Swinburne University of Technology

Advanced Programming (COS10033) Assignment 2

Lecturer: Ms. Lushaka Nisansala

Student Name: M.A. Nidula Sanketh Mallikarachchi

Course: UniLink Diploma in IT

Swinburne ID: 104756611

NCHS ID: 2023040050

Email: 104756611@student.swin.edu.au

Contact Number: 0741966164

Hotel Management System Documentation

Overview

This Hotel Management System is a Java-based application designed to manage room bookings in a hotel. It provides functionalities for initializing hotel rooms, booking and checking out rooms, displaying room status, and calculating the hotel's total income. The system also includes file handling capabilities for data persistence.

System Requirements

- Java Runtime Environment (JRE)
- Access to file system for reading/writing Database.txt

Class Overview

- Entrance: The main class that orchestrates the hotel management operations.

Key Methods and Functionalities

Initialization

- Entrance(): Constructor that initializes the room list and recommendations.
- initializeHotel(): Sets up hotel rooms with different types and features.

File Handling

- readFromFile(): Reads room booking data from Database.txt.
- writeToFile(): Writes current room booking data to Database.txt.

Room Display Functions

- displayAllRooms(): Displays all rooms and their statuses.
- displayAvailableRooms(): Shows only rooms that are currently unbooked.
- displayBookedRooms(): Lists all the rooms that are currently booked.

Room Booking

- checkInRoom(Room room, String guestName, int noOfPeople, int noOfDays, int guestID): Handles room check-in.
- validateCheckInRoom(Room room): Validates if a room is available for check-in.
- case2_1_BookingANormalRoom(): Facilitates booking of normal rooms.
- case2_2_BookingASpecialRoom(): Facilitates booking of special rooms.

Check-Out Process

- validateRoomCheckOut(Room room): Ensures that a room is currently booked before checkout.
- case3_CheckOutRoom(): Handles the room check-out process.

Statistical Reporting

- case4_calculateTotalIncome(): Calculates and displays the total income from room bookings.

Utility Methods

- selectRoomByRoomID(int roomID): Returns a room based on the room ID.
- displayRecommendedRooms(): Displays a list of recommended rooms.
- normalRoomRecommendations(String roomCategory, String roomType): Recommends rooms based on category and type.
- specialRoomRecommendations(String roomType): Provides recommendations for special rooms.

Validation Functions

- validateRoomType(String roomType): Validates the input room type.
- validateRoomCategory(String roomCategory): Validates the input room category.
- validateRoomID(int roomID): Checks if the provided room ID is valid.
- validateRoomRecommendations(int roomID): Validates if the chosen room is among the recommended ones.
- validateNoOfGuestVsRoomTypeForNormalRoom(String roomType, int noOfPeople): Ensures that the number of guests is suitable for the room type.
- validateSpecialRoomType(String roomType): Validates room type for special rooms.
- validateSpecialRoomVsNoOfGuest(String roomType, int noOfPeople): Ensures the number of guests is appropriate for special rooms.

Main Method

- main(String[] args): The entry point of the application. Contains the menu-driven interface for interacting with the system.

Global Variables

- noOfRoomsBooked: Tracks the number of booked rooms.
- redColor, greenColor, resetColor: Used for color coding the output for better readability.

Exception Handling

- Custom exceptions like AccomodationException are used for handling specific scenarios related to accommodation.

User Interface

The system uses a console-based menu-driven interface to interact with the user. It provides options to display rooms, check in and check out rooms, and view statistical reports.

File Structure

The system relies on a file named Database.txt for persisting room booking data. This file is read at startup and updated upon exit.

Table Of Content

Room.java

```
package assignment2;

public abstract class Room {
    protected int roomID;
    protected int guestID;
    protected boolean roomStatus;
    protected String roomStatusDescription;
    protected String roomCategory;
    protected String roomType;
    protected double roomCostPerDay;
    protected int noOfDays;
    protected double roomTotalPrice;
    protected int noOfPeople;
    protected String guestName;
    protected String roomDescription;
    protected int maxNumberOfGuestPerRoom;
    protected int emergencyServicesCount;
    protected boolean specialRoom;

    public Room(int roomID, String roomType) {
        if (!roomType.equalsIgnoreCase("SINGLE") && !roomType.equalsIgnoreCase("DOUBLE")
&& !roomType.equalsIgnoreCase("TRIPLE")) {
            throw new IllegalArgumentException("Wrong Room Type!");
        }
        this.roomID = roomID;
        this.guestID = 0;
        this.guestName = null;
        this.roomType = roomType;
        this.roomCostPerDay = 0;
        this.roomTotalPrice = 0;
        this.noOfPeople = 0;
        this.noOfDays = 0;
        this.roomStatus = false;
        this.roomStatusDescription = "Available";
        this.specialRoom = false;
    }

    protected abstract double totalCostOfStay();
    protected abstract void determinePrice();
    public void checkInRoom(int guestID, int numDays) {
        this.roomStatus = true;
        this.roomStatusDescription = "Booked";
        this.guestID = guestID;
        this.noOfDays = numDays;
        this.roomTotalPrice = totalCostOfStay();
    }
    public void checkOutRoom(){
        this.guestID = 0;
        this.roomStatus = false;
        this.roomStatusDescription = "Available";
        this.noOfDays = 0;
        this.roomTotalPrice = 0;
        this.noOfPeople = 0;
        guestName = null;
    }

    @Override
```

```
    public String toString() {
        return  "Room ID: " + roomID +
                "\tGuest ID: " + guestID +
                "\t\tRoom Status: " + roomStatusDescription +
                "\t\tRoom Category: " + roomCategory +
                "\t\tRoom Type: " + roomType +
                "\t\tRoom Cost:" + roomCostPerDay +
                "\t\tDays Staying: " + noOfDays +
                "\t\tTotal Price: " + roomTotalPrice +
                "\t\tNo Of People: " + noOfPeople +
                "\t\tName: " + guestName;
    }

    //Setters
    public void setRoomCostPerDay(double roomCostPerDay) {
        this.roomCostPerDay = roomCostPerDay;
    }
    public void setMaxNumberOfGuestPerRoom(int maxNumberOfGuestPerRoom) {
        this.maxNumberOfGuestPerRoom = maxNumberOfGuestPerRoom;
    }
    public void setGuestID(int guestID) {
        this.guestID = guestID;
    }
    public void setNoOfDays(int noOfDays) {
        this.noOfDays = noOfDays;
    }
    public void setRoomTotalPrice(double roomTotalPrice) {
        this.roomTotalPrice = roomTotalPrice;
    }
    public void setNoOfPeople(int noOfPeople) {
        this.noOfPeople = noOfPeople;
    }
    public void setGuestName(String guestName) {
        this.guestName = guestName;
    }
    public void setRoomStatus(boolean roomStatus) {
        this.roomStatus = roomStatus;
    }
    public void setRoomStatusDescription(String roomStatusDescription) {
        this.roomStatusDescription = roomStatusDescription;
    }

    //Getters
    public boolean isRoomStatus() {
        return roomStatus;
    }
    public String getRoomCategory() {
        return roomCategory;
    }
    public String getRoomType() {
        return roomType;
    }
    public int getRoomID() {
        return roomID;
    }
    public int getGuestID() {
        return guestID;
    }
```

```
    public int getNoOfDays() {
        return noOfDays;
    }
    public double getRoomTotalPrice() {
        return roomTotalPrice;
    }
    public int getNoOfPeople() {
        return noOfPeople;
    }
    public String getGuestName() {
        return guestName;
    }
    public boolean isSpecialRoom() {
        return specialRoom;
    }


    public void printReceipt(){
    String formattedString =
            "------------RECEIPT-------------\n"+
            "Guest Name: " + guestName + "\n" +
            "Room Category: " + roomCategory + "\n" +
            "Room Type: " + roomType + "\n" +
            "Room ID: " + roomID + "\n" +
            "Room Price Per Day: " + roomCostPerDay + "\n" +
            "Number of Days Staying: " + noOfDays + "\n" +
            "Total Cost of the Room: " + roomTotalPrice + "\n" +
            "Guest ID: " + guestID+ "\n" +
            "Room Description: " + roomDescription;

    System.out.println(formattedString);
   }
}
```

