost:8080/room" 2) Click the "Book" button	Navigate to the booking page		
4	Cottage page (cottage.vue)	Test the cottage page display	Browser link: "localhost:8080/cottage"	Display the cottage page		Accepted

4	Cottage page (cottage.vue)	Test the “Book” button	1) Browser link: “localhost:8080/cottage” 2) Click the “Book” button	Navigate to the booking page	Accepted
5	Booking page (booking.vue)	Test the booking page display	Browser link: “localhost:8080/booking”	Display the booking page	Accepted
		Test the Email address field	No input	Display error message under the text field	
			Input “b0124@hotmail.com”, “b0124@mail.co”	Accept the input value	
		Test the Email address field	Input “b0124”, “b0124@”, “b0124@mail”	Display error message under the text field	Accepted
		Test the name field	No input	Display error message under the text field	Accepted
			Input name “Liew Soon Jing”	Accept the input value	
		Test the address field	No input	Display error message under the text field	Accepted
			Input address of “Petaling Jaya”	Accept the input value	

5	Booking page (booking.vue)	Test the phone number field	No input	Display error message under the text field	Accepted
			Input 012-2952660, 0122952660, +6012-2952660	Accept the input value	
		Test the gender radio button	No select	Display error message under the text field	Accepted
			Select male or female	Accept the input value	
		Test the hotel room drop down list	No select	Display error message under the text field	Accepted
			Select one option	Accept the input value	
		Test the room view drop down list	No select	Display error message under the text field	Accepted
			Select one option	Accept the input value	
		Test the date picker, check in and check out date	No select	Display error message under the text field	Accepted
			Select 2 date (check in date and check out date)	Accept the input value and the date will be shown in the field as selected.	
		Test the “Book” button	1) Ensure all input field above is filled up. 2) Click the “Book” button	Form data sent to the middleware to updated into database	Accepted
			Click the “Book” button	Display error to fill up the empty field in the form.	
		Test the “Reset: button	Click the “Reset” button	All the input field above is reset to blank or the default value.	Accepted

B.2 Back-End Testing

Table 2: Test Plan for Back-End Testing

Test Case #	Test Case	Description	Steps	Expected Result	Actual Result	Status
1	Select item in table	Able to display the row selected from the table to the text field to allow user to do modifications	Click on a row on the table in the system interface	Display the text fields accordingly from the table to the text field inputs		Accepted
2	Search bar	Search for data in any row of the system	Type in data present in the table	Display only the rows relevant to the search		Accepted
			Type in data non present in the table	Hide all rows until user deletes the search key		Accepted
3	Add a record with and without the same ID(s)	Used to check if the unique identifiers of the table data are redundant when users try to insert a new record	Enter the same ID existing in a row of record and click on the “Add” button	Display dialog box with suitable error message to inform on duplicate ID(s)		Accepted
			Enter unique ID(s) and click on the “Add” button	Accept the insert action		Accepted

4	Update a record with and without the exact same data in the record	Check if the user tries to update an existing record without changing any details in the record	Select a row of record from the interface and click on the “Update” button	Display dialog box with suitable error message to inform that the exact record already exists	Accepted
			Select a row of record from the interface, modify the record and click on the “Update” button	Accept the update action	Accepted
5	Submit Empty Fields	Check if user tries to add, update or delete any records with empty inputs	Leave all fields blank	Display dialog box with suitable error message to inform that all fields must be filled	Accepted
			Leave any fields blank	Display dialog box with suitable error message to inform that all fields must be filled	Accepted
			Fill in all fields	Accept the add/ update/ delete action	Accepted
6	Delete records with valid and invalid ID(s)	Check if user tries to delete any records with valid or invalid ID(s)	Enter valid ID	Accept the delete action	Accepted
			Enter invalid or nonexistent ID	Display dialog box with suitable error message to that no such ID is found in the records	Accepted
7	Integer field inputs	Checks user input when accepting integer values	Enter 1	Accept the input value	Accepted
			Enter 1.00 / abc/ abc1123/ @1	Display dialog box to inform user to enter a valid input	Accepted

8	Double field inputs	Checks user input when accepting double values	Enter 1/ 1.00	Accept the input value	Accepted
			Enter abc/ abc1123/ @1	Display dialog box to inform user to enter a valid input	Accepted
9	Email field inputs	Checks user input when accepting email values	Enter abc@abc.com	Accept the input value	Accepted
			Enter abc/ abc1123/ @1/abc.com /abc.ac.com	Display dialog box to inform user to enter a valid input	Accepted
10	Phone Number field inputs	Checks user input when accepting phone number values	Enter 1234567890 / 123-456-7890	Accept the input value	Accepted
			Enter abc/ abc1123/ @1/ +123-456-7890	Display dialog box to inform user to enter a valid input	Accepted

Group Report

ORIGINALITY REPORT

14%	3%	1%	13%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of Wales Institute, Cardiff Student Paper	3%
2	Submitted to De Montfort University Student Paper	2%
3	Submitted to University of Hertfordshire Student Paper	2%
4	Submitted to City University of Hong Kong Student Paper	1%
5	www.rug.nl Internet Source	1%
6	www.dsv.kth.se Internet Source	1%
7	Submitted to University of Derby Student Paper	1%
8	Submitted to First City University College Student Paper	1%
9	Submitted to Barnsley College, South Yorkshire	

	Student Paper	<1%
10	dmst.aueb.gr Internet Source	<1%
11	internationalscienceindex.org Internet Source	<1%
12	Submitted to UC, Boulder Student Paper	<1%
13	Submitted to University of Westminster Student Paper	<1%
14	Submitted to Central Queensland University Student Paper	<1%
15	Submitted to Swansea Metropolitan University Student Paper	<1%
16	Submitted to Colorado Technical University Online Student Paper	<1%
17	Submitted to Informatics Education Limited Student Paper	<1%
18	Submitted to Napier University Student Paper	<1%
19	Submitted to Botswana Accountancy College Student Paper	<1%
	www.halvorsen.blog	

20	Internet Source	<1%
21	Submitted to University of Salford Student Paper	<1%
22	Submitted to Austin Peay State Uniersity Student Paper	<1%
23	Submitted to American University of Beirut Student Paper	<1%
24	Submitted to Liverpool John Moores University Student Paper	<1%
25	Submitted to Queen Mary and Westfield College Student Paper	<1%
26	Submitted to University of Nevada, Las Vegas Student Paper	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		