

Complex Computing Problem (CCP)

|  |  |
| --- | --- |
| **Name** | **Reg.ID** |
| **Hammad Saeed** | **63146** |

|  |  |
| --- | --- |
| **Course Title** | **Class Day & Timings** |
| Software Construction & Development | Thursday (8:30 AM**)** |

**Question 1:**

**Implementing cancelReservation(int) in GuestHouse:**

Below is the full implementation of the GuestHouse class’s cancelReservation method. This version incorporates defensive programming, comments, clean code practices, and proper domain modeling.

**GuestHouse.java**

package com.company; import java.util.HashMap;

public class GuestHouse {

private HashMap<Integer, Reservation> allReservations;

public GuestHouse() { allReservations = new HashMap();

}

*/\*\**

* *Cancels the specified reservation by:*
* *- Removing it from guest house's reservations*
* *- Removing it from the reserverPayer's reservations*
* *- Deallocating the room*
* ***@param*** *reservationNumber The number of the reservation to cancel*
* ***@return*** *Feedback about the cancelled reservation*

*\*/*

public String cancelReservation(int reservationNumber) {

// Defensive: Validate reservation number if (reservationNumber <= 0) {

return "Invalid reservation number. Please enter a valid positive number.";

}

// Defensive: Check if reservation exists

if (!allReservations.containsKey(reservationNumber)) { return "No reservation found with the given number.";

}

Reservation reservation = allReservations.get(reservationNumber);

// Remove from GuestHouse record allReservations.remove(reservationNumber);

// Remove from ReserverPayer's record

ReserverPayer reserver = reservation.getReserverPayer(); if (reserver != null) {

reserver.removeReservation(reservationNumber);

}

// Deallocate room

Room room = reservation.getRoom(); if (room != null) {

room.deallocateRoom(reservation);

}

return "Reservation #" + reservationNumber + " has been successfully cancelled.";

}

// Optional: Method to add reservation for testing/demo

public void addReservation(int number, Reservation reservation) { allReservations.put(number, reservation);

}

}

**Supporting Classes:**

**ReserverPayer.java**

package com.company; import java.util.HashMap;

public class ReserverPayer {

private HashMap<Integer, Reservation> myReservations = new HashMap();

public void removeReservation(int reservationNumber) { if (reservationNumber > 0 &&

myReservations.containsKey(reservationNumber)) { myReservations.remove(reservationNumber);

}

}

public void addReservation(int reservationNumber, Reservation reservation) {

myReservations.put(reservationNumber, reservation);

}

}

**Room.java**

package com.company;

import java.util.ArrayList;

public class Room {

private ArrayList<Reservation> roomReservations = new ArrayList();

public void deallocateRoom(Reservation reservation) { roomReservations.remove(reservation);

}

public void allocateRoom(Reservation reservation) { roomReservations.add(reservation);

}

}

**Reservation.java**

package com.company;

public class Reservation {

private int reservationNumber; private ReserverPayer reserverPayer; private Room room;

public Reservation(int reservationNumber, ReserverPayer reserverPayer, Room room) {

this.reservationNumber = reservationNumber; this.reserverPayer = reserverPayer; this.room = room;

}

public ReserverPayer getReserverPayer() { return reserverPayer;

}

public Room getRoom() { return room;

}

public int getReservationNumber() { return reservationNumber;

}

}

**Main.java:** (main class containing main method)

package com.company;

import com.company.GuestHouse; import com.company.Reservation; import com.company.Room;

import com.company.ReserverPayer;

public class Main {

public static void main(String[] args) {

// Create GuestHouse instance

GuestHouse guestHouse = new GuestHouse();

// Create supporting objects

ReserverPayer reserver = new ReserverPayer(); Room room = new Room();

// Create a Reservation with ID 1001

Reservation reservation = new Reservation(1001, reserver, room);

// Add the reservation to all relevant places guestHouse.addReservation(1001, reservation); // Add to GuestHouse reserver.addReservation(1001, reservation); // Add to

ReserverPayer

room.allocateRoom(reservation); // Add to Room

// Cancel the reservation

String result = guestHouse.cancelReservation(1001);

// Print the result System.*out*.println(result);

}

}

**OUTPUT:**

**Question 2:**

**Refactor GUI Code to Remove Code Smell:**

**Code Smell Identified:**

**Long Method + Multiple Responsibilities** in confirmReservationButtonActionPerformed:

* Performs multiple unrelated checks (date, name, address, credit card)
* UI logic tightly coupled with business logic

**Refactoring Applied:**

* **Extract Method** – each validation check moved to a separate method.
* Improves **readability**, **testability**, and aligns with **SRP (Single Responsibility Principle)**.

**Refactored Java Code:**

private void confirmReservationButtonActionPerformed(java.awt.event.ActionEvent evt) {

if (!isDateValid()) return; if (!isNameValid()) return;

if (!isAddressValid()) return;

if (!isCreditCardValid()) return;

String reservationMessage = guestHouseChainFacade.makeReservation( guestHouseName, startDate, endDate, roomType,

reserverName, multiLineAddress, creditCardNo

);

reservationFeedbackField.setText(reservationMessage);

if (reservationMessage.startsWith("Confirmed")) { setEnabledMakeReservationPanel(false);

}

}

private boolean isDateValid() {

if (startDate == null || endDate == null) { reservationFeedbackField.setText("Enter valid dates"); return false;

}

return true;

}

private boolean isNameValid() {

if (reserverName.trim().isEmpty()) { reservationFeedbackField.setText("Enter valid name"); customerNameField.requestFocus();

return false;

}

return true;

}

private boolean isAddressValid() {

if (address.trim().isEmpty()) { reservationFeedbackField.setText("Enter valid address"); customerAddressTextArea.requestFocus();

return false;

}

return true;

}

private boolean isCreditCardValid() {

if (existingAddresses == null &&

!GuestHouseFacade.isValidCreditCardNumberFormat(creditCardNo)) {

reservationFeedbackField.setText(GuestHouseFacade.creditCardNumberFormatErr or(creditCardNo));

creditCardNumberField.selectAll(); creditCardNumberField.requestFocus(); return false;

}

return true;

}