StandardRoom.java

```
package assignment2;

public class StandardRoom extends Room{
    public StandardRoom(int roomID, String roomType) {
        super(roomID, roomType);
        this.roomCategory = "STANDARD";
        this.roomDescription = "Standard Room With Basic Amenities";
        determinePrice();

    }

    @Override
    protected void determinePrice() {
        if(roomType == "SINGLE"){
            setRoomCostPerDay(230);
        }
        else if (roomType =="DOUBLE") {
            setRoomCostPerDay(280);
        }
        else if (roomType == "TRIPLE"){
```

```
            setRoomCostPerDay(350);
        }
    }

    @Override

    public double totalCostOfStay() {
        return roomCostPerDay*noOfDays;
    }
}
```

DeluxeRoom.java

```
package assignment2;

public class DeluxeRoom extends Room{
    public DeluxeRoom(int roomID, String roomType) {
        super(roomID, roomType);
        this.roomCategory = "DELUXE";
        this.roomDescription = "Deluxe Rooms with More Space and a Bathtub";
        determinePrice();
    }

    @Override
    protected void determinePrice() {
        if(roomType == "SINGLE"){
            setRoomCostPerDay(350);
        }
        else if (roomType =="DOUBLE") {
            setRoomCostPerDay(430);
        }
        else if (roomType == "TRIPLE"){
            setRoomCostPerDay(500);
        }
    }

    @Override
    public double totalCostOfStay() {
        return roomCostPerDay*noOfDays;
    }
}
```

PremiumRoom.java

```
package assignment2;

public class PremiumRoom extends Room{
    public PremiumRoom(int roomID, String roomType) {
        super(roomID, roomType);
        this.roomCategory = "PREMIUM";
        this.roomDescription = "Premium Room with Spa Area and a Kitchenette";
        determinePrice();
    }

    @Override
    protected void determinePrice() {
        if(roomType == "SINGLE"){
            setRoomCostPerDay(500);
        }
        else if (roomType =="DOUBLE") {
            setRoomCostPerDay(600);
        }
```

```
        else if (roomType == "TRIPLE"){
            setRoomCostPerDay(690);
        }
    }

    @Override
    public double totalCostOfStay() {
        return roomCostPerDay*noOfDays;
    }
}
```

SpecialServices.java

```
package assignment2;

public interface SpecialServices {
    int  rampLength = 10;
    int rampWidth = 10;

    void provideAssistance();
    void callEmergencyServices();
}
```

SpecialDouble.java

```
package assignment2;

public class SpecialDouble extends StandardRoom implements SpecialServices{
    protected int emergencyNumber;

    public SpecialDouble(int roomID, String roomType) {
        super(roomID, roomType);
        this.roomDescription = "Special Double Room With Fitted Ramps, Emergency Calling
Facilities & Close to Beach";
        this.emergencyNumber = 911;
        this.specialRoom = true;
    }

    @Override
    public void provideAssistance() {

    }

    @Override
    public void callEmergencyServices() {
        emergencyServicesCount++;
    }


    @Override
    public String toString() {
        return  "Room ID: " + roomID +
                "\tGuest ID: " + guestID +
                "\t\tRoom Status: " + roomStatusDescription +
                "\t\tRoom Category: " + roomCategory +
                "\t\tRoom Type: " + roomType +
                "\t\tRoom Cost:" + roomCostPerDay +
                "\t\tDays Staying: " + noOfDays +
                "\t\tTotal Price: " + roomTotalPrice +
                "\t\tNo Of People: " + noOfPeople +
```

```
                "\t\tName: " + guestName +
                "\t\tSPECIAL DOUBLE ROOM"
                ;

    }

    @Override
    public void printReceipt(){
    String formattedString =
            "------------RECEIPT-------------\n"+
            "Guest Name: " + guestName + "\n" +
            "Room Category: " + roomCategory + " SPECIAL "+"\n" +
            "Room Type: " + roomType + "\n" +
            "Room ID: " + roomID + "\n" +
            "Room Price Per Day: " + roomCostPerDay + "\n" +
            "Number of Days Staying: " + noOfDays + "\n" +
            "Total Cost of the Room: " + roomTotalPrice + "\n" +
            "Guest ID: " + guestID+ "\n" +
            "Room Description: " + roomDescription;

    System.out.println(formattedString);

    }


}
```

SpecialTriple.java

```
package assignment2;

public class SpecialTriple extends StandardRoom implements SpecialServices{
    protected int emergencyNumber;

    public SpecialTriple(int roomID, String roomType) {
        super(roomID, roomType);
        this.roomDescription = "Special Triple Room With Fitted Ramps, Emergency Calling
Facilities & Close to Beach";
        this.emergencyNumber = 911;
        this.specialRoom = true;
    }


    @Override
    public void provideAssistance() {

    }

    @Override
    public void callEmergencyServices() {
        emergencyServicesCount++;
    }

    @Override
    public String toString() {
        return  "Room ID: " + roomID +
                "\tGuest ID: " + guestID +
                "\t\tRoom Status: " + roomStatusDescription +
```

```
                "\t\tRoom Category: " + roomCategory +
                "\t\tRoom Type: " + roomType +
                "\t\tRoom Cost:" + roomCostPerDay +
                "\t\tDays Staying: " + noOfDays +
                "\t\tTotal Price: " + roomTotalPrice +
                "\t\tNo Of People: " + noOfPeople +
                "\t\tName: " + guestName +
                "\t\tSPECIAL TRIPLE ROOM";
    }

    @Override
    public void printReceipt(){
    String formattedString =
            "------------RECEIPT-------------\n"+
            "Guest Name: " + guestName + "\n" +
            "Room Category: " + roomCategory + " SPECIAL "+"\n" +
            "Room Type: " + roomType + "\n" +
            "Room ID: " + roomID + "\n" +
            "Room Price Per Day: " + roomCostPerDay + "\n" +
            "Number of Days Staying: " + noOfDays + "\n" +
            "Total Cost of the Room: " + roomTotalPrice + "\n" +
            "Guest ID: " + guestID+ "\n" +
            "Room Description: " + roomDescription;

    System.out.println(formattedString);

    }



}
```

