INFORMATICS INSTITUTE OF TECHNOLOGY

UNIVERSITY OF WESTMINSTER

Informatics Institute of Technology
Department of Computing (B.Eng.) in Software Engineering

Module: 4COSC010C.2 Software Development II

Module Leader: Rajitha Viraj De Silva

Course Work 1

Date of Submission	:	2021/19/3
Student ID	:	2019784
Student UoW ID	:	w1809821
Name	:	Rukshan Fernando

Contents

Hotel version 1 Code(not a screenshot).

Hotel.java

```
package arrays;

import java.io.File;
import java.util.ArrayList;
import java.util.Formatter;
import java.util.Scanner;

public class Hotel {

    private static Room hotelRooms[]= new Room[8];
    private static Room queue[] = new Room[8]; // waiting list
    private static int start = 0; // start person's number
    private static int end = 0; // last person's number

    public static void main(String[] args) {
        // create menu
        // handle multiple inputs
        System.out.println("\n\t\t***** WELCOME TO HOTEL WARNAKULA *****");

        Scanner input = new Scanner(System.in);
        menu:// help to break while loop
        while (true) {
            System.out.println("\n\n*** Menu ***");
            System.out.println("\tA ---> Add Customer to a Room");
            System.out.println("\tV ---> View All Rooms");
            System.out.println("\tE ---> Display Empty Rooms");
            System.out.println("\tD ---> Delete Customer From Room");
            System.out.println("\tF ---> Find Room From Customer Name");
            System.out.println("\tS ---> Store Program Data");
            System.out.println("\tL ---> Load Program Data");
            System.out.println("\tO ---> View Guests Ordered Alphabetically by
Name");
            System.out.println("\tQ ---> Quit");
            System.out.print("\nPlease Enter the Option: ");
            // help to manage capital inputs and simple inputs
            String customerInput = input.next().toUpperCase();
            switch (customerInput) {
                case "A":
                    // access customer adding method
                    addsCustomerToRoom();
                    break;
                case "V":
                    // access customer viewing method
                    viewsAllRoom();
                    break;
                case "E":
                    //access customer show empty method
                    emptyRooms();
                    break;
                case "D":
                    //access customer deleting method
                    deleteCustomer();
                    break;
                case "F":
                    //access customer finding method
                    roomFromCustomerName();
                    break;
```

```
                case "S":
                    //access store all data into text file methods
                    storeProgram();
                    break;
                case "L":
                    //access load data and crete hotel rooms methods
                    loadProgram();
                    break;
                case "O":
                    //access customer sorting by alphabetically order methods
                    orderedAlphabetically();
                    break;
                case "Q":
                    //exit from program
                    System.out.println("\n\n\t\t***** Thank You! Have a Nice Day
*****");
                    break menu;
                default:
                    // show error when user enters wrong inputs
                    System.out.println("\n\tSorry, Invalid Input. Please Check
the Letter and Try Again.");
                    break;
            }
        }
    }

    public static void addsCustomerToRoom() {
        // get customer's data
        // add customer to hotel
        // show message if hotel is full
        // if hotel is full user program call waiting list method

        System.out.println("\n\n\t---Add Customer to a Room---");
        boolean available = false;

        // initialize all variables
        int roomNum = 0;
        String firstName = "";
        String surname = "";
        int cardNum = 0;
        int guest = 0;
        int avaCount = 0;

        for (Room room : hotelRooms){
            // check available room count
            if (room == null){
                avaCount++;// help to count available count
            }
            roomNum++;
        }

        if (avaCount > 0){
            // if hotel have available room this part runs
            Scanner scanner = new Scanner(System.in);
            while (true){


                System.out.print("\nEnter the Room Number (0-7) or 8 to stop: ");
                try {
                    // get room number from user
                    roomNum = Integer.parseInt(scanner.nextLine());
```

```
                    if (roomNum >= 0 && roomNum <= 8){
                        if(roomNum == 8){
                            // check users input
                            // if user need to exit this part runs
                            System.out.println("\n\tAdd Customer Option is
Stopped");
                            break;
                        }
                        else {
                            // if user need to add customer this part runs
                            if (hotelRooms[roomNum] == null){
                                // if user entered room is empty this art runs
                                available = true;// change available to true
                                System.out.println("\n\tRoom is Available");
                                break;
                            }
                            else {
                                // if room unavailable this message display
                                System.out.println("\n\tRoom " + roomNum + " is
Unavailable");
                            }
                        }
                    }
                    else {
                        // if user entered invalid number this message runs
                        System.out.println("\n\tSorry, Enter Valid Number and Try
Again.");
                    }
                }
                catch (NumberFormatException e) {
                    // if any error comes within this adding part this error
message display
                    System.out.println("\n\tSorry, Invalid Input. Try Again.");
                }
            }

            if (available){
                // if room available this part runs
                while (true) {
                    // get user's details
                    System.out.print("\nEnter Customer First Name: ");
                    firstName = scanner.nextLine().toUpperCase();

                    System.out.print("\nEnter Customer SurName: ");
                    surname = scanner.nextLine().toUpperCase();

                    try {
                        // check error in card number
                        System.out.print("\nEnter Customer Card Number: ");
                        cardNum = Integer.parseInt(scanner.nextLine());
                    }
                    catch (NumberFormatException e) {
                        // card number error message
                        System.out.println("\n\tSorry, Invalid Input. Try
Again.");
                    }

                    try {
                        // check error in guest count
                        System.out.print("\nEnter Customer's guest counts: ");
                        guest = Integer.parseInt(scanner.nextLine());
```

```
                    }
                    catch (NumberFormatException e) {
                        // check error in guest count
                        System.out.println("\n\tSorry, Invalid Input. Try
Again.");
                    }


                    if (!firstName.isEmpty()){
                        // check first name whether is empty or not
                        // if it not empty this parts runs
                        Room rsvRoom = new Room(roomNum, firstName, surname,
cardNum, guest);// add all data to room class
                        hotelRooms[roomNum] = rsvRoom;
                        System.out.println("\n\tRoom " + rsvRoom.getRoomNum() + "
Reserved by " + rsvRoom.getCusName() + " " + rsvRoom.getSecondName());
                        break;
                    }
                    else {
                        // if name is empty this one runs
                        System.out.println("\n\tSorry, Invalid Input. Try
Again.");
                    }
                }
            }
        }
        else {
            // if all rooms are full this part runs
            System.out.println("\n\tHotel is Fulled.");


        }
    }

    public static void viewsAllRoom () {
        // show all room details
        System.out.println("\n\n\t---View All Rooms---\n");

        int roomNum = 0;

        for (Room room : hotelRooms){
            if (room == null){
                // if any room is empty this one runs
                // show room as empty
                System.out.println("\tRoom " + roomNum + " is Empty");
            }
            else {
                // if any room is not empty this one runs
                // show room with user' details
                System.out.println("\tRoom " + room.getRoomNum() + "\tOccupied by
Customer: " + room.getCusName()+" "+room.getSecondName()+" [ Card
No."+room.getCardNum()+" | "+"Count of Guest: "+ room.getNumOfGuest()+" ]");
            }
            roomNum++;
        }
    }

    public static void emptyRooms () {
        // show all empty rooms
        System.out.println("\n\n\t---Display Empty Rooms---\n");
```

```
        int avaCount = 0;
        int roomNum = 0;

        for (Room room : hotelRooms){
            if (room == null){
                // if room is empty show it as empty
                System.out.println("\tRoom " + roomNum);
                avaCount++;
            }
            roomNum++;
        }

        if (avaCount == 0){
            // if no empty rooms this part runs
            System.out.println("\tNo Empty Rooms");
        }
    }

    private static void deleteCustomer () {
        // delete customer from hotel
        // go to get details from waiting list
        System.out.println("\n\n\t---Delete Customer From Room---");

        // initialize
        int roomNum = 0;
        int unAvaCount = 0;

        for (Room room : hotelRooms){
            // check all rooms with details
            if (room != null){
                unAvaCount++;
            }
        }

        if (unAvaCount > 0){
            // if hotel have rooms with uses this part runs
            Scanner scanner = new Scanner(System.in);
            while (true) {
                System.out.print("\nPlease Enter the room number (0-7) or 8 to
stop: ");
                try {
                    // get room number
                    roomNum = Integer.parseInt(scanner.nextLine());
                    if (roomNum >= 0 && roomNum <= 8) {
                        if (roomNum != 8) {
                            if (hotelRooms[roomNum] != null) {
                                // check room is available to delete or not
                                // if room has details delete all details
                                System.out.println("\n\tRoom " + roomNum + " is
Available Again");
                                hotelRooms[roomNum] = null;// deleting part

                            } else {
                                // if room have not data already this message
display
                                System.out.println("\n\tRoom " + roomNum + " is
Not Reserved");
                            }
                        }
                        else {
                            // if user need to exit from deleting part this one
```

```
runs
                            System.out.println("\n\tDelete Customer Option is
Stopped");
                        }
                        break;
                    } else {
                        // if user entered invalid input this part runs
                        System.out.println("\n\tSorry, Enter Valid Number and Try
Again.");
                    }
                } catch (NumberFormatException e) {
                    // if any error comes within deleting part this error message
