

**Department of Computer Science**

Data Structures and Algorithms Group Assignment

Due Date:14 October 2024

Group Members

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Student Name | Student Number | Mode |
| 1 | Jean-Paul Seraun(TL) | 220076669 | FT |
| 2 | Yvonne Andima | 223009350 | FT |
| 3 | Dago Tsuses | 223014648 | FT |
| 4 | Eila Haimbili | 221102426 | FT |
| 5 | Elkana KamaJ | 223093971 | FT |

Table of Contents

# Section A……………………………………………………………………………………………………..……4

Pseudocode………………………………………………………………………………………………………………………………………...4

Flowchart………………………………………………………………………………………………………………………………………….…6

# Section B……………………………………………………………………………………………………..……7

Functionality, Services, and Requirements (FSR)**………..………………………………………………………………….....**7

Code Breakdown………………………………………………………………………………………………………………………………….8

Method to Create GUI..………………………………………………………………………………………………………………………10 Button Creation………………………………………………………………………………………………………………………...12

Main Method to Run the Application……………………………………………………………………………………….15

Buttons……………………………………………………..……………………………………………………………………………….17

Section A

Pseudocode for Phonebook Applica7on

START

Initialize an empty list called contacts

Create the GUI with the following components:

* Text fields for name, phone, address, email
* Combo box for gender selection (Male, Female)
* Text area to display contacts
* Buttons: Insert, Search, Delete, Update, Display All, Sort

Display the GUI

WHILE the application is running:

IF Insert button is clicked:

Get name, phone, address, gender, and email from input fields

IF name and phone are not empty AND gender is valid (Male/Female):

Create a new Contact object with the input values

Add the Contact to the contacts list

Display "Contact added" message in the text area

Clear all input fields ELSE:

Show error message "Name and Phone cannot be empty, and Gender must be Male or Female"

IF Search button is clicked:

Prompt the user to enter a name to search

Search the contacts list for a Contact whose name matches (case-insensitive) IF a matching contact is found:

Display the contact's details in the text area ELSE:

Display "Contact not found" message in the text area

IF Delete button is clicked:

Prompt the user to enter a name to delete

Search and remove any contact from the contacts list whose name matches (caseinsensitive)

IF a contact is removed:

Display "Contact deleted" message in the text area

ELSE:

Display "Contact not found" message in the text area

IF Update button is clicked:

Prompt the user to enter a name to update

Search the contacts list for a Contact whose name matches (case-insensitive) IF a matching contact is found:

Prompt the user for new phone, address, gender, and email

Update the contact's details with the new values Display "Contact updated" message in the text area

ELSE:

Display "Contact not found" message in the text area

IF Display All button is clicked:

Clear the text area

FOR each contact in the contacts list:

Display the contact's details in the text area

IF Sort button is clicked:

Sort the contacts list alphabetically by name

Display "Contacts sorted" message in the text area

END WHILE

END



**Section B**

Functionality, Services, and Requirements (FSR) for the Phonebook Application

|  |  |  |  |
| --- | --- | --- | --- |
| **S No.** |  | **Module Name** | **Descrip3on/Role** |
|  | 1 | Contact Class | Represents a contact with name, phone number, address, gender, and email attributes. |
|  | 2 | Phonebook Class | Manages the list of contacts, providing methods for adding, searching, updating, and deleting contacts. |
|  | 2.1 | insertContact | Adds a new contact to the phonebook. |
|  | 2.2 | searchContact | Finds and returns a contact by name. |
|  | 2.3 | deleteContact | Removes a contact by name from the list. |
|  | 2.4 | updateContact | Updates the details of an existing contact. |
|  | 2.5 | sortContact | Sorts contacts alphabetically by name. |
|  | 3 | GUI (Main Window) | Provides a graphical interface for with the phonebook. |
|  | 3.1 | Buttons | Buttons for inserting, searching, displaying, deleting, updating, and sorting contacts. |
|  | 4 | Acton Listeners | Handles the button click events and triggers the corresponding phonebook actions. |