AccomodationException.java

```
package assignment2;

public class AccomodationException extends Exception {
    public AccomodationException(String message) {
        super(message);
    }
}
```

Entrance.java

```
package assignment2;

import java.util.ArrayList;
import java.util.Scanner;
import java.io.*;

public class Entrance {

    //-------------------------------------------------IMPORTANT-------------------------
-----------------------------------
    ArrayList<Room> accommodations;

    public static Entrance entrance = new Entrance();
    public static Scanner scan = new Scanner(System.in);
```

```
//INITIALIZATION
public Entrance() {
    this.accommodations = new ArrayList<>();
    this.roomRecommendations = new ArrayList<>();
}
public void initializeHotel() {
    Room room;
    int roomNumber = 101;

    int standardRoomCount = 0, deluxeRoomCount = 0, premiumRoomCount = 0;

    while (standardRoomCount < 12) {
        if (standardRoomCount < 4) {
            room = new StandardRoom(roomNumber, "SINGLE");
            room.setMaxNumberOfGuestPerRoom(2);
        } else if (standardRoomCount < 8) {
            room = new StandardRoom(roomNumber, "DOUBLE");
            room.setMaxNumberOfGuestPerRoom(4);
        } else {
            room = new StandardRoom(roomNumber, "TRIPLE");
            room.setMaxNumberOfGuestPerRoom(6);
        }
        accommodations.add(room);
        standardRoomCount++;
        roomNumber++;
    }

    while (deluxeRoomCount < 12) {
        if (deluxeRoomCount < 4) {
            room = new DeluxeRoom(roomNumber, "SINGLE");
            room.setMaxNumberOfGuestPerRoom(2);
        } else if (deluxeRoomCount < 8) {
            room = new DeluxeRoom(roomNumber, "DOUBLE");
            room.setMaxNumberOfGuestPerRoom(4);
        } else {
            room = new DeluxeRoom(roomNumber, "TRIPLE");
            room.setMaxNumberOfGuestPerRoom(6);
        }
        accommodations.add(room);
        deluxeRoomCount++;
        roomNumber++;
    }

    while (premiumRoomCount < 12) {
        if (premiumRoomCount < 4) {
            room = new PremiumRoom(roomNumber, "SINGLE");
            room.setMaxNumberOfGuestPerRoom(2);
        } else if (premiumRoomCount < 8) {
            room = new PremiumRoom(roomNumber, "DOUBLE");
            room.setMaxNumberOfGuestPerRoom(4);
        } else {
            room = new PremiumRoom(roomNumber, "TRIPLE");
            room.setMaxNumberOfGuestPerRoom(6);
        }
        accommodations.add(room);
        premiumRoomCount++;
        roomNumber++;
    }


    room = new SpecialDouble(roomNumber,"DOUBLE");
    accommodations.add(room);
```

```
        roomNumber++;

        room = new SpecialTriple(roomNumber,"TRIPLE");
        accommodations.add(room);

    }
    public void readFromFile(){
        File file = new File("Database.txt");

        if (!file.exists()) {
            try {
                file.createNewFile();
            } catch (IOException e) {
                e.printStackTrace();
            }
            return;
        }

        try (BufferedReader reader = new BufferedReader(new FileReader("Database.txt")))
{
            String line;
            while ((line = reader.readLine()) != null) {
                // Split the line using "\t|\t" as the delimiter
                String[] elements = line.split("\t\\|\t");

                // Check if the element is not empty before parsing
                if (!elements[0].trim().isEmpty()) {
                    int roomID = Integer.parseInt(elements[0].trim());
                    int guestID = Integer.parseInt(elements[1].trim());
                    int noOfDays = Integer.parseInt(elements[2].trim());
                    double roomTotalPrice = Double.parseDouble(elements[3].trim());
                    int noOfPeople = Integer.parseInt(elements[4].trim());
                    String guestName = elements[5].trim();

                    // Use the variables as needed
                    Room room = selectRoomByRoomID(roomID);
                    if (room != null) {
                        room.setGuestID(guestID);
                        room.setNoOfDays(noOfDays);
                        room.setRoomTotalPrice(roomTotalPrice);
                        room.setNoOfPeople(noOfPeople);
                        room.setGuestName(guestName);
                        room.setRoomStatus(true);
                        room.setRoomStatusDescription("Booked");
                    } else {
                        // Handle the case where the room with the specified ID is not
found
                        System.out.println("Room with ID " + roomID + " not found.");
                    }
                } else {
                    // Handle the case where the string is empty (if needed)
                    System.out.println("Skipping empty line.");
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }

    }

    //DISPLAY ROOMS
    public void displayAllRooms() {
```

```
        for (Room room : accommodations) {
            if (room.isRoomStatus()) {
                System.out.println(redColor + room + resetColor);
            } else {
                System.out.println(greenColor + room + resetColor);
            }
        }
    }
    public void displayAvailableRooms(){
        for(Room room: accommodations){
            if(!room.isRoomStatus()){
                System.out.println(greenColor+room+resetColor);
            }
        }
    }
    public void displayBookedRooms(){
        for(Room room: accommodations){
            if(room.isRoomStatus()){
                System.out.println(redColor+room+resetColor);
            }
        }
    }

    //CHECK-IN ROOM
    public void checkInRoom(Room room, String guestName, int noOfPeople, int noOfDays,
int guestID){
        room.setNoOfPeople(noOfPeople);
        room.setGuestName(guestName);
        room.checkInRoom(guestID,noOfDays);
//        noOfRoomsBooked++;
//        totalIncomeOfHotel+=room.totalCostOfStay();
    }
    public boolean validateCheckInRoom(Room room) throws AccomodationException{
        boolean isAvailable = false;
        try {
            if(!room.isRoomStatus()){
                isAvailable = true;
            }
            else {
                throw new AccomodationException("This room is Already Booked!");
            }
        }catch (AccomodationException e){
            System.out.println(e.getMessage());
        }
        return isAvailable; } //User Tries to Book a Room That has Already Been Booked

    //CHECK-OUT ROOM
    public boolean validateRoomCheckOut(Room room) throws AccomodationException {
        boolean isBooked = false;
        try{
            if(room.isRoomStatus()){
                isBooked = true;
            }
            else {
                throw new AccomodationException("This Room is Not Booked!");
            }
        }catch (AccomodationException e){
            System.out.println(e.getMessage());
        }

        return isBooked;
    } //User Tires to Check Out a Room That has noot been booked before
```

```
    //OBJECT PERSISTENCE
    public void writeToFile(){
        StringBuilder lines = new StringBuilder();

        for (Room room : accommodations) {
            if (room.isRoomStatus()) {
                String a = "\t|\t";
                String line = room.getRoomID() + a +
                        room.getGuestID() + a +
                        room.getNoOfDays() + a +
                        room.getRoomTotalPrice() + a +
                        room.getNoOfPeople() + a +
                        room.getGuestName();

                lines.append(line).append(System.lineSeparator()); // Use
System.lineSeparator() for cross-platform newline
            }
        }

        try (BufferedWriter writer = new BufferedWriter(new FileWriter("Database.txt")))
{
            writer.write(lines.toString());
            System.out.println("Data written to the file successfully.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    //BASIC UTILITIES
    public Room selectRoomByRoomID(int roomID) {
        Room targetRoom = null;
        for (Room room : accommodations) {
            if (roomID == room.getRoomID()) {
                targetRoom = room;
            }
        }
        if (targetRoom == null) {
            System.out.println("No Room Found!");
        }
        return targetRoom;
    }

    //MAIN METHODS
    public void case2_1_BookingANormalRoom() throws AccomodationException {

        boolean specialRoom = false;

        scan.nextLine();

        System.out.print("Enter Guest Name: ");
        String guestName = scan.nextLine();

        // scan.nextLine(); // Remove this line
        System.out.print("Enter Number of Days of Stay: ");
        int noOfDays = scan.nextInt();
        scan.nextLine(); // Add this here to consume the newline after the integer input


        int noOfPeople;
        do{
            System.out.print("Enter Number of People Staying: ");