displayed
                    System.out.println("\n\tSorry, Invalid Input. Try Again.");
                }
            }
        }
        else {
            //if all rooms are empty this part runs
            System.out.println("\n\tHotel is Empty");
        }
    }

    private static void roomFromCustomerName () {
        // find room from customer name
        // show customer details
        System.out.println("\n\n\t---Find Room From Customer Name---");

        // initialize
        String cusName = "";

        boolean roomsRsv = false;

        boolean foundRoom = false;

        for (Room room : hotelRooms){
            // check hotel have occupied room or not
            if (room != null){
                roomsRsv = true;
            }
        }

        if (roomsRsv){
            // if hotel have occupied rooms this part run
            Scanner scanner = new Scanner(System.in);
            while (true) {
                System.out.print("\nEnter Customer First Name or 8 to Stop: ");
                cusName = scanner.nextLine().toUpperCase();// get customer name

                if (!cusName.isEmpty()){

                    if (cusName.equals("8")){
                        // check user need to exit or not
                        // if user need to exit this part runs
                        System.out.println("\n\tFind Room Option is Stop");
                    }
                    else {
                        //if user need to delete this part runs
                        int roomNum = 0;

                        System.out.println("");
```

```
                        for (Room room : hotelRooms){
                            if (room != null) {
                                // find name
                                if (cusName.equals(room.getCusName())) {
                                    // if user found this part runs
                                    System.out.println("Room No." + roomNum+"
Customer Name :"+cusName+ " "+room.getSecondName());
                                    foundRoom = true;
                                }
                            }
                            roomNum++;
                        }

                        if (!foundRoom){
                            // if user not found this part runs
                            System.out.println("\n\tNo Customer Found Under This
Name " + cusName);
                        }
                    }
                    break;
                }
                else {
                    // if user entered invalid input this part runs
                    System.out.println("\n\tSorry, Invalid Input. Try Again");
                }
            }
        }
        else {
            // if all rooms are empty this part runs
            System.out.println("\n\tHotel is Empty");
        }
    }

    public static void storeProgram () {
        // get all data from array
        // store all data into txt file

        System.out.println("\n\n\t---Store Program Data---\n");
        try{
            Formatter StoreFile = new
Formatter("src/arrays/Hotel_Reservations.txt");// get file

            StoreFile.format("%s", "Hotel Warnakula");// write first line
            int roomNum = 0;
            for (Room room : hotelRooms) {
                // write other lines
                if (room == null) {
                    // if empty room found this one write
                    StoreFile.format("\nRoom %s\tCustomer Name: %s\tSurname:
%s\tCard: %s\tGuest: %s", roomNum, "Empty","Empty","Empty","Empty");
                }
                else {
                    // if found details this one writes
                    StoreFile.format("\nRoom %s\tCustomer Name: %s\tSurname:
%s\tCard: %s\tGuest:
%s",roomNum,room.getCusName(),room.getSecondName(),room.getCardNum(),room.getNumO
fGuest());
                }
                roomNum++;
            }
```

```
            StoreFile.close();// close object

            System.out.println("\n\tSuccessfully Stored to Hotel_Reservations
File");
        }
        catch(Exception e){
            // error happened within storing part this part runs
            System.out.println("\n\tStoring Error");
        }
    }

    public static void loadProgram () {
        // load data from file
        // fill all details into rooms

        System.out.println("\n\n\t---Load Program Data---\n");
        try{
            Scanner loadFile = new Scanner(new
File("src/arrays/Hotel_Reservations.txt"));// get file

            ArrayList<String> loadData = new ArrayList<>();// for get all data
            while(loadFile.hasNext()){
                // load all data from file
                String data = loadFile.next();
                loadData.add(data);
            }

            //initialize

            ArrayList<Integer> strRoomIndexList = new ArrayList<Integer>();
            ArrayList<Integer> strNameIndexList = new ArrayList<Integer>();
            ArrayList<Integer> strsurNameIndexList = new ArrayList<Integer>();
            ArrayList<Integer> strcardIndexList = new ArrayList<Integer>();
            ArrayList<Integer> strguestIndexList = new ArrayList<Integer>();


            int index = 0;
            for (String item : loadData){
                if (item.equals("Room")){
                    // get al room numbers indexes
                    strRoomIndexList.add(index);
                }
                else if (item.equals("Name:")){
                    // get all names indexes
                    strNameIndexList.add(index);
                }
                else if (item.equals("Surname:")){
                    // get all surname indexes
                    strsurNameIndexList.add(index);
                }
                else if (item.equals("Card:")){
                    // get all card details indexes
                    strcardIndexList.add(index);
                }
                else if (item.equals("Guest:")){
                    // get guest counts indexes
                    strguestIndexList.add(index);
                }
                index++;
            }
```

```
            strRoomIndexList.remove(0);// help to remove first line

            int count = 0;

            for (int strNameIndex : strNameIndexList){
                //initialize
                String firstName = "";
                String surname = "";
                String strCard = "";
                int card = 0;
                String strGuest = "";
                int guest = 0;

                for (int i = strNameIndex+1; i < strsurNameIndexList.get(count);
i++) {

                    firstName = firstName + loadData.get(i).toUpperCase() ;// get
first names
                }

                if (firstName.equals("EMPTY")) {
                    // if first name is equals to empty this part runs
                    hotelRooms[count] = null;// make room as empty
                }
                else {
                    for (int j = strsurNameIndexList.get(count) + 1; j <
strcardIndexList.get(count); j++) {
                        surname = surname + loadData.get(j).toUpperCase();// get
surname
                    }

                    for (int k = strcardIndexList.get(count) + 1; k <
strguestIndexList.get(count); k++) {
                        strCard = strCard + loadData.get(k).toUpperCase();// get
card
                    }

                    try{
                        // convert to integer
                        card = Integer.parseInt(strCard);
                    }
                    catch (NumberFormatException e){

                    }

                    if (count == strRoomIndexList.size()) {
                        for (int l = strguestIndexList.get(count) + 1; l <
loadData.size(); l++) {
                            strGuest = strGuest +
loadData.get(l).toUpperCase();// get guest count data as string
                        }
                    }
                    else {
                        for (int m = strguestIndexList.get(count) + 1; m <
strRoomIndexList.get(count); m++) {
                            strGuest = strGuest +
loadData.get(m).toUpperCase();// get guest count data as string
                        }
                    }

                    try{
```

```
                        guest = Integer.parseInt(strGuest);// convert  as integer
                    }
                    catch (NumberFormatException e){

                    }

                    Room room = new Room(count, firstName, surname, card,
guest);// add all data to room
                    hotelRooms[count] = room;// adding part
                }
                count++;
            }

            loadFile.close();// close object

            System.out.println("\n\tSuccessfully Load from Hotel_Reservations
File");
        }
        catch(Exception e){
            // if error comes this part runs
            System.out.println("\tLoading Error");
        }
    }

    private static void orderedAlphabetically (){

        // get details and sort all data by customer's name


        ArrayList<String> sortingList = new ArrayList<String>();

        for (Room itemx : hotelRooms){
            if (itemx != null){
                // get all room details
                // added data to list
                sortingList.add(itemx.getCusName().replace("_"," ")+"
"+itemx.getSecondName().replace("_"," "));
            }
        }


        int arrayListSize = sortingList.size();// get array length
        for (int traversal = 1; traversal <= arrayListSize; traversal++) {
            for (int leftIndex = 0; leftIndex < arrayListSize - 1; leftIndex++) {
                int rightIndex = leftIndex + 1;
                if
(sortingList.get(rightIndex).compareTo(sortingList.get(leftIndex)) < 0) {
                    // if right index is grater than left index this part runs

                    // shifting part
                    String rightStringInSortingList =
sortingList.get(rightIndex);

                    sortingList.remove(rightIndex);

                    sortingList.add(leftIndex, rightStringInSortingList);
                }
            }
            for (int backwardsRightIndex = arrayListSize - 1; backwardsRightIndex
> 0; backwardsRightIndex--) {
                int backwardsLeftIndex = backwardsRightIndex - 1;
```

```
                if
(sortingList.get(backwardsRightIndex).compareTo(sortingList.get(backwardsRightInd
ex - 1)) < 0) {
                    // check two data and find biggest name

                    //shifting part
                    String backwardsRightStringInSortingList =
sortingList.get(backwardsRightIndex);

                    sortingList.remove(backwardsRightIndex);

                    sortingList.add(backwardsLeftIndex,
backwardsRightStringInSortingList);
                }
            }
        }
        System.out.println("\n\n\t---Alphabetically Name Order---\n");
        for (int index = 0; index < arrayListSize; index++) {
            // show all sorted names
            System.out.println(sortingList.get(index));
        }


    }


    public static class Room {
        //subclass for handle user's data
        //initilize
        private int roomNum;
        private String firstName;
        private String secondName;


        private int cardNum;
        private int numOfGuest;

        public Room(int roomNum, String firstName, String secondName, int
cardNum, int numOfGuest) {
            //cunstractor
            this.roomNum = roomNum;
            this.firstName = firstName;
            this.secondName = secondName;
            this.cardNum = cardNum;
            this.numOfGuest = numOfGuest;
        }

        public int getRoomNum(){
            return roomNum;
        }//getter for roomNum

        public String getSecondName() {
            return secondName;
        }//getter for secondNum

        public int getCardNum() {
            return cardNum;
        }//getter for cardNum

        public int getNumOfGuest() {
            return numOfGuest;
```

```
        }//getter for numberOfGuest

        public String getCusName(){
            return firstName;
        }//getter for cusName
    }

}
```

Hotel class version Code(not a screenshot).