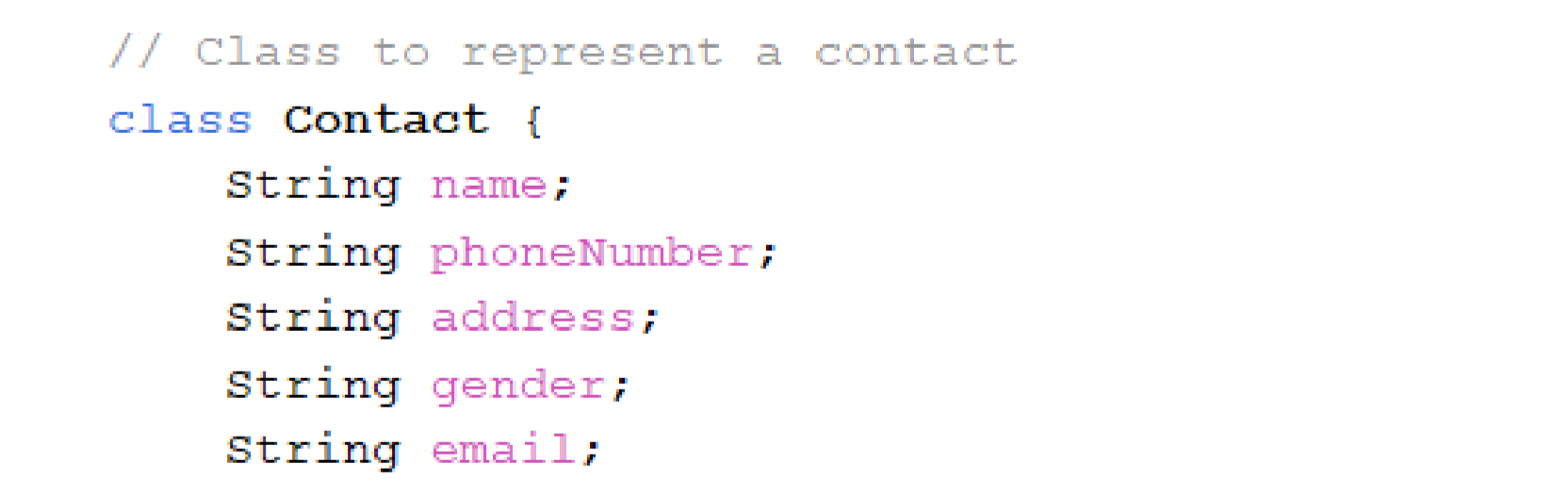
# The code

**Package declaration and import statements**



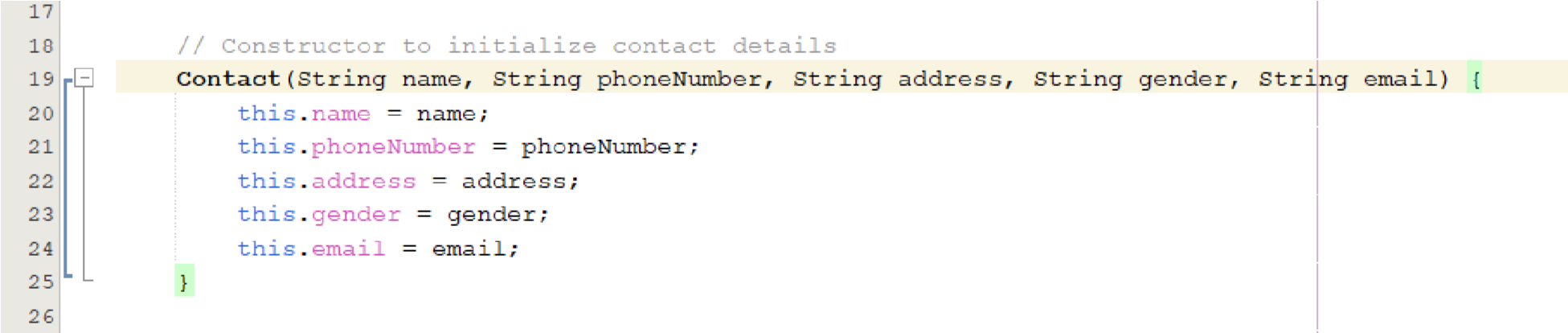
**Class Definition**

The contacts class represents a contact in the phonebook application



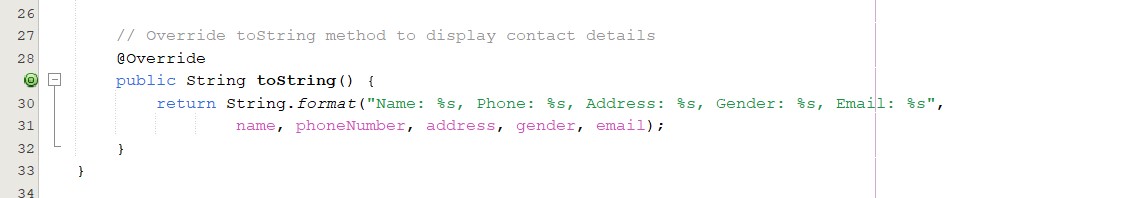
**Constructor**

The constructor initializes the contact details.