            noOfPeople = scan.nextInt();
            scan.nextLine();
```

```
            if(noOfPeople<=0){
                System.out.println("Enter Valid Number!");
            }
            if(noOfPeople>6){
                System.out.println("Maximum Accommodation is 6 People per Room");
            }
        }while (noOfPeople<=0 || noOfPeople>6);

        String roomType;
        do{
            do{
                System.out.print("Enter Room Type: ");
                roomType = scan.nextLine();
            }while (!entrance.validateRoomType(roomType));
        }while (!entrance.validateNoOfGuestVsRoomTypeForNormalRoom(roomType,noOfPeople));

        String roomCategory;
        do{
            System.out.print("Enter Room Category: ");
            roomCategory = scan.nextLine();
        }while (!entrance.validateRoomCategory(roomCategory));

        entrance.normalRoomRecommendations(roomCategory,roomType);
        entrance.displayRecommendedRooms();

        int roomID;
        Room room;
        do{
            do{
                System.out.print("Enter Room ID: ");
                roomID = scan.nextInt();
            }while (!entrance.validateRoomRecommendations(roomID));
            room = entrance.selectRoomByRoomID(roomID);
        }while (!entrance.validateCheckInRoom(room));

        int guestID = roomID+100;

        room = entrance.selectRoomByRoomID(roomID);

        entrance.checkInRoom(room, guestName, noOfPeople, noOfDays, guestID);

        entrance.roomRecommendations.clear();

        System.out.println(greenColor);
        room.printReceipt();
        System.out.println(resetColor);


    }
    public void case2_2_BookingASpecialRoom() throws AccomodationException {

        String roomType;

        scan.nextLine();
        do{
            System.out.print("Enter Room Type: ");
            roomType = scan.nextLine();
        }while (!entrance.validateSpecialRoomType(roomType));


        int noOfPeople = 0;

        do{
```

```
            do{
                System.out.print("Enter Number of People Staying: ");
                noOfPeople = scan.nextInt();
                if(noOfPeople<=0){
                    System.out.println("Enter a Valid Number!");
                }
                if(noOfPeople>6){
                    System.out.println("No More than 6 People Can Stay in a Room!");
                }
            }while (noOfPeople<0 ||noOfPeople>6);
        }while (!entrance.validateSpecialRoomVsNoOfGuest(roomType,noOfPeople));

        Room room;

        if(entrance.specialRoomRecommendations(roomType)){
            entrance.displayRecommendedRooms();

            int roomID;
            do{
                do{
                    System.out.print("Enter Room ID to Book Room: ");
                    roomID =  scan.nextInt();
                    room = entrance.selectRoomByRoomID(roomID);
                }while (!entrance.validateRoomRecommendations(roomID));
            }while (!entrance.validateCheckInRoom(room));

            scan.nextLine();

            System.out.print("Enter Guest Name: ");
            String guestName = scan.nextLine();

            System.out.print("Enter Number of Days Staying: ");
            int noOfDays = scan.nextInt();

            int guestID = roomID+100;

            entrance.checkInRoom(room,guestName,noOfPeople,noOfDays,guestID);

            System.out.println(greenColor);
            room.printReceipt();
            System.out.println(resetColor);
        }
    }
    public void case3_CheckOutRoom() throws AccomodationException {

        entrance.displayBookedRooms();

        Room room;
        int roomID;
        do{
            do{
                System.out.print("Enter Room ID: ");
                roomID = scan.nextInt();
            }while (!entrance.validateRoomID(roomID));

            room = entrance.selectRoomByRoomID(roomID);
        }while (!entrance.validateRoomCheckOut(room));

        room.checkOutRoom();
        noOfRoomsBooked--;
    }
    public void case4_calculateTotalIncome() {
        double totalIncome = 0;
```

```
        for (Room room : accommodations) {
            totalIncome = 0;
            if (room.isRoomStatus()) {
                totalIncome += room.getRoomTotalPrice();
            }
        }
        System.out.println("Total Income of Hotel: " + totalIncome);
    }

    //MAIN
    public static void main(String[] args) throws AccomodationException {

        entrance.initializeHotel();
        entrance.readFromFile();

        while (true) {
            System.out.println("Hotel Management System Menu:");
            System.out.println("1. Display Rooms");
            System.out.println("2. Check In Rooms");
            System.out.println("3. Check Out Rooms");
            System.out.println("4. Display Statistical Report");
            System.out.println("5. Exit");
            System.out.print("Enter your choice (1-5): ");

            int choice = scan.nextInt();

            switch (choice) {
                case 1:
                    System.out.println("Displaying Rooms\n");
                    System.out.println("Display Rooms Submenu:");
                    System.out.println("\t1. Display All Rooms");
                    System.out.println("\t2. Display Available Rooms");
                    System.out.println("\t3. Display Booked Rooms");
                    System.out.println("\t4. Back to Main Menu");
                    System.out.print("Enter your choice (1-4): ");

                    int subChoice = scan.nextInt();

                    switch (subChoice) {
                        case 1:
                            System.out.println("Displaying All Rooms");
                            entrance.displayAllRooms();
                            break;
                        case 2:
                            System.out.println("Displaying Available Rooms");
                            entrance.displayAvailableRooms();
                            break;
                        case 3:
                            System.out.println("Displaying Booked Rooms");
                            entrance.displayBookedRooms();
                            break;
                        case 4:
                            return;
                        default:
                            System.out.println("Invalid choice. Please enter a number
between 1 and 4.");
                    }
                    break;
                case 2:
                    System.out.println("Checking In Rooms\n");
                    System.out.println("Check In Submenu:");
                    System.out.println("\t1. Book a Normal Room");
                    System.out.println("\t2. Book a Special Room");
```

```
                    System.out.println("\t3. Back to Main Menu");
                    System.out.print("Enter your choice (1-3): ");

                    subChoice = scan.nextInt();

                    switch (subChoice) {
                        case 1:
                            entrance.case2_1_BookingANormalRoom();
                            break;
                        case 2:
                            entrance.case2_2_BookingASpecialRoom();
                            break;
                        case 3:
                            return; // Return to the main menu
                        default:
                            System.out.println("Invalid choice. Please enter a number
between 1 and 3.");
                    }
                    break;
                case 3:
                    entrance.case3_CheckOutRoom();
                    break;
                case 4:
                    // Call a method to display statistical report
                    System.out.println(greenColor);
                    System.out.println("Displaying Statistical Report...");
                    entrance.case4_calculateTotalIncome();
                    System.out.println(resetColor);
                    break;
                case 5:
                    System.out.println("Exiting the program. Goodbye!");
                    entrance.writeToFile();
                    System.exit(0);
                    break;
                default:
                    System.out.println("Invalid choice. Please enter a number between 1
and 5.");
            }
        }
    }

    //-----------------------------------------------------EXTRA WORK-----------------------
-------------------------------------

    String redColor = "\u001B[31m";
    static String greenColor = "\u001B[38;2;11;244;102m";
    static String resetColor = "\u001B[0m";

    ArrayList<Room> roomRecommendations;
    public void displayRecommendedRooms(){
        for(Room room: roomRecommendations){
            if (!room.isRoomStatus()) {
                System.out.println(greenColor + room + resetColor);
            }
        }
    }
    public void normalRoomRecommendations(String roomCategory, String roomType){