Hotel.java

```
import java.io.File;
import java.util.ArrayList;
import java.util.Formatter;
import java.util.Scanner;

public class Hotel {

    private static Room hotelRooms[]= new Room[8];
    private static Room queue[] = new Room[8]; // waiting list
    private static int start = 0; // start person's number
    private static int end = 0; // last person's number

    public static void main(String[] args) {
        // create menu
        // handle multiple inputs
        System.out.println("\n\t\t***** WELCOME TO HOTEL WARNAKULA *****");

        Scanner input = new Scanner(System.in);
        menu:// help to break while loop
        while (true) {
            System.out.println("\n\n*** Menu ***");
            System.out.println("\tA ---> Add Customer to a Room");
            System.out.println("\tV ---> View All Rooms");
            System.out.println("\tE ---> Display Empty Rooms");
            System.out.println("\tD ---> Delete Customer From Room");
            System.out.println("\tF ---> Find Room From Customer Name");
            System.out.println("\tS ---> Store Program Data");
            System.out.println("\tL ---> Load Program Data");
            System.out.println("\tO ---> View Guests Ordered Alphabetically by
Name");
            System.out.println("\tQ ---> Quit");
            System.out.print("\nPlease Enter the Option: ");
            // help to manage capital inputs and simple inputs
            String customerInput = input.next().toUpperCase();
            switch (customerInput) {
                case "A":
                    // access customer adding method
                    addsCustomerToRoom();
                    break;
                case "V":
                    // access customer viewing method
                    viewsAllRoom();
                    break;
                case "E":
                    //access customer show empty method
                    emptyRooms();
                    break;
                case "D":
                    //access customer deleting method
                    deleteCustomer();
                    break;
                case "F":
                    //access customer finding method
                    roomFromCustomerName();
                    break;
                case "S":
                    //access store all data into text file methods
```

```
                    storeProgram();
                    break;
                case "L":
                    //access load data and crete hotel rooms methods
                    loadProgram();
                    break;
                case "O":
                    //access customer sorting by alphabetically order methods
                    orderedAlphabetically();
                    break;
                case "Q":
                    //exit from program
                    System.out.println("\n\n\t\t***** Thank You! Have a Nice Day
*****");
                    break menu;
                default:
                    // show error when user enters wrong inputs
                    System.out.println("\n\tSorry, Invalid Input. Please Check
the Letter and Try Again.");
                    break;
            }
        }
    }

    public static void addsCustomerToRoom() {
        // get customer's data
        // add customer to hotel
        // show message if hotel is full
        // if hotel is full user program call waiting list method

        System.out.println("\n\n\t---Add Customer to a Room---");
        boolean available = false;

        // initialize all variables
        int roomNum = 0;
        String firstName = "";
        String surname = "";
        int cardNum = 0;
        int guest = 0;
        int avaCount = 0;

        for (Room room : hotelRooms){
            // check available room count
            if (room == null){
                avaCount++;// help to count available count
            }
            roomNum++;
        }

        if (avaCount > 0){
            // if hotel have available room this part runs
            Scanner scanner = new Scanner(System.in);
            while (true){


                System.out.print("\nEnter the Room Number (0-7) or 8 to stop: ");
                try {
                    // get room number from user
                    roomNum = Integer.parseInt(scanner.nextLine());
                    if (roomNum >= 0 && roomNum <= 8){
                        if(roomNum == 8){
```

```
                            // check users input
                            // if user need to exit this part runs
                            System.out.println("\n\tAdd Customer Option is
Stopped");
                            break;
                        }
                        else {
                            // if user need to add customer this part runs
                            if (hotelRooms[roomNum] == null){
                                // if user entered room is empty this art runs
                                available = true;// change available to true
                                System.out.println("\n\tRoom is Available");
                                break;
                            }
                            else {
                                // if room unavailable this message display
                                System.out.println("\n\tRoom " + roomNum + " is
Unavailable");
                            }
                        }
                    }
                    else {
                        // if user entered invalid number this message runs
                        System.out.println("\n\tSorry, Enter Valid Number and Try
Again.");
                    }
                }
                catch (NumberFormatException e) {
                    // if any error comes within this adding part this error
message display
                    System.out.println("\n\tSorry, Invalid Input. Try Again.");
                }
            }

            if (available){
                // if room available this part runs
                while (true) {
                    // get user's details
                    System.out.print("\nEnter Customer First Name: ");
                    firstName = scanner.nextLine().toUpperCase();

                    System.out.print("\nEnter Customer SurName: ");
                    surname = scanner.nextLine().toUpperCase();

                    try {
                        // check error in card number
                        System.out.print("\nEnter Customer Card Number: ");
                        cardNum = Integer.parseInt(scanner.nextLine());
                    }
                    catch (NumberFormatException e) {
                        // card number error message
                        System.out.println("\n\tSorry, Invalid Input. Try
Again.");
                    }

                    try {
                        // check error in guest count
                        System.out.print("\nEnter Customer's guest counts: ");
                        guest = Integer.parseInt(scanner.nextLine());
                    }
                    catch (NumberFormatException e) {
```

```
                        // check error in guest count
                        System.out.println("\n\tSorry, Invalid Input. Try
Again.");
                    }


                    if (!firstName.isEmpty()){
                        // check first name whether is empty or not
                        // if it not empty this parts runs
                        Room rsvRoom = new Room(roomNum, firstName, surname,
cardNum, guest);// add all data to room class
                        hotelRooms[roomNum] = rsvRoom;
                        System.out.println("\n\tRoom " + rsvRoom.getRoomNum() + "
Reserved by " + rsvRoom.getCusName() +" " + rsvRoom.getSecondName());
                        break;
                    }
                    else {
                        // if name is empty this one runs
                        System.out.println("\n\tSorry, Invalid Input. Try
Again.");
                    }
                }
            }
        }
        else {
            // if all rooms are full this part runs
            System.out.println("\n\tHotel is Fulled.");
            System.out.println("\n\tAdd to Waiting List");
            createQueue();// if all rooms are full this method runs to call
waiting room method

        }
    }

    public static void viewsAllRoom () {
        // show all room details
        System.out.println("\n\n\t---View All Rooms---\n");

        int roomNum = 0;

        for (Room room : hotelRooms){
            if (room == null){
                // if any room is empty this one runs
                // show room as empty
                System.out.println("\tRoom " + roomNum + " is Empty");
            }
            else {
                // if any room is not empty this one runs
                // show room with user' details
                System.out.println("\tRoom " + room.getRoomNum() + "\tOccupied by
Customer: " + room.getCusName()+" "+room.getSecondName()+" [ Card
No."+room.getCardNum()+" | "+"Count of Guest: "+ room.getNumOfGuest()+" ]");
            }
            roomNum++;
        }
    }

    public static void emptyRooms () {
        // show all empty rooms
        System.out.println("\n\n\t---Display Empty Rooms---\n");
```

```
        int avaCount = 0;
        int roomNum = 0;

        for (Room room : hotelRooms){
            if (room == null){
                // if room is empty show it as empty
                System.out.println("\tRoom " + roomNum);
                avaCount++;
            }
            roomNum++;
        }

        if (avaCount == 0){
            // if no empty rooms this part runs
            System.out.println("\tNo Empty Rooms");
        }
    }

    private static void deleteCustomer () {
        // delete customer from hotel
        // go to get details from waiting list
        System.out.println("\n\n\t---Delete Customer From Room---");

        // initialize
        int roomNum = 0;
        int unAvaCount = 0;

        for (Room room : hotelRooms){
            // check all rooms with details
            if (room != null){
                unAvaCount++;
            }
        }

        if (unAvaCount > 0){
            // if hotel have rooms with uses this part runs
            Scanner scanner = new Scanner(System.in);
            while (true) {
                System.out.print("\nPlease Enter the room number (0-7) or 8 to
stop: ");
                try {
                    // get room number
                    roomNum = Integer.parseInt(scanner.nextLine());
                    if (roomNum >= 0 && roomNum <= 8) {
                        if (roomNum != 8) {
                            if (hotelRooms[roomNum] != null) {
                                // check room is available to delete or not
                                // if room has details delete all details
                                System.out.println("\n\tRoom " + roomNum + " is
Available Again");
                                hotelRooms[roomNum] = null;// deleting part
                                getCusFromQueue(roomNum);// if customer deleted
tis one runs
                            } else {
                                // if room have not data already this message
display
                                System.out.println("\n\tRoom " + roomNum + " is
Not Reserved");
                            }
                        }
                        else {
```

```
                            // if user need to exit from deleting part this one
runs
                            System.out.println("\n\tDelete Customer Option is
Stopped");
                        }
                        break;
                    } else {
                        // if user entered invalid input this part runs
                        System.out.println("\n\tSorry, Enter Valid Number and Try
Again.");
                    }
                } catch (NumberFormatException e) {
                    // if any error comes within deleting part this error message
displayed
                    System.out.println("\n\tSorry, Invalid Input. Try Again.");
                }
            }
        }
        else {
            //if all rooms are empty this part runs
            System.out.println("\n\tHotel is Empty");
        }
    }

    private static void roomFromCustomerName () {
        // find room from customer name
        // show customer details
        System.out.println("\n\n\t---Find Room From Customer Name---");

        // initialize
        String cusName = "";

        boolean roomsRsv = false;

        boolean foundRoom = false;

        for (Room room : hotelRooms){
            // check hotel have occupied room or not
            if (room != null){
                roomsRsv = true;
            }
        }

        if (roomsRsv){
            // if hotel have occupied rooms this part run
            Scanner scanner = new Scanner(System.in);
            while (true) {
                System.out.print("\nEnter Customer First Name or 8 to Stop: ");
                cusName = scanner.nextLine().toUpperCase();// get customer name

                if (!cusName.isEmpty()){

                    if (cusName.equals("8")){
                        // check user need to exit or not
                        // if user need to exit this part runs
                        System.out.println("\n\tFind Room Option is Stop");
                    }
                    else {
                        //if user need to delete this part runs
                        int roomNum = 0;
```

```
                        System.out.println("");
                        for (Room room : hotelRooms){
                            if (room != null) {