**Overridden toString Method**

The toString method returns a formatted string with the contact details

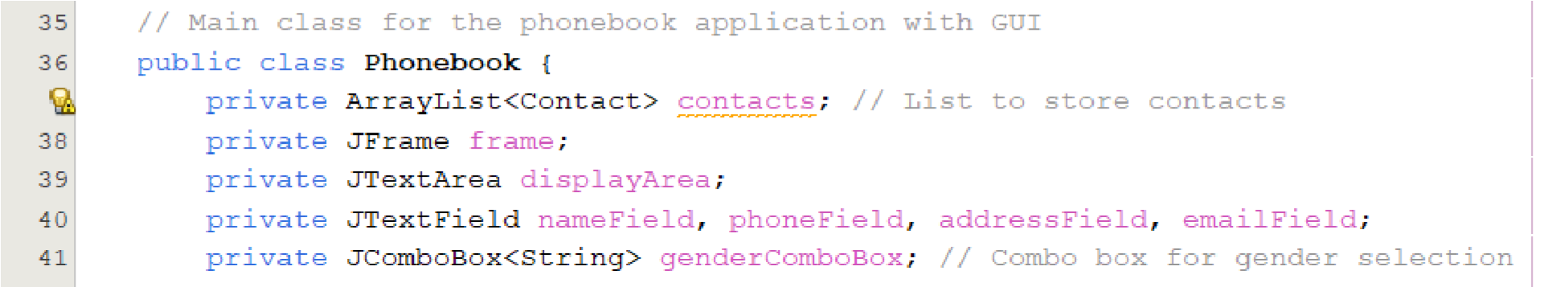


**Class Definition**

The PhoneBook class is the main class for the phonebook application, which includes a graphical user interface (GUI).

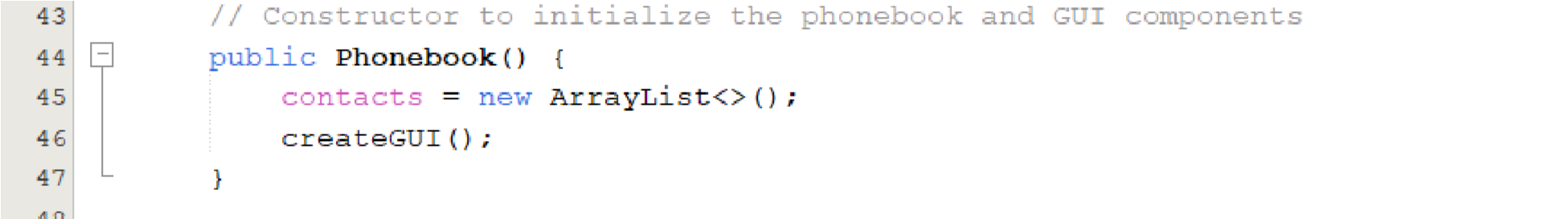
**Member Variables**

* private ArrayList<Contact> contacts: A list to store contact objects.
* private JFrame frame: The main frame for the GUI.
* private JTextArea displayArea: A text area to display contact information.
* private JTextField nameField, phoneField, addressField, emailField: Text fields for user input.
* private JComboBox<String> genderComboBox: A combo box for gender selection.



