Student Accommodation Finder

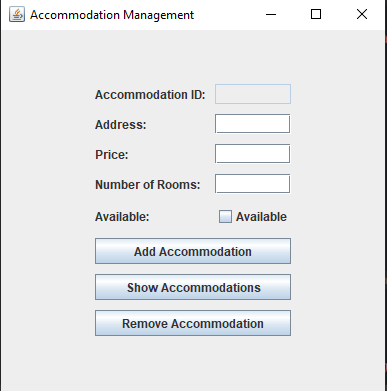
(DormNest)

50% of Code Implementation of the Student Accommodation Finder (DormNest)

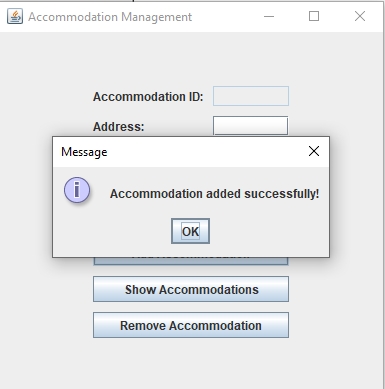
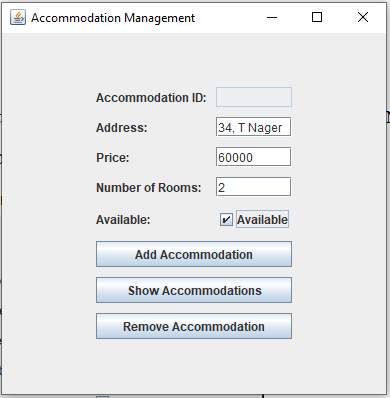
**Accommodation**

**To Adding Accommodation Details:**

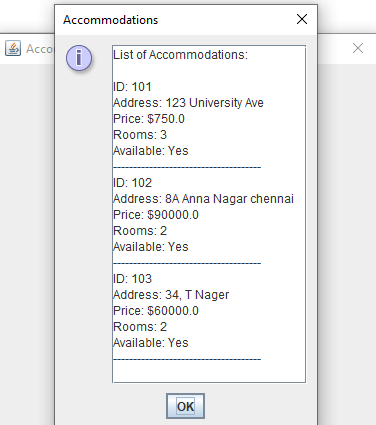
**Step 1:**



**Step 2:**

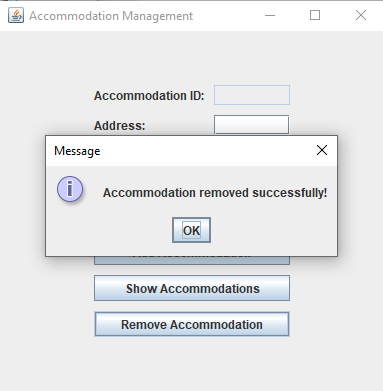
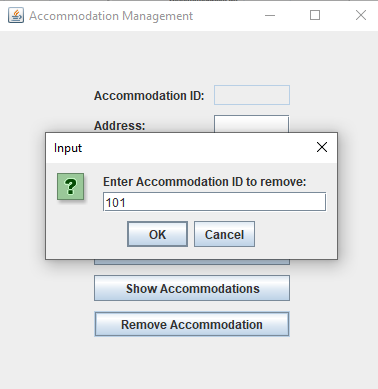


**Show Accommodations:**

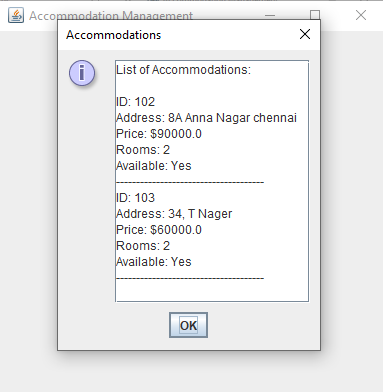


**To Remove Accommodation:**

**Step 1:**



**Final List of Accommodation List after Removing Accommodation:**



**Code:**

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.util.ArrayList;  
  