                                // find name
                                if (cusName.equals(room.getCusName())) {
                                    // if user found this part runs
                                    System.out.println("Room No." + roomNum+"
Customer Name :"+cusName+ " "+room.getSecondName());
                                    foundRoom = true;
                                }
                            }
                            roomNum++;
                        }

                        if (!foundRoom){
                            // if user not found this part runs
                            System.out.println("\n\tNo Customer Found Under This
Name " + cusName);
                        }
                    }
                    break;
                }
                else {
                    // if user entered invalid input this part runs
                    System.out.println("\n\tSorry, Invalid Input. Try Again");
                }
            }
        }
        else {
            // if all rooms are empty this part runs
            System.out.println("\n\tHotel is Empty");
        }
    }

    public static void storeProgram () {
        // get all data from array
        // store all data into txt file

        System.out.println("\n\n\t---Store Program Data---\n");
        try{
            Formatter StoreFile = new Formatter("src/Hotel_Reservations.txt");//
get file

            StoreFile.format("%s", "Hotel Warnakula");// write first line
            int roomNum = 0;
            for (Room room : hotelRooms) {
                // write other lines
                if (room == null) {
                    // if empty room found this one write
                    StoreFile.format("\nRoom %s\tCustomer Name: %s\tSurname:
%s\tCard: %s\tGuest: %s", roomNum, "Empty","Empty","Empty","Empty");
                }
                else {
                    // if found details this one writes
                    StoreFile.format("\nRoom %s\tCustomer Name: %s\tSurname:
%s\tCard: %s\tGuest:
%s",roomNum,room.getCusName(),room.getSecondName(),room.getCardNum(),room.getNumO
fGuest());
                }
                roomNum++;
            }
```

```
            StoreFile.close();// close object

            System.out.println("\n\tSuccessfully Stored to Hotel_Reservations
File");
        }
        catch(Exception e){
            // error happened within storing part this part runs
            System.out.println("\n\tStoring Error");
        }
    }

    public static void loadProgram () {
        // load data from file
        // fill all details into rooms

        System.out.println("\n\n\t---Load Program Data---\n");
        try{
            Scanner loadFile = new Scanner(new
File("src/Hotel_Reservations.txt"));// get file

            ArrayList<String> loadData = new ArrayList<>();// for get all data
            while(loadFile.hasNext()){
                // load all data from file
                String data = loadFile.next();
                loadData.add(data);
            }

            //initialize

            ArrayList<Integer> strRoomIndexList = new ArrayList<Integer>();
            ArrayList<Integer> strNameIndexList = new ArrayList<Integer>();
            ArrayList<Integer> strsurNameIndexList = new ArrayList<Integer>();
            ArrayList<Integer> strcardIndexList = new ArrayList<Integer>();
            ArrayList<Integer> strguestIndexList = new ArrayList<Integer>();


            int index = 0;
            for (String item : loadData){
                if (item.equals("Room")){
                    // get al room numbers indexes
                    strRoomIndexList.add(index);
                }
                else if (item.equals("Name:")){
                    // get all names indexes
                    strNameIndexList.add(index);
                }
                else if (item.equals("Surname:")){
                    // get all surname indexes
                    strsurNameIndexList.add(index);
                }
                else if (item.equals("Card:")){
                    // get all card details indexes
                    strcardIndexList.add(index);
                }
                else if (item.equals("Guest:")){
                    // get guest counts indexes
                    strguestIndexList.add(index);
                }
                index++;
            }
```

```
            strRoomIndexList.remove(0);// help to remove first line

            int count = 0;

            for (int strNameIndex : strNameIndexList){
                //initialize
                String firstName = "";
                String surname = "";
                String strCard = "";
                int card = 0;
                String strGuest = "";
                int guest = 0;

                for (int i = strNameIndex+1; i < strsurNameIndexList.get(count);
i++) {

                    firstName = firstName + loadData.get(i).toUpperCase() ;// get
first names
                }

                if (firstName.equals("EMPTY")) {
                    // if first name is equals to empty this part runs
                    hotelRooms[count] = null;// make room as empty
                }
                else {
                    for (int j = strsurNameIndexList.get(count) + 1; j <
strcardIndexList.get(count); j++) {
                        surname = surname + loadData.get(j).toUpperCase();// get
surname
                    }

                    for (int k = strcardIndexList.get(count) + 1; k <
strguestIndexList.get(count); k++) {
                        strCard = strCard + loadData.get(k).toUpperCase();// get
card
                    }

                    try{
                        // convert to integer
                        card = Integer.parseInt(strCard);
                    }
                    catch (NumberFormatException e){

                    }

                    if (count == strRoomIndexList.size()) {
                        for (int l = strguestIndexList.get(count) + 1; l <
loadData.size(); l++) {
                            strGuest = strGuest +
loadData.get(l).toUpperCase();// get guest count data as string
                        }
                    }
                    else {
                        for (int m = strguestIndexList.get(count) + 1; m <
strRoomIndexList.get(count); m++) {
                            strGuest = strGuest +
loadData.get(m).toUpperCase();// get guest count data as string
                        }
                    }
```

```
                    try{
                        guest = Integer.parseInt(strGuest);// convert  as integer
                    }
                    catch (NumberFormatException e){

                    }

                    Room room = new Room(count, firstName, surname, card,
guest);// add all data to room
                    hotelRooms[count] = room;// adding part
                }
                count++;
            }

            loadFile.close();// close object

            System.out.println("\n\tSuccessfully Load from Hotel_Reservations
File");
        }
        catch(Exception e){
            // if error comes this part runs
            System.out.println("\tLoading Error");
        }
    }

    private static void orderedAlphabetically (){

        // get details and sort all data by customer's name


        ArrayList<String> sortingList = new ArrayList<String>();

        for (Room itemx : hotelRooms){
            if (itemx != null){
                // get all room details
                // added data to list
                sortingList.add(itemx.getCusName().replace("_"," ")+"
"+itemx.getSecondName().replace("_"," "));
            }
        }


        int arrayListSize = sortingList.size();// get array length
        for (int traversal = 1; traversal <= arrayListSize; traversal++) {
            for (int leftIndex = 0; leftIndex < arrayListSize - 1; leftIndex++) {
                int rightIndex = leftIndex + 1;
                if
(sortingList.get(rightIndex).compareTo(sortingList.get(leftIndex)) < 0) {
                    // if right index is grater than left index this part runs

                    // shifting part
                    String rightStringInSortingList =
sortingList.get(rightIndex);

                    sortingList.remove(rightIndex);

                    sortingList.add(leftIndex, rightStringInSortingList);
                }
            }
            for (int backwardsRightIndex = arrayListSize - 1; backwardsRightIndex
> 0; backwardsRightIndex--) {
```

```
                int backwardsLeftIndex = backwardsRightIndex - 1;
                if
(sortingList.get(backwardsRightIndex).compareTo(sortingList.get(backwardsRightInd
ex - 1)) < 0) {
                    // check two data and find biggest name

                    //shifting part
                    String backwardsRightStringInSortingList =
sortingList.get(backwardsRightIndex);

                    sortingList.remove(backwardsRightIndex);

                    sortingList.add(backwardsLeftIndex,
backwardsRightStringInSortingList);
                }
            }
        }
        System.out.println("\n\n\t---Alphabetically Name Order---\n");
        for (int index = 0; index < arrayListSize; index++) {
            // show all sorted names
            System.out.println(sortingList.get(index));
        }


    }

    private static void getCusFromQueue(int roomNum){

        // add customer to hotel
        // remove customer from waiting list
        // change starting person's number

        boolean x =false;// check waiting list is empty or not

        for (int i = 0; i< queue.length; i++){

            if (!(queue[i] == null)){

                x = true;// if one value hav e in waiting list this one runs
                break;
            }

        }

        if (x){
            // if value in waiting list this one runs

            if (checkStart()) {
                // if start index in waiting lis has value this one runs

                //get data from queue
                Room rsvRoom = queue[start];
                //add data to hotel
                hotelRooms[roomNum] = rsvRoom;
                hotelRooms[roomNum].setRoomNum(roomNum);
                System.out.println("\n\tCustomer: " + rsvRoom.getCusName() +" "+
rsvRoom.getSecondName() +" Successfully Add to Room No."+ roomNum +" From
Queue");

                // remove person from queue
                queue[start] = null;
```

```
                //change starting value
                start++;

            }

        }

    }
    private static boolean checkStart(){

        // check starting value is grater equal waiting list length

        boolean x = false;

        if(start == queue.length){
            // check stating number and change it to 0 if it grater equal waiting
list length
            start = 0;
        }

        if (!(queue[start]== null)){
            // if starting index in waiting list has value this one runs
            x = true;

        }

        return x;

    }

    private static boolean checkEnd(){
        // check end
        // if end if grater than queue length change it to 0

        boolean x = false;// return value

        if(end == queue.length){
            // change end
            end = 0;

        }

        if (queue[end]== null){
            // if end index is not full this one runs
            // help to stop getting data in createQueue method

            x = true;

        }else {
            // if end is full this one runs
            // it means all waiting list is full
            System.out.println("\n\tWaiting List is Full");
        }

        return x;

    }


    private static void createQueue (){
```

```
        // adding person
        //change end person's number

        Scanner scanner = new Scanner(System.in);

        if (checkEnd()){

            //get details

            while (true) {
                // get user's details
                System.out.print("\nEnter Customer First Name: ");
                String firstName = scanner.nextLine().toUpperCase();

                System.out.print("\nEnter Customer SurName: ");
                String surname = scanner.nextLine().toUpperCase();

                try {
                    // check error in card number
                    System.out.print("\nEnter Customer Card Number: ");
                    int cardNum = Integer.parseInt(scanner.nextLine());
                }
                catch (NumberFormatException e) {
                    // card number error message
                    System.out.println("\n\tSorry, Invalid Input. Try Again.");
                }

                try {
                    // check error in guest count
                    System.out.print("\nEnter Customer's Guest Count: ");
                    int guest = Integer.parseInt(scanner.nextLine());
                }
                catch (NumberFormatException e) {
                    // check error in guest count
                    System.out.println("\n\tSorry, Invalid Input. Try Again.");
                }

                if (!firstName.isEmpty()){

                    // add data to waiting list

                    int cardNum = 0;
                    int guest = 0;
                    Room rsvRoom = new Room(0, firstName, surname, cardNum,
guest);
                    // create object and add data
                    queue[end] = rsvRoom;

                    System.out.println("\n\tAdded to Queue List");
                    end ++;
                    break;
                }
                else {
                    //if user insert invalid input this part runs
                    System.out.println("\n\tSorry, Invalid Input. Try Again.");
                }
            }
        }

    }
```

```
}
```