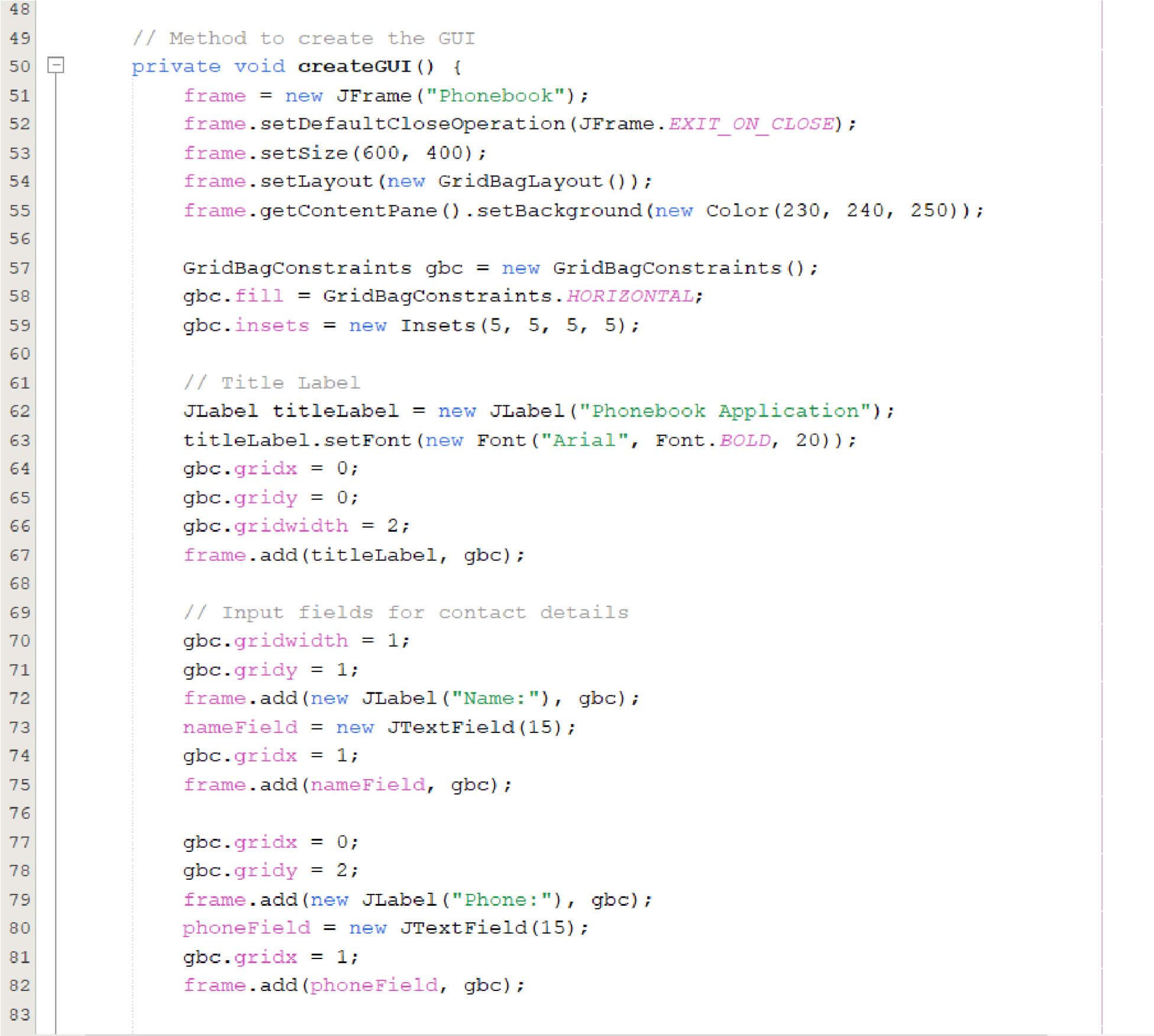
**Constructor**

The constructor initializes the phonebook and GUI components.



**Method to Create the GUI**

The createGUI method sets up the main frame and its components.