        boolean roomsFound = false;

        roomRecommendations.clear();

        for (Room room : accommodations) {
            if (roomCategory.equalsIgnoreCase(room.getRoomCategory()) &&
```

```
roomType.equalsIgnoreCase(room.getRoomType())) {
                if(!room.isRoomStatus() && !room.isSpecialRoom()){
                    roomRecommendations.add(room);
                    roomsFound = true;  // Update the flag when a matching room is found
                }else {
                    if(!room.isSpecialRoom()){
                        roomRecommendations.add(room);
                    }
                    // booked rooms are also added to handel exceptions here
                }
            }
        }

        if (!roomsFound) {
            boolean roomsFound2 = false;

            System.out.println("This Type of Room is Not Available at The Moment,
Consider Following Recommended Rooms!");

            if (roomType.equalsIgnoreCase("SINGLE")) {
                for (Room room : accommodations) {
                    if (room.getRoomType().equalsIgnoreCase("SINGLE") ||
                            room.getRoomType().equalsIgnoreCase("DOUBLE") ||
                            room.getRoomType().equalsIgnoreCase("TRIPLE")) {
                        if (!room.isRoomStatus() && !room.isSpecialRoom()) {
                            roomRecommendations.add(room);
                            roomsFound2 = true;
                        }
                    }
                }
            }

            if (roomType.equalsIgnoreCase("DOUBLE")) {
                for (Room room : accommodations) {
                    if (room.getRoomType().equalsIgnoreCase("DOUBLE") ||
                            room.getRoomType().equalsIgnoreCase("TRIPLE")) {
                        if (!room.isRoomStatus() && !room.isSpecialRoom()) {
                            roomRecommendations.add(room);
                            roomsFound2 = true;
                        }
                    }
                }
            }

            if (roomType.equalsIgnoreCase("TRIPLE")) {
                for (Room room : accommodations) {
                    if (room.getRoomType().equalsIgnoreCase("TRIPLE")) {
                        if (!room.isRoomStatus() && !room.isSpecialRoom()) {
                            roomRecommendations.add(room);
                            roomsFound2 = true;
                        }
                    }
                }
            }

            if (!roomsFound2) {
                System.out.println("No recommended rooms available.");
            }
        }
    }
    public boolean specialRoomRecommendations(String roomType) {
        boolean isValid = false;
        boolean roomsAvailable = false;
```

```
        // Clear previous recommendations
        roomRecommendations.clear();

        for (Room room : accommodations) {
            if (roomType.equalsIgnoreCase(room.getRoomType())) {
                if (room.isSpecialRoom() && !room.isRoomStatus()) {
                    roomRecommendations.add(room);
                    roomsAvailable = true;
                    isValid = true;
                }
            }
        }

        if (!roomsAvailable) {
            for (Room room : accommodations) {
                if (room.isSpecialRoom() && !room.isRoomStatus()) {
                    System.out.println("The Required Room Type is Not Available! We
Recommend the Following Rooms!");
                    isValid = true;
                    roomRecommendations.add(room);
                }
            }
        }

        if (!roomsAvailable && roomRecommendations.isEmpty()) {
            System.out.println("Sorry, We are completely Out of Special Rooms!");
            isValid = false;
        }

        return isValid;
    }

    public static int noOfRoomsBooked;

    //VALIDATIONS
    public boolean validateRoomType(String roomType){
        boolean valid = false;
        try {
            if(roomType.equalsIgnoreCase("SINGLE") || roomType.equalsIgnoreCase("DOUBLE")
|| roomType.equalsIgnoreCase("TRIPLE")){
                valid = true;
            }
            else {
                throw new IllegalArgumentException("Enter Valid Input (SINGLE | DOUBLE |
TRIPLE) ");
            }
        }catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
        }
        return valid;
    }
    public boolean validateRoomCategory(String roomCategory){
        boolean valid = false;
        try {
            if(roomCategory.equalsIgnoreCase("STANDARD") ||
roomCategory.equalsIgnoreCase("DELUXE") || roomCategory.equalsIgnoreCase("PREMIUM")){
                valid = true;
            }
            else {
                throw new IllegalArgumentException("Enter Valid Input (STANDARD | DELUXE
| PREMIUM) ");
            }
```

```
        }catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
        }
        return valid;

        //        String roomCategory;
//        do{
//            System.out.print("Enter Room Category: ");
//            roomCategory = scan.nextLine();
//        }while (!entrance.validateRoomCategory(roomCategory));
    }
    public boolean validateRoomID(int roomID){
        boolean valid = false;
        try {
            for (Room room: accommodations){
                if(roomID == room.getRoomID()){
                    valid = true;
                    break;
                }
            }
            if(!valid){
                throw new IndexOutOfBoundsException("Invalid Room ID");
            }
        }catch (IndexOutOfBoundsException e){
            System.out.println(e.getMessage());
        }
        return valid;
    } //User Enters an invalid Room ID
    public boolean validateRoomRecommendations(int roomID){
        Room room = selectRoomByRoomID(roomID);
        boolean isValid = false;
        try {
            if(roomRecommendations.contains(room)){
                isValid = true;
            }else {
                throw new IndexOutOfBoundsException("Choose a Recommended Room!");
            }
        }catch (IndexOutOfBoundsException e){
            System.out.println(e.getMessage());
        }
        return isValid;} //User Tries to Book a Room Out of the Recommended Range
    public boolean validateNoOfGuestVsRoomTypeForNormalRoom(String roomType, int
noOfPeople){
        boolean isValid = true;
        //If Number of People are greater than 2 && Room Type selected is Single
        boolean a = roomType.equalsIgnoreCase("Single") && (noOfPeople>2 &&
noOfPeople<=4);
        //if Number of People are greater than 4 && Room Type Selected is Single or
Double
        boolean b = (roomType.equalsIgnoreCase("Single") ||
roomType.equalsIgnoreCase("Double")) && (noOfPeople>4 && noOfPeople<=6);

        try{
            if(a){
                isValid = false;
                throw new IndexOutOfBoundsException(noOfPeople + " People Cannot Book a
"+roomType.toUpperCase() +" Room");
            } else if (b) {
                isValid = false;
                throw new IndexOutOfBoundsException(noOfPeople + " People Cannot Book a
"+roomType.toUpperCase() +" Room");
            }
        }
```

```
        catch (IndexOutOfBoundsException e){
            System.out.println(e.getMessage());
        }
        return isValid;
    }
    public boolean validateSpecialRoomType(String roomType) {
        boolean isValid = false;
        try{
            if (roomType.equalsIgnoreCase("DOUBLE") ||
roomType.equalsIgnoreCase("TRIPLE")) {
                isValid = true;
            } else if (roomType.equalsIgnoreCase("SINGLE")) {
                throw new IllegalArgumentException("There are No Special Single Rooms!");
            } else {
                throw new IllegalArgumentException("Enter Valid Room Type");
            }
        }catch (IllegalArgumentException e){
            System.out.println(e.getMessage());
        }
        return isValid;
    }
    public boolean validateSpecialRoomVsNoOfGuest(String roomType, int noOfPeople){
        boolean isValid = true;

        try{
            //Can't be less than 2 people
            if(noOfPeople<2){
                isValid = false;
                throw new IndexOutOfBoundsException("At Least 2 People Should Stay in a
Special Room!");
            }

            //if a double room cant be more than 4 people
            if(roomType.equalsIgnoreCase("double")&& noOfPeople>4){
                isValid = false;
                throw new IndexOutOfBoundsException(noOfPeople+" People Cannot Stay in a
Double Room");
            }

            //if a triple room can't be more than 6 people
            if(roomType.equalsIgnoreCase("triple") && noOfPeople>6){
                isValid = false;
                throw new IndexOutOfBoundsException(noOfPeople + " People cannot stay in
a Triple Room");
            }
        } catch (IndexOutOfBoundsException e){
            System.out.println(e.getMessage());
        }

        return isValid;
    }

}
```