Room.java

```
public class Room {
    // initialize variables
    private int roomNum;
    private String firstName;
    private String secondName;
    private int cardNum;
    private int numOfGuest;




    public Room(int roomNum, String firstName, String secondName, int cardNum,
int numOfGuest) {
        //cunstructor
        this.roomNum = roomNum;
        this.firstName = firstName;
        this.secondName = secondName;
        this.cardNum = cardNum;
        this.numOfGuest = numOfGuest;
    }
    // setter for roomNum
    public void setRoomNum(int roomNum) {
        this.roomNum = roomNum;
    }

    public int getRoomNum(){
        return roomNum;
    }// getter for roomnum

    public String getSecondName() {
        return secondName;
    } //getter for secondName

    public int getCardNum() {
        return cardNum;
    }// getter for cardNum

    public int getNumOfGuest() {
        return numOfGuest;
    }// getter for numOfGuest

    public String getCusName(){
        return firstName;
    }// getter for cusName
}
```

Test Case.

Using the black-box method, all of the test cases were selected. To that end, all test cases were created with the assumption that the user was unconcerned about internal code constructs or implementation specifics. The inputs and outputs were the primary concern in all the test cases.

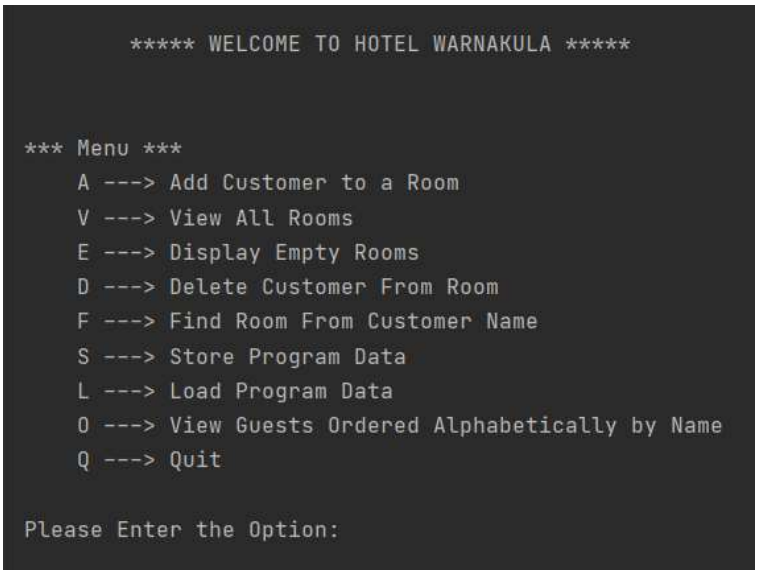
No.	Test Case	Expected Result	Actual Result	Pass/ Fail
1	Select an option in the menu.	User must enter correct letter in capital or simple. If the letter input by the user is not in the menu, displaying the error message and returned to the previous menu.	Displayed the menu.	Pass
2	(Add customer to room) Select “A” or “a”, Enter letters or space to Room Number, Enter space to Room Number.	Display the error messages to user and ask to input correct values.	Displayed the error messages.	Pass
3	(Add customer “Rukshan” to room 0) Select “A” or “a”, Enter Room Number “0”, Enter Customer details “Rukshan Fernando”.	User must enter “A” or “a” to Add customer. Then the user must input the customer desired room number and name. Room 3 should be reserved for the customer named Rukshan Fernando.	Room 3 is reserved for a customer named Rukshan Fernando.	Pass
4	(Add customer “Ruchintha Dias” to room 0) Select “A” or “a”, Enter Room Number “0”.	Display the error message of that not available. Ask to re input data.	Displayed the error messages.	Pass
5	Select “V” or “v”.	Display all the rooms with customers details.	Display all rooms.	Pass
6	Select “E” or “e”.	Display all the empty rooms.	Display all empty rooms.	Pass
7	Select “D” or “d”, Enter Room Number “0”.	Delete Customer from room.	Delete Customer of Rukshan from room “0”.	Pass
8	Select “D” or “d”, Remove empty room	Display error message	Display “Room 4 is Not Reserved”.	
9	Select “D” or “d”, Enter Room Number “8”.	Stop process and return to menu.	Stop process and return to menu.	Pass

10	Select “F” or “f”, Enter Customer First Name “Rukshan”.	Display room number with full name	Display Room No.0 Customer Name :RUKSHAN FERNANDO and Room No.7 Customer Name :RUKSHAN FERNANDEZ	Pass
11	Select “F” or “f”, Enter Room Number “8”.	Stop process and return to menu.	Stop process and return to menu.	Pass
12	(Not Found) Select “F” or “f”, Enter Customer First Name “Ruwanthi”.	Display Error Message.	Display message “No Customer Found Under This Name RUWANTHI”.	Pass
13	Select “S” or “s”.	Stored information into the text file. Display “Data stored” message.	Update Hotel_Reservations Text file. Display message “Successfully Stored to Hotel_Reservations File”.	Pass
14	Select “L” or “l”.	Loaded text file information into the program. Display “Data loaded” message.	Update program Display message “Successfully loaded from Hotel_Reservations File”.	Pass
15	(The data file is empty) Select “L” or “l”.	Display error message.	Display "Loading Error“ message.	Pass
16	Select “O” or “o”,	Show all the customers in alphabetical order.	Display all the rooms in alphabetical order.	Pass
17	Select “Q” or “q”	Program end with display message.	Program end. Display ” ***** Thank You! Have a Nice Day *****” message.	Pass
18	(in the class version) If the hotel is full.	Ask from user to input customer details in waiting list	Get details and added them in the waiting list	Pass
19	(in the class version) When the hotel is full time User can delete a customer from room.	Remove customer from a room and check waiting list, who is first comes. And add them into the empty room. Then clear customer data in the waiting list.	Delete customer and added customer from the waiting list. Clear that customer data from the waiting list.	Pass
20	(Adding to room path) Enter string value in the integer variable	Display error message and ask user to input it again.	Display Error message but user cannot input them again it save as null.	Fail

21	(Adding to room path) Enter 12 numbers in Card.	Data save	Showing error message save as null value	Fail

Test cases output (Screenshots).