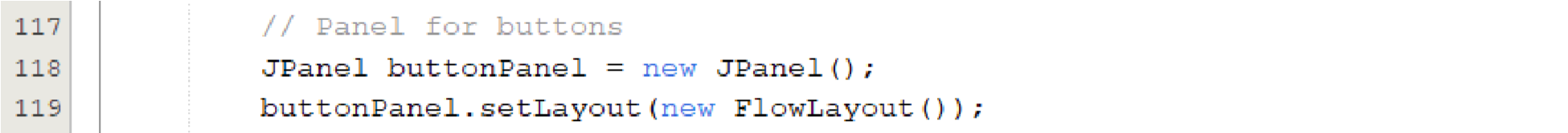




**Button Panel Creation**

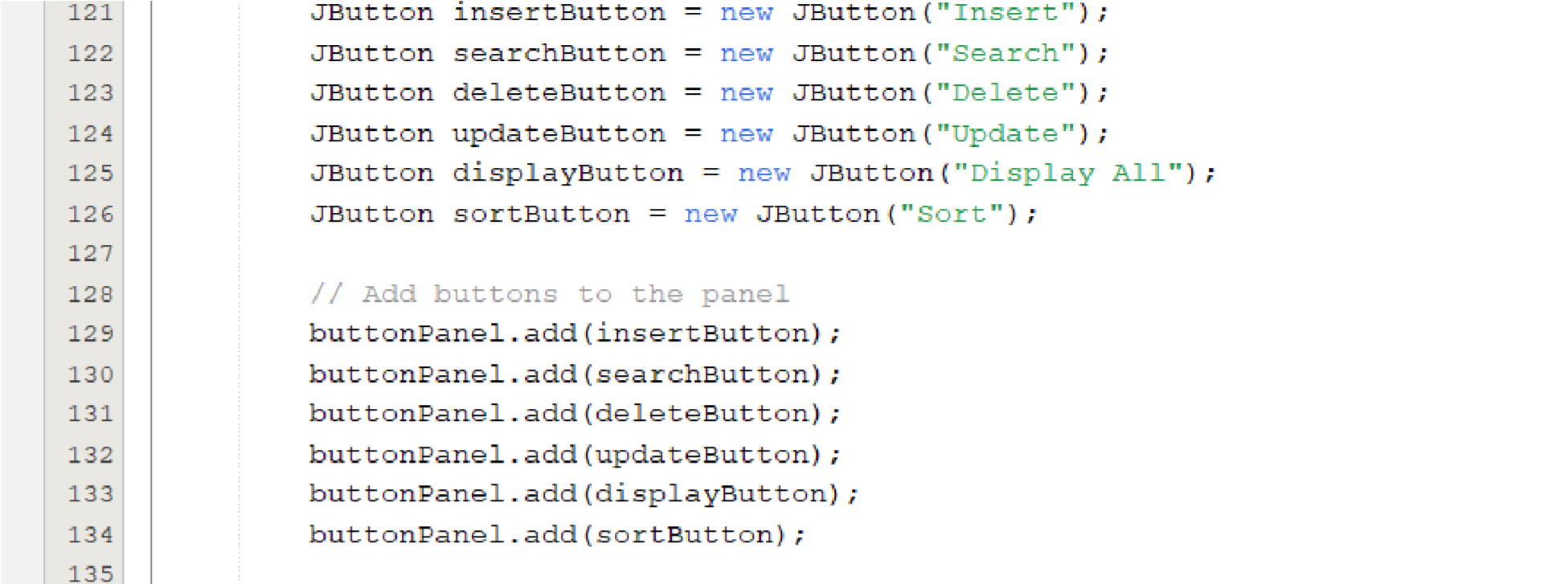
The button panel is created to hold various buttons for the phonebook application.