Screenshots:

Booking a Normal Room:

```
Hotel Management System Menu:
1. Display Rooms
2. Check In Rooms
3. Check Out Rooms
4. Display Statistical Report
5. Exit
Enter your choice (1-5): 2
Checking In Rooms

Check In Submenu:
    1. Book a Normal Room
    2. Book a Special Room
    3. Back to Main Menu
Enter your choice (1-3): 1
Enter Guest Name: Nidula
Enter Number of Days of Stay: 2
Enter Number of People Staying: 2
Enter Room Type: single
Enter Room Category: standard
Room ID: 101    Guest ID: 0     Room Status: Available      Room Category: STANDARD     Room Type: SINGLE       Room Cost:230.0     Days Staying: 0     Total Price: 0.0        No Of People: 0     Name: null
Room ID: 102    Guest ID: 0     Room Status: Available      Room Category: STANDARD     Room Type: SINGLE       Room Cost:230.0     Days Staying: 0     Total Price: 0.0        No Of People: 0     Name: null
Room ID: 103    Guest ID: 0     Room Status: Available      Room Category: STANDARD     Room Type: SINGLE       Room Cost:230.0     Days Staying: 0     Total Price: 0.0        No Of People: 0     Name: null
Room ID: 104    Guest ID: 0     Room Status: Available      Room Category: STANDARD     Room Type: SINGLE       Room Cost:230.0     Days Staying: 0     Total Price: 0.0        No Of People: 0     Name: null
Enter Room ID: 101

------------RECEIPT-------------
Guest Name: Nidula
Room Category: STANDARD
Room Type: SINGLE
Room ID: 101
Room Price Per Day: 230.0
Number of Days Staying: 2
Total Cost of the Room: 460.0
Guest ID: 201
Room Description: Standard Room With Basic Amenities
```