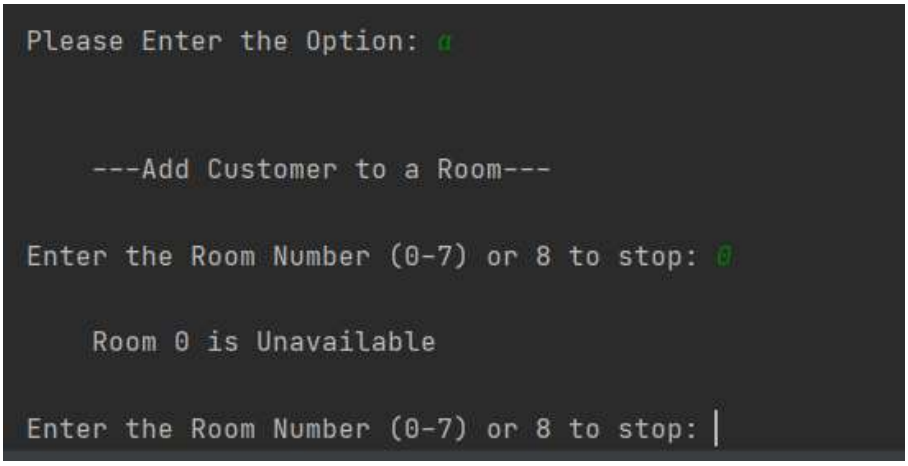
TestCase 1:



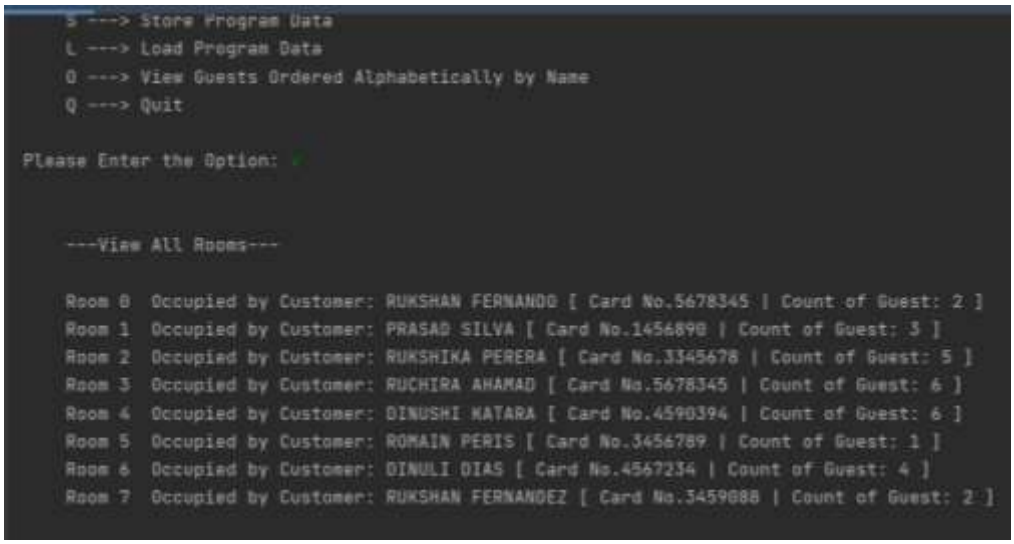
TestCase 2:

TestCase 3:

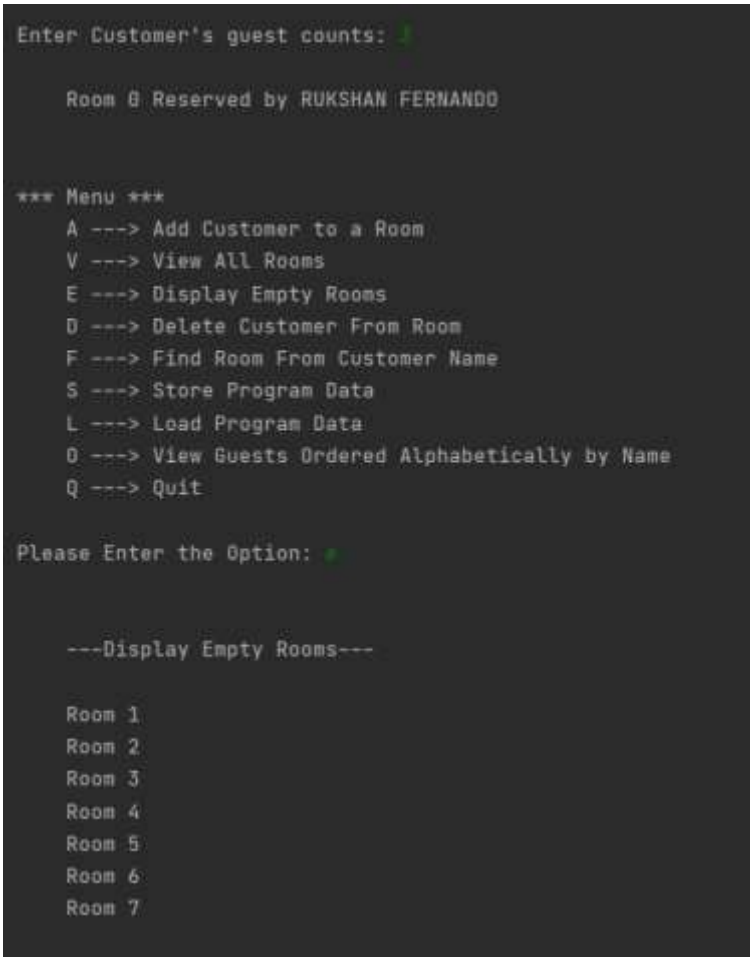
TestCase 4:



TestCase 5:



TestCase 6:



TestCase 7:

```
Please Enter the Option: d

    ---Delete Customer From Room---

Please Enter the room number (0-7) or 8 to stop: 0

    Room 0 is Available Again
```

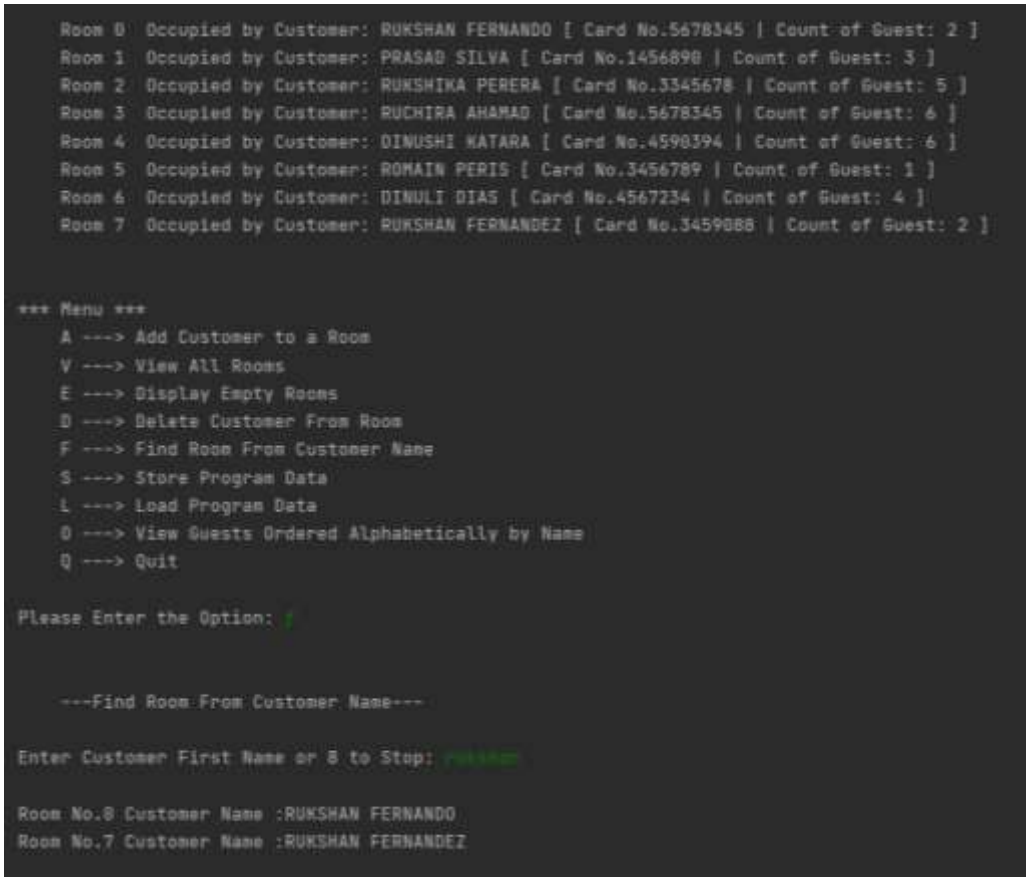
TestCase 8:

```
Please Enter the Option: d

    ---Delete Customer From Room---

Please Enter the room number (0-7) or 8 to stop: 0

    Room 0 is Not Reserved
```