**Button Panel Initialization**

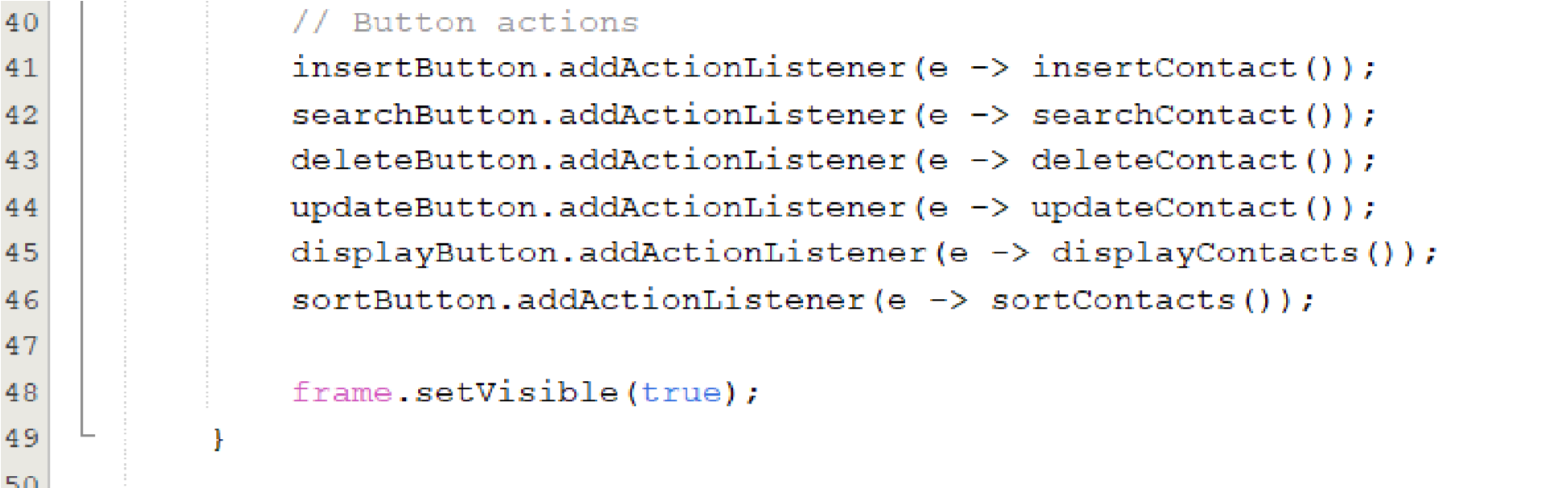


**Button Creation**

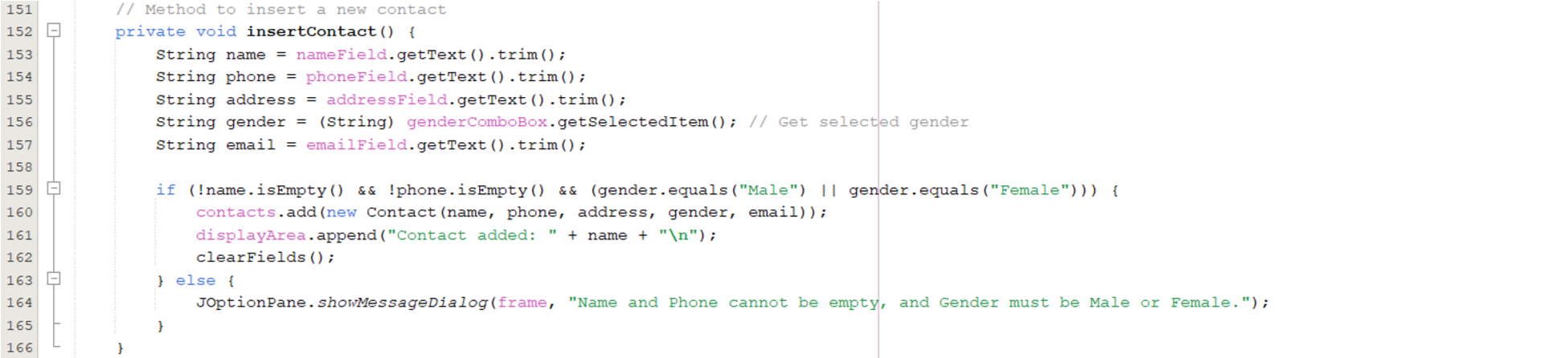
Several buttons are created for different actions:



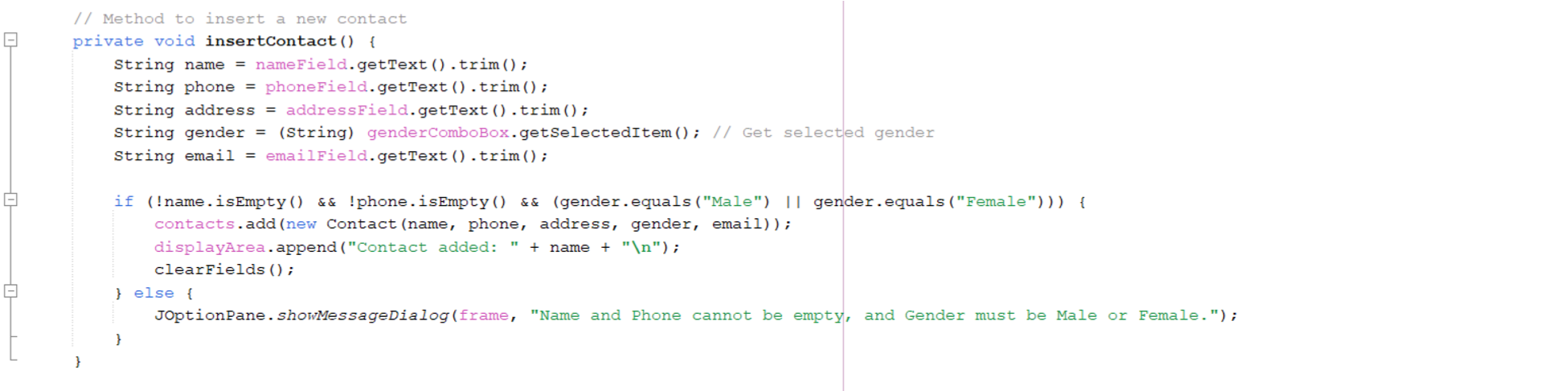
Action listeners are added to each button to define their behavior when clicked:



Method to Insert a New Contact

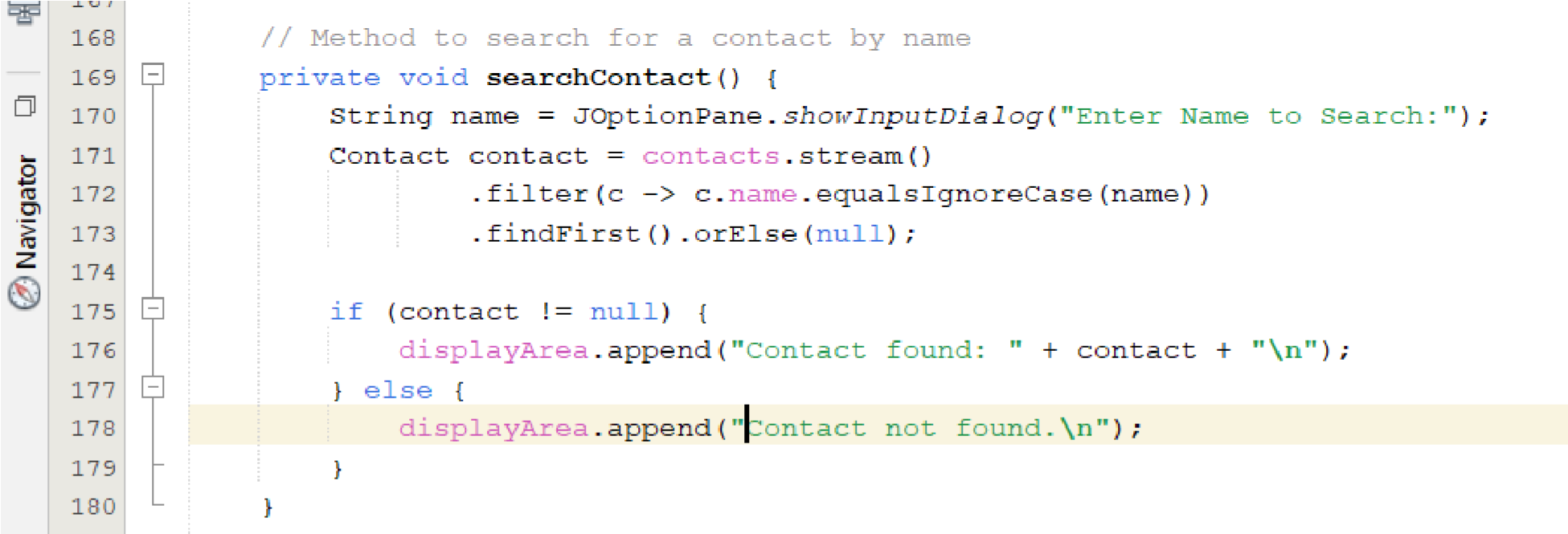


The insertContact method adds a new contact to the list if the required fields are not empty and the gender is valid.