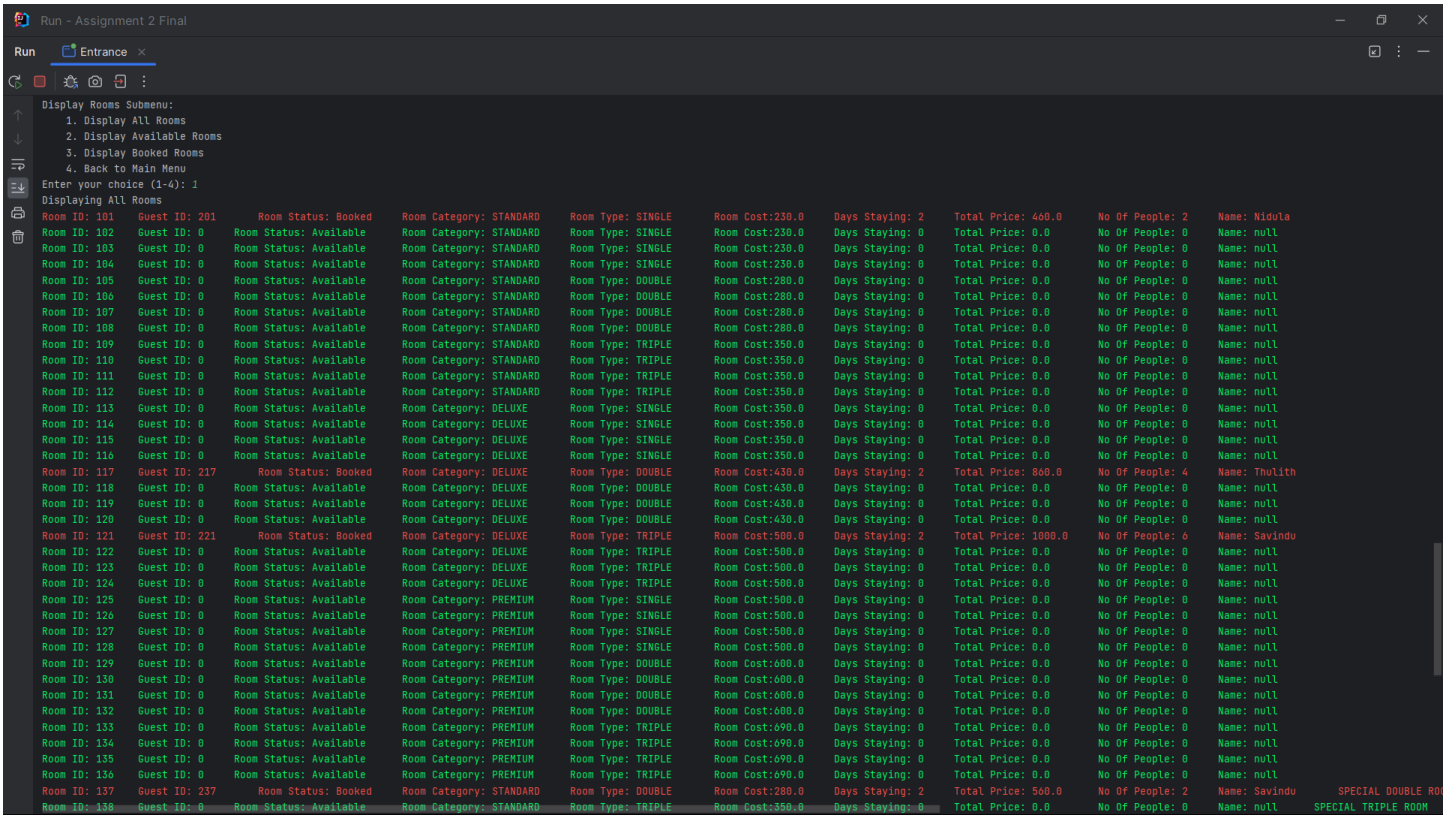
Booking a Special Room

```
Hotel Management System Menu:
1. Display Rooms
2. Check In Rooms
3. Check Out Rooms
4. Display Statistical Report
5. Exit
Enter your choice (1-5): 2
Checking In Rooms

Check In Submenu:
    1. Book a Normal Room
    2. Book a Special Room
    3. Back to Main Menu
Enter your choice (1-3): 2
Enter Room Type: double
Enter Number of People Staying: 2
Room ID: 137    Guest ID: 0     Room Status: Available      Room Category: STANDARD     Room Type: DOUBLE       Room Cost:280.0     Days Staying: 0     Total Price: 0.0        No Of People: 0     Name: null      SPECIAL DOUBLE ROOM
Enter Room ID to Book Room: 137
Enter Guest Name: Savindu
Enter Number of Days Staying: 2

------------RECEIPT-------------
Guest Name: Savindu
Room Category: STANDARD SPECIAL
Room Type: DOUBLE
Room ID: 137
Room Price Per Day: 280.0
Number of Days Staying: 2
Total Cost of the Room: 560.0
Guest ID: 237
Room Description: Special Double Room With Fitted Ramps, Emergency Calling Facilities & Close to Beach
```

Display All Rooms



Room Validations

More than 2 People Trying to buy a Single Room

```
Hotel Management System Menu:
1. Display Rooms
2. Check In Rooms
3. Check Out Rooms
4. Display Statistical Report
5. Exit
Enter your choice (1-5): 2
Checking In Rooms

Check In Submenu:
    1. Book a Normal Room
    2. Book a Special Room
    3. Back to Main Menu
Enter your choice (1-3): 1
Enter Guest Name: Nidula
Enter Number of Days of Stay: 2
Enter Number of People Staying: 4
Enter Room Type: single
4 People Cannot Book a SINGLE Room
Enter Room Type: double
Enter Room Category:
```

More than 4 People Trying to buy a Double Room

```
Hotel Management System Menu:
1. Display Rooms
2. Check In Rooms
3. Check Out Rooms
4. Display Statistical Report
5. Exit
Enter your choice (1-5): 2
Checking In Rooms

Check In Submenu:
    1. Book a Normal Room
    2. Book a Special Room
    3. Back to Main Menu
Enter your choice (1-3): 1
Enter Guest Name: Nidula
Enter Number of Days of Stay: 2
Enter Number of People Staying: 5
Enter Room Type: double
5 People Cannot Book a DOUBLE Room
Enter Room Type: triple
Enter Room Category: premium
```

More than 6 People Trying to but a Room

```
Hotel Management System Menu:
1. Display Rooms
2. Check In Rooms
3. Check Out Rooms
4. Display Statistical Report
5. Exit
Enter your choice (1-5): 2
Checking In Rooms

Check In Submenu:
    1. Book a Normal Room
    2. Book a Special Room
    3. Back to Main Menu
Enter your choice (1-3): 1
Enter Guest Name: Nidula
Enter Number of Days of Stay: 2
Enter Number of People Staying: 7
Maximum Accommodation is 6 People per Room
Enter Number of People Staying:
```

User Enters an Invalid Argument for Room Type

```
Check In Submenu:
    1. Book a Normal Room
    2. Book a Special Room
    3. Back to Main Menu
Enter your choice (1-3): 1
Enter Guest Name: Nidula
Enter Number of Days of Stay: 2
Enter Number of People Staying: 2
Enter Room Type: deluxe
Enter Valid Input (SINGLE | DOUBLE | TRIPLE)
```

User Enters an Invalid Argument for Room Category

```
Enter Room Type: single
Enter Room Category: single
Enter Valid Input (STANDARD | DELUXE | PREMIUM)
Enter Room Category:
```

User Trying to Check in a Room that is Out of Scope (Not the Required Type)

```
Enter Room Type: triple
Enter Room Category: deluxe
Room ID: 122    Guest ID: 0     Room Status: Available      Room Category: DELUXE       Room Type: TRIPLE       Room Cost:500.0     Days Staying: 0     Total Price: 0.0        No Of People: 0     Name: null
Room ID: 123    Guest ID: 0     Room Status: Available      Room Category: DELUXE       Room Type: TRIPLE       Room Cost:500.0     Days Staying: 0     Total Price: 0.0        No Of People: 0     Name: null
Room ID: 124    Guest ID: 0     Room Status: Available      Room Category: DELUXE       Room Type: TRIPLE       Room Cost:500.0     Days Staying: 0     Total Price: 0.0        No Of People: 0     Name: null
Enter Room ID: 125
Choose a Recommended Room!
Enter Room ID:
```