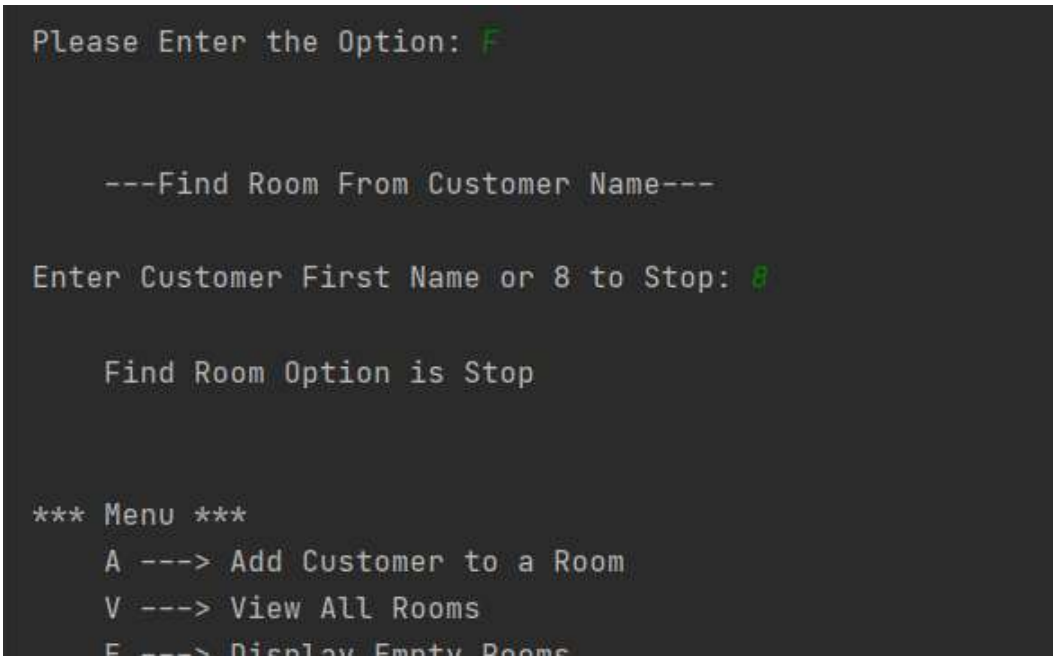
TestCase 9:

```
Please Enter the Option: d

    ---Delete Customer From Room---

Please Enter the room number (0-7) or 8 to stop: 8

    Delete Customer Option is Stopped


*** Menu ***
    A ---> Add Customer to a Room
    V ---> View All Rooms
```

TestCase 10:



TestCase 11:



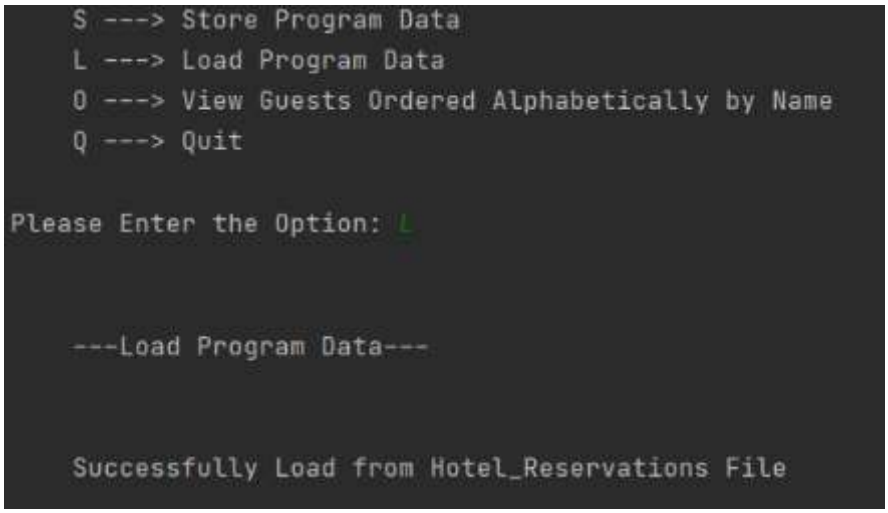
TestCase 12:

```
Please Enter the Option: f


    ---Find Room From Customer Name---

Enter Customer First Name or 8 to Stop: RUWANTHI


    No Customer Found Under This Name RUWANTHI


*** Menu ***
    A ---> Add Customer to a Room
```

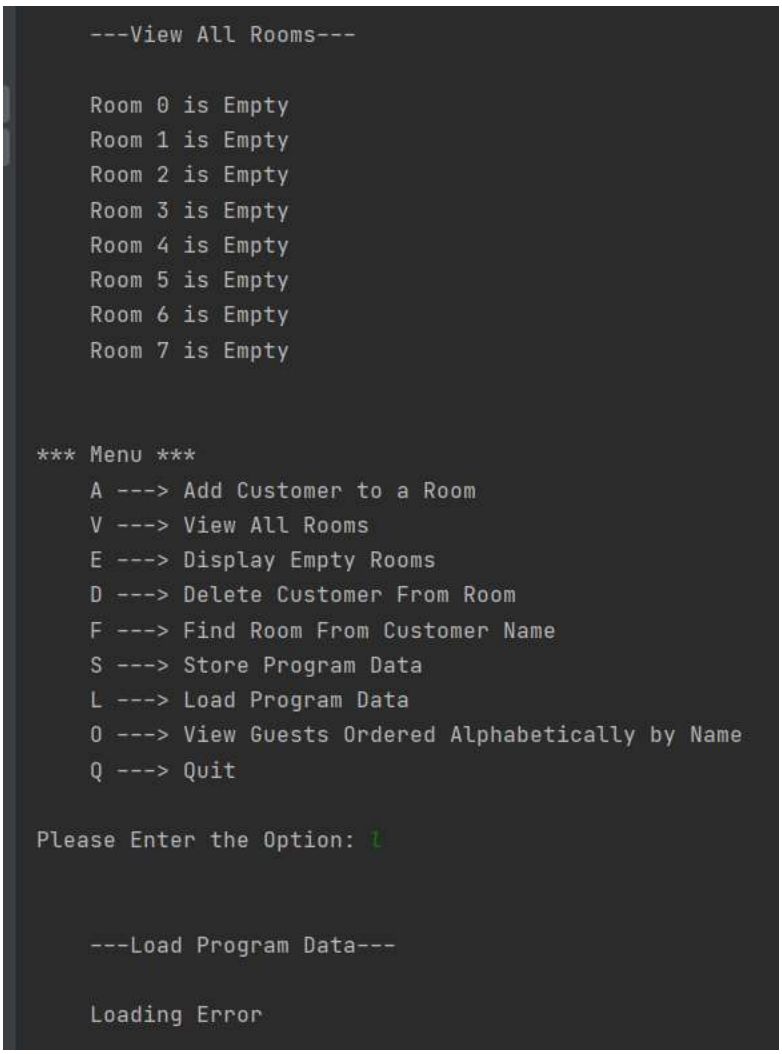
TestCase 13:

```
    O ---> View Guests Ordered Alphabetically by Name
    Q ---> Quit

Please Enter the Option: S


    ---Store Program Data---


    Successfully Stored to Hotel_Reservations File


*** Menu ***
    A ---> Add Customer to a Room
```

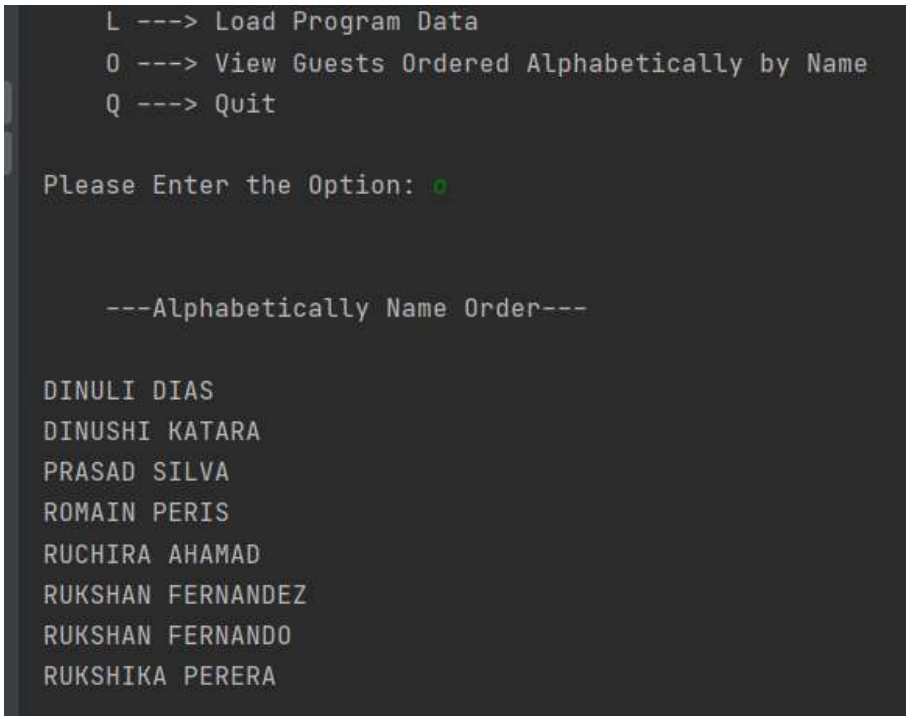
TestCase 14:



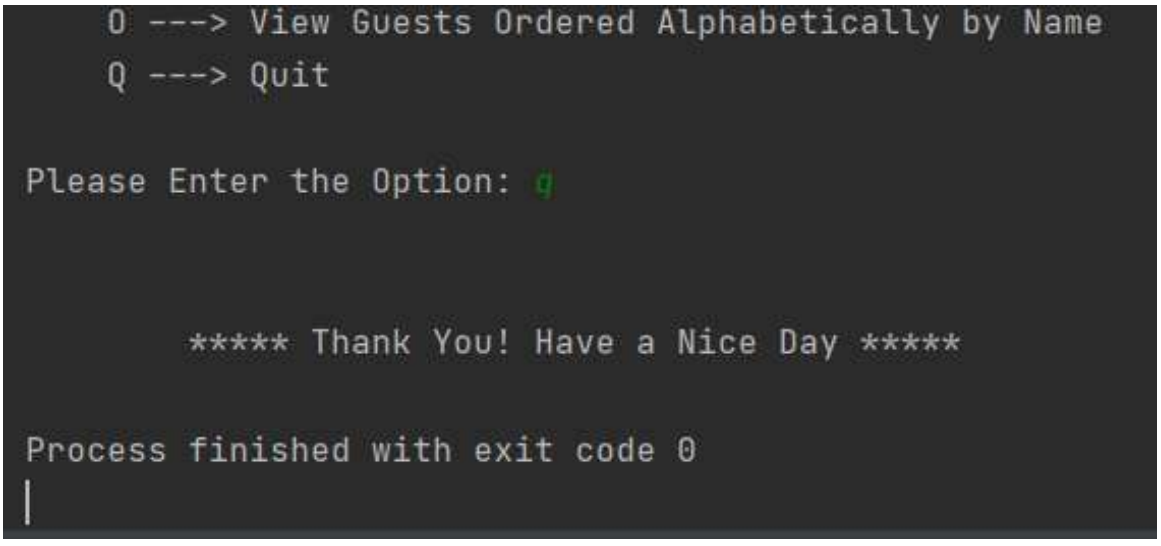
TestCase 15:



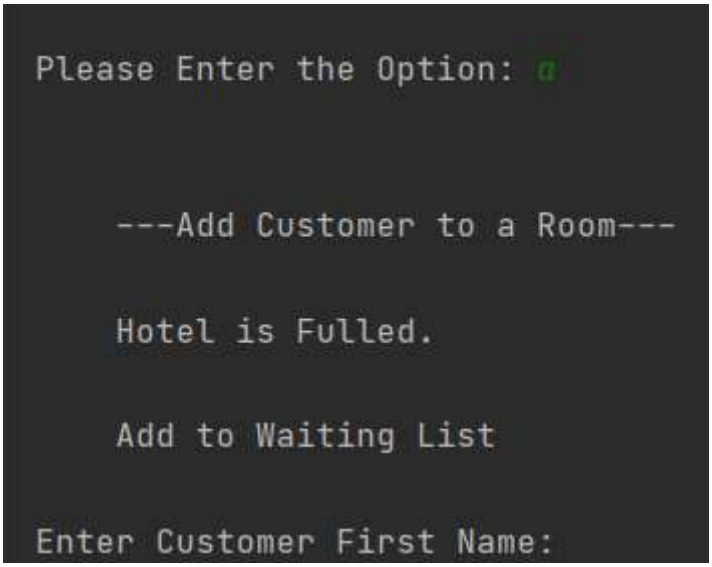
TestCase 16:



TestCase 17:

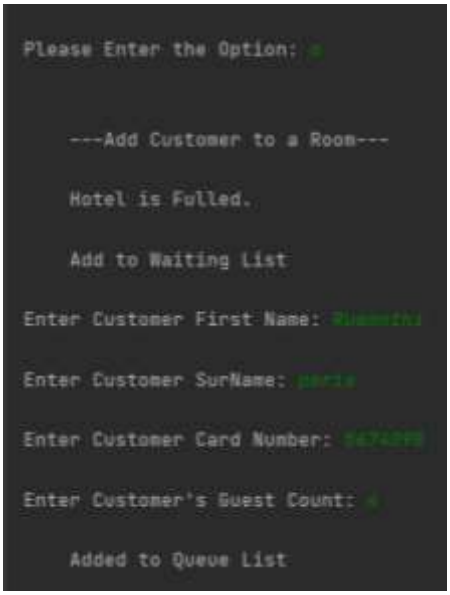


TestCase 18:




```
Hotel Warnakula
Room 0 Customer Name: RUKSHAN Surname: FERNANDO  Card: 5678345  Guest: 2
Room 1 Customer Name: PRASAD  Surname: SILVA Card: 1456890  Guest: 3
Room 2 Customer Name: RUKSHIKA    Surname: PERERA    Card: 3345678  Guest: 5
Room 3 Customer Name: RUCHIRA Surname: AHAMAD    Card: 5678345  Guest: 6
Room 4 Customer Name: DINUSHI Surname: KATARA    Card: 4590394  Guest: 6
Room 5 Customer Name: ROMAIN  Surname: PERIS Card: 3456789  Guest: 1
Room 6 Customer Name: DINULI  Surname: DIAS  Card: 4567234  Guest: 4
Room 7 Customer Name: Rukshan Surname: Fernandez Card: 3459088  Guest: 2
```

TestCase 19:



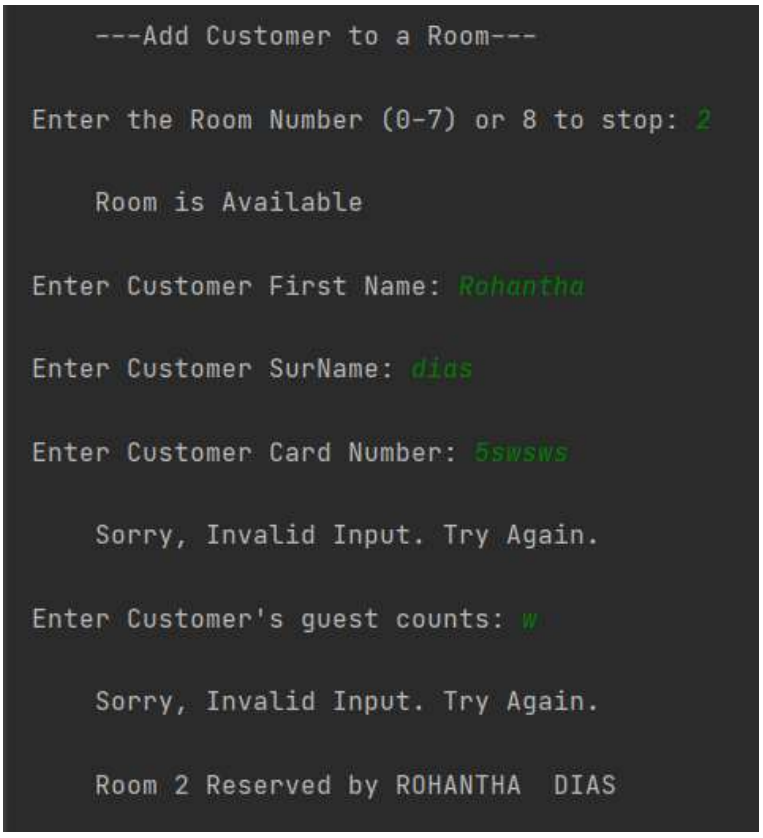
```
Please Enter the Option: d


    ---Delete Customer From Room---

Please Enter the room number (0-7) or 8 to stop: 6

    Room 6 is Available Again

    Customer: RUWANTHI PERIS Successfully Add to Room No.6 From Queue
```

```
Please Enter the Option: d


    ---Delete Customer From Room---

Please Enter the room number (0-7) or 8 to stop: 3

    Room 3 is Available Again

    Customer: AMMAR FARAHT Successfully Add to Room No.3 From Queue
```

```
Hotel Warnakula
Room 0  Customer Name: RUKSHAN  Surname: FERNANDO   Card: 5678345   Guest: 2
Room 1  Customer Name: PRASAD   Surname: SILVA  Card: 1456890   Guest: 3
Room 2  Customer Name: RUKSHIKA Surname: PERERA Card: 3345678   Guest: 5
Room 3  Customer Name: AMMAR    Surname: FARAHT Card: 0 Guest: 0
Room 4  Customer Name: DINUSHI  Surname: KATARA Card: 4590394   Guest: 6
Room 5  Customer Name: ROMAIN   Surname: PERIS  Card: 3456789   Guest: 1
Room 6  Customer Name: RUWANTHI Surname: PERIS  Card: 0 Guest: 0
Room 7  Customer Name: RUKSHAN  Surname: FERNANDEZ  Card: 3459088   Guest: 2
```

TestCase 20:

TestCase 21:



Discussion

Testcases which are applicable to each assignment are specifically included in this article. So first check that the methods are functioning properly in this setting, with the exception of the bubbble sort method. In the above cases or options, you can do what you want. As a result, there are A, V, E, D, F, S, L and O processes, as well as Q.

In task 2, the class version is the same as the array, however a Room class and a Hotel class have been developed. As with task 1, we've checked that each system is functioning properly.

In task 3, I inserted personal information such as first name, title, and credit card number to the Room setter and getter void class in the array version, and in the class version, it was in the Room class. And there are several test cases after the first 10 in the above test case.

In task 4, I have included a queue version for the class version, and under 18 and 19 I have included some test cases for g to ensure it's functioning properly.

References.

Furthermore knowledge,

- GeekforGeeks[2020], Bubble Sort Available on: https://www.geeksforgeeks.org/bubble-sort/ [Accessed 15th March 2021].

- Docs.oracle.com[2021], Formatter class methods in Java, Available on: https://docs.oracle.com/javase/7/docs/api/java/util/Formatter.html [Accessed 15th March 2021].

- Programiz [2021], Circular Queue Data Structure , Available on: http://quiz.geeksforgeeks.org/queue-set-1introduction-and-array-implementation/ [Accessed 17th March 2021].

...END...