public class AccommodationGUI {  
 // ArrayList to store accommodation objects  
 private final ArrayList<Accommodation> accommodations;  
  
 public static void main(String[] args) {  
 // Create the GUI instance  
 SwingUtilities.*invokeLater*(AccommodationGUI::new);  
 }  
  
 public AccommodationGUI() {  
 // Initialize accommodation list  
 accommodations = new ArrayList<>();  
 accommodations.add(new Accommodation(101, "123 University Ave", 750.0, 3, true)); // Sample data  
  
 // Create the main JFrame  
 JFrame frame = new JFrame("Accommodation Management");  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 frame.setSize(400, 400);  
 frame.setLayout(new GridBagLayout());  
 GridBagConstraints gbc = new GridBagConstraints();  
 gbc.insets = new Insets(5, 5, 5, 5); // Padding around components  
 gbc.fill = GridBagConstraints.*HORIZONTAL*;  
  
 // Labels and TextFields for Accommodation details  
 gbc.gridx = 0; // Column  
 gbc.gridy = 0; // Row  
 frame.add(new JLabel("Accommodation ID:"), gbc);  
  
 gbc.gridx = 1;  
 JTextField idField = new JTextField();  
 idField.setEditable(false);  
 frame.add(idField, gbc);  
  
 gbc.gridx = 0;  
 gbc.gridy++;  
 frame.add(new JLabel("Address:"), gbc);  
  
 gbc.gridx = 1;  
 JTextField addressField = new JTextField();  
 frame.add(addressField, gbc);  
  
 gbc.gridx = 0;  
 gbc.gridy++;  
 frame.add(new JLabel("Price:"), gbc);  
  
 gbc.gridx = 1;  
 JTextField priceField = new JTextField();  
 frame.add(priceField, gbc);  
  
 gbc.gridx = 0;  
 gbc.gridy++;  
 frame.add(new JLabel("Number of Rooms:"), gbc);  
  
 gbc.gridx = 1;  
 JTextField roomsField = new JTextField();  
 frame.add(roomsField, gbc);  
  
 gbc.gridx = 0;  
 gbc.gridy++;  
 frame.add(new JLabel("Available:"), gbc);  
  
 gbc.gridx = 1;  
 JCheckBox availableCheckbox = new JCheckBox("Available");  
 frame.add(availableCheckbox, gbc);  
  
 // Buttons for actions  
 gbc.gridx = 0;  
 gbc.gridy++;  
 gbc.gridwidth = 2; // Span across both columns  
 JButton addButton = new JButton("Add Accommodation");  
 frame.add(addButton, gbc);  
  
 gbc.gridy++;  
 JButton showButton = new JButton("Show Accommodations");  
 frame.add(showButton, gbc);  
  
 gbc.gridy++;  
 JButton removeButton = new JButton("Remove Accommodation");  
 frame.add(removeButton, gbc);  
  
 // Action listeners using lambda expressions  
 addButton.addActionListener(e -> {  
 // Validate user input  
 String address = addressField.getText();  
 String priceText = priceField.getText();  
 String roomsText = roomsField.getText();  
  
 if (address.isEmpty() || priceText.isEmpty() || roomsText.isEmpty()) {  
 JOptionPane.*showMessageDialog*(frame, "Please fill in all fields!", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 return;  
 }  
  
 try {  
 double price = Double.*parseDouble*(priceText);  
 int rooms = Integer.*parseInt*(roomsText);  
 int id = accommodations.size() + 101; // Generate a new ID based on current size  
 boolean available = availableCheckbox.isSelected();  
  
 // Create a new Accommodation and add it to the list  
 Accommodation newAccommodation = new Accommodation(id, address, price, rooms, available);  
 accommodations.add(newAccommodation);  
  