User Trying to Check in a Room that is Already Booked

```
Enter Number of Days of Stay: 2
Enter Number of People Staying: 2
Enter Room Type: triple
Enter Room Category: deluxe
Room ID: 122    Guest ID: 0     Room Status: Available      Room Category: DELUXE       Room Type: TRIPLE       Room Cost:500.0     Days Staying: 0     Total Price: 0.0        No Of People: 0     Name: null
Room ID: 123    Guest ID: 0     Room Status: Available      Room Category: DELUXE       Room Type: TRIPLE       Room Cost:500.0     Days Staying: 0     Total Price: 0.0        No Of People: 0     Name: null
Enter Room ID: 124
This room is Already Booked!
Enter Room ID:
```

User Trying to Check Out a Room that isn't Already Booked

```
Hotel Management System Menu:
1. Display Rooms
2. Check In Rooms
3. Check Out Rooms
4. Display Statistical Report
5. Exit
Enter your choice (1-5): 3
Room ID: 101    Guest ID: 201       Room Status: Booked     Room Category: STANDARD     Room Type: SINGLE       Room Cost:230.0     Days Staying: 2     Total Price: 460.0      No Of People: 2     Name: Nidula
Room ID: 102    Guest ID: 202       Room Status: Booked     Room Category: STANDARD     Room Type: SINGLE       Room Cost:230.0     Days Staying: 2     Total Price: 460.0      No Of People: 2     Name: Nidula
Room ID: 105    Guest ID: 205       Room Status: Booked     Room Category: STANDARD     Room Type: DOUBLE       Room Cost:280.0     Days Staying: 2     Total Price: 560.0      No Of People: 4     Name: Nidula
Room ID: 117    Guest ID: 217       Room Status: Booked     Room Category: DELUXE       Room Type: DOUBLE       Room Cost:430.0     Days Staying: 2     Total Price: 860.0      No Of People: 4     Name: Thulith
Room ID: 121    Guest ID: 221       Room Status: Booked     Room Category: DELUXE       Room Type: TRIPLE       Room Cost:500.0     Days Staying: 2     Total Price: 1000.0     No Of People: 6     Name: Savindu
Room ID: 122    Guest ID: 222       Room Status: Booked     Room Category: DELUXE       Room Type: TRIPLE       Room Cost:500.0     Days Staying: 2     Total Price: 1000.0     No Of People: 2     Name: Nidula
Room ID: 124    Guest ID: 224       Room Status: Booked     Room Category: DELUXE       Room Type: TRIPLE       Room Cost:500.0     Days Staying: 2     Total Price: 1000.0     No Of People: 2     Name: Nidula
Room ID: 125    Guest ID: 225       Room Status: Booked     Room Category: PREMIUM      Room Type: SINGLE       Room Cost:500.0     Days Staying: 2     Total Price: 1000.0     No Of People: 2     Name: Nidula
Room ID: 129    Guest ID: 229       Room Status: Booked     Room Category: PREMIUM      Room Type: DOUBLE       Room Cost:600.0     Days Staying: 2     Total Price: 1200.0     No Of People: 4     Name: Nidula
Room ID: 133    Guest ID: 233       Room Status: Booked     Room Category: PREMIUM      Room Type: TRIPLE       Room Cost:690.0     Days Staying: 2     Total Price: 1380.0     No Of People: 5     Name: Nidula
Room ID: 134    Guest ID: 234       Room Status: Booked     Room Category: PREMIUM      Room Type: TRIPLE       Room Cost:690.0     Days Staying: 2     Total Price: 1380.0     No Of People: 6     Name: Nidula
Room ID: 137    Guest ID: 237       Room Status: Booked     Room Category: STANDARD     Room Type: DOUBLE       Room Cost:280.0     Days Staying: 2     Total Price: 560.0      No Of People: 2     Name: Savindu       SPECIAL DOUBLE
Enter Room ID: 120
This Room is Not Booked!
Enter Room ID:
```

Room Recommendations if all the Rooms of Required Type are booked Out

```
Check In Submenu:
    1. Book a Normal Room
    2. Book a Special Room
    3. Back to Main Menu
Enter your choice (1-3): 1
Enter Guest Name: Nidula
Enter Number of Days of Stay: 2
Enter Number of People Staying: 2
Enter Room Type: triple
Enter Room Category: deluxe
This Type of Room is Not Available at The Moment, Consider Following Recommended Rooms!
Room ID: 109    Guest ID: 0     Room Status: Available      Room Category: STANDARD     Room Type: TRIPLE       Room Cost:350.0     Days Staying: 0     Total Price: 0.0        No Of People: 0     Name: null
Room ID: 110    Guest ID: 0     Room Status: Available      Room Category: STANDARD     Room Type: TRIPLE       Room Cost:350.0     Days Staying: 0     Total Price: 0.0        No Of People: 0     Name: null
Room ID: 111    Guest ID: 0     Room Status: Available      Room Category: STANDARD     Room Type: TRIPLE       Room Cost:350.0     Days Staying: 0     Total Price: 0.0        No Of People: 0     Name: null
Room ID: 112    Guest ID: 0     Room Status: Available      Room Category: STANDARD     Room Type: TRIPLE       Room Cost:350.0     Days Staying: 0     Total Price: 0.0        No Of People: 0     Name: null
Room ID: 134    Guest ID: 0     Room Status: Available      Room Category: PREMIUM      Room Type: TRIPLE       Room Cost:690.0     Days Staying: 0     Total Price: 0.0        No Of People: 0     Name: null
Room ID: 135    Guest ID: 0     Room Status: Available      Room Category: PREMIUM      Room Type: TRIPLE       Room Cost:690.0     Days Staying: 0     Total Price: 0.0        No Of People: 0     Name: null
Room ID: 136    Guest ID: 0     Room Status: Available      Room Category: PREMIUM      Room Type: TRIPLE       Room Cost:690.0     Days Staying: 0     Total Price: 0.0        No Of People: 0     Name: null
Enter Room ID:
```

Database.txt File

```
Hotel Management System Menu:
1. Display Rooms
2. Check In Rooms
3. Check Out Rooms
4. Display Statistical Report
5. Exit
Enter your choice (1-5): 5
Exiting the program. Goodbye!
Data written to the file successfully.
```

```
101 | 201 | 2 | 460.0 | 2 | Nidula
102 | 202 | 2 | 460.0 | 2 | Nidula
105 | 205 | 2 | 560.0 | 4 | Nidula
109 | 209 | 2 | 700.0 | 2 | Nidula
117 | 217 | 2 | 860.0 | 4 | Thulith
121 | 221 | 2 | 1000.0 | 6 | Savindu
122 | 222 | 2 | 1000.0 | 2 | Nidula
123 | 223 | 2 | 1000.0 | 2 | Nidula
124 | 224 | 2 | 1000.0 | 2 | Nidula
125 | 225 | 2 | 1000.0 | 2 | Nidula
129 | 229 | 2 | 1200.0 | 4 | Nidula
133 | 233 | 2 | 1380.0 | 5 | Nidula
137 | 237 | 2 | 560.0 | 2 | Savindu
```

Object Persistence