 // Clear the input fields  
 addressField.setText("");  
 priceField.setText("");  
 roomsField.setText("");  
 availableCheckbox.setSelected(false);  
  
 JOptionPane.*showMessageDialog*(frame, "Accommodation added successfully!");  
 } catch (NumberFormatException ex) {  
 JOptionPane.*showMessageDialog*(frame, "Invalid number format!", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 });  
  
 showButton.addActionListener(e -> displayAccommodations());  
  
 removeButton.addActionListener(e -> {  
 String removeIdText = JOptionPane.*showInputDialog*(frame, "Enter Accommodation ID to remove:");  
  
 if (removeIdText == null || removeIdText.isEmpty()) {  
 JOptionPane.*showMessageDialog*(frame, "Please enter an ID to remove!", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 return;  
 }  
  
 try {  
 int removeId = Integer.*parseInt*(removeIdText);  
 boolean removed = removeAccommodationById(removeId);  
  
 if (removed) {  
 JOptionPane.*showMessageDialog*(frame, "Accommodation removed successfully!");  
 } else {  
 JOptionPane.*showMessageDialog*(frame, "Accommodation ID not found!", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 } catch (NumberFormatException ex) {  
 JOptionPane.*showMessageDialog*(frame, "Invalid ID format!", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 });  
  
 // Finalize frame settings  
 frame.setVisible(true);  
 frame.setLocationRelativeTo(null); // Center the frame on the screen  
 }  
  
 private void displayAccommodations() {  
 StringBuilder accommodationDetails = new StringBuilder();  
 accommodationDetails.append("List of Accommodations:\n\n");  
  
 for (Accommodation acc : accommodations) {  
 accommodationDetails.append("ID: ").append(acc.getAccommodationID()).append("\n");  
 accommodationDetails.append("Address: ").append(acc.getAddress()).append("\n");  
 accommodationDetails.append("Price: $").append(acc.getPrice()).append("\n");  
 accommodationDetails.append("Rooms: ").append(acc.getNumOfRooms()).append("\n");  
 accommodationDetails.append("Available: ").append(acc.isAvailStatus() ? "Yes" : "No").append("\n");  
 accommodationDetails.append("-------------------------------------\n");  
 }  
  
 JTextArea textArea = new JTextArea(accommodationDetails.toString());  
 textArea.setEditable(false);  
 JOptionPane.*showMessageDialog*(null, new JScrollPane(textArea), "Accommodations", JOptionPane.*INFORMATION\_MESSAGE*);  
 }  
  
 private boolean removeAccommodationById(int id) {  
 for (Accommodation acc : accommodations) {  
 if (acc.getAccommodationID() == id) {  
 accommodations.remove(acc);  
 return true; // Return true if removal was successful  
 }  
 }  
 return false; // Return false if no accommodation with that ID was found  
 }  
}  
  
// Accommodation class  
class Accommodation {  
 private final int accommodationID; // Make final if not reassigned  
 private String address;  
 private double price;  
 private int numOfRooms; // Updated naming to follow convention  
 private boolean availStatus;  
  
 public Accommodation(int accommodationID, String address, double price, int numOfRooms, boolean availStatus) {  
 this.accommodationID = accommodationID;  
 this.address = address;  
 this.price = price;  
 this.numOfRooms = numOfRooms;  
 this.availStatus = availStatus;  
 }  
  
 // Getters and Setters  
 public int getAccommodationID() {  
 return accommodationID;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
  
 public double getPrice() {  
 return price;  
 }  
  
 public void setPrice(double price) {  
 this.price = price;  
 }  
  
 public int getNumOfRooms() { // Updated naming  
 return numOfRooms;  
 }  
  
 public void setNumOfRooms(int numOfRooms) { // Updated naming  
 this.numOfRooms = numOfRooms;  
 }  
  
 public boolean isAvailStatus() {  
 return availStatus;  
 }  
  
 public void setAvailStatus(boolean availStatus) {  
 this.availStatus = availStatus;  
 }  
}