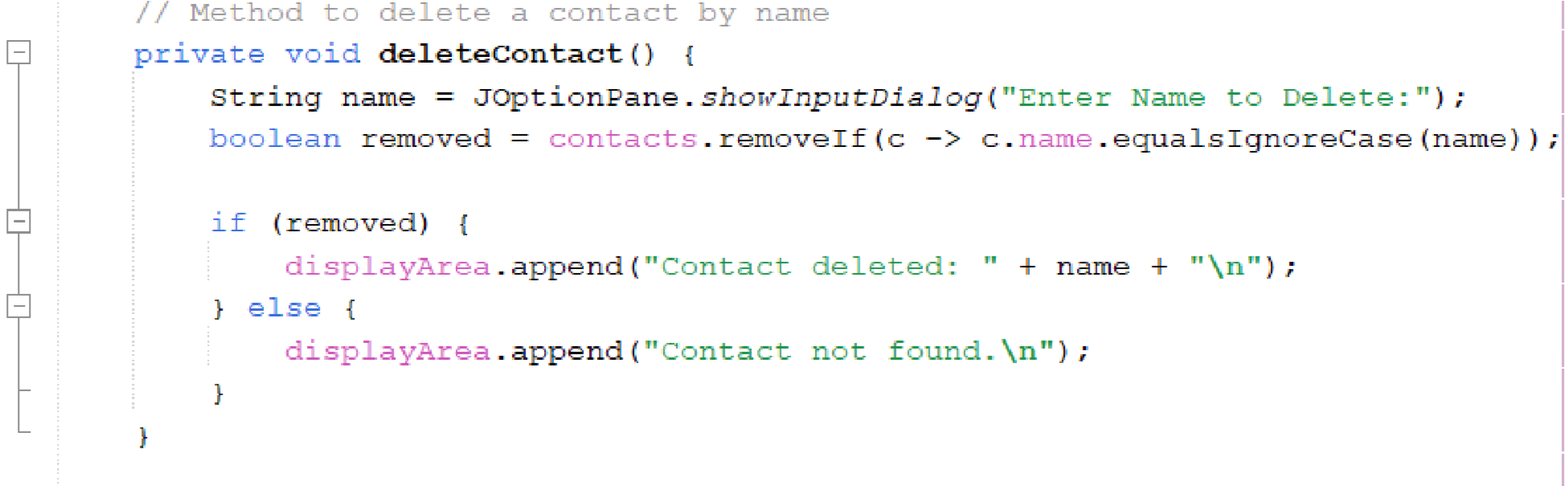


Method to Search for a Contact by Name

The searchContact method searches for a contact by name and displays the result.



Method to Delete a Contact by Name



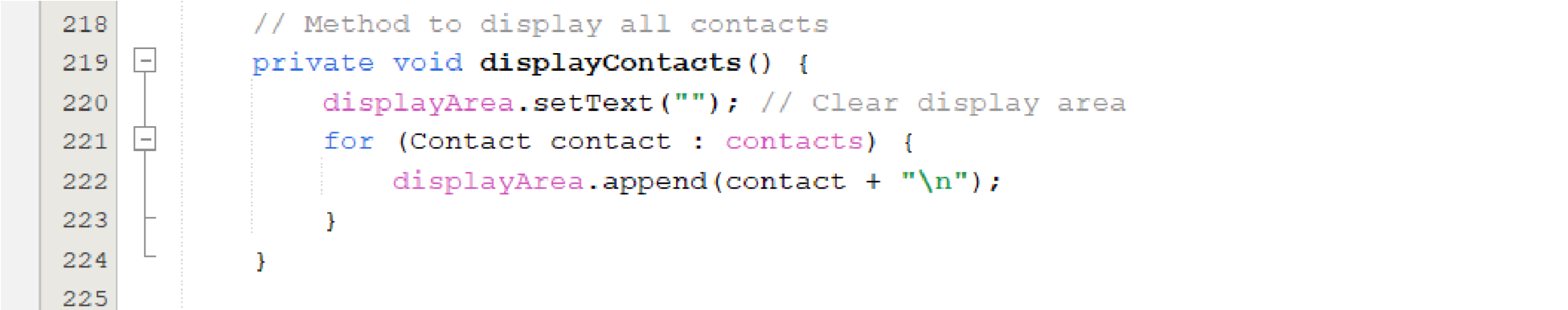
Method to Update an Existing Contact’s Information

The updateContact method updates the information of an existing contact.



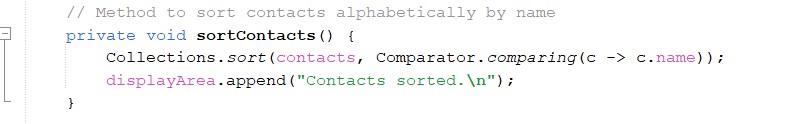
Method to Display All Contacts

The displayContacts method displays all contacts in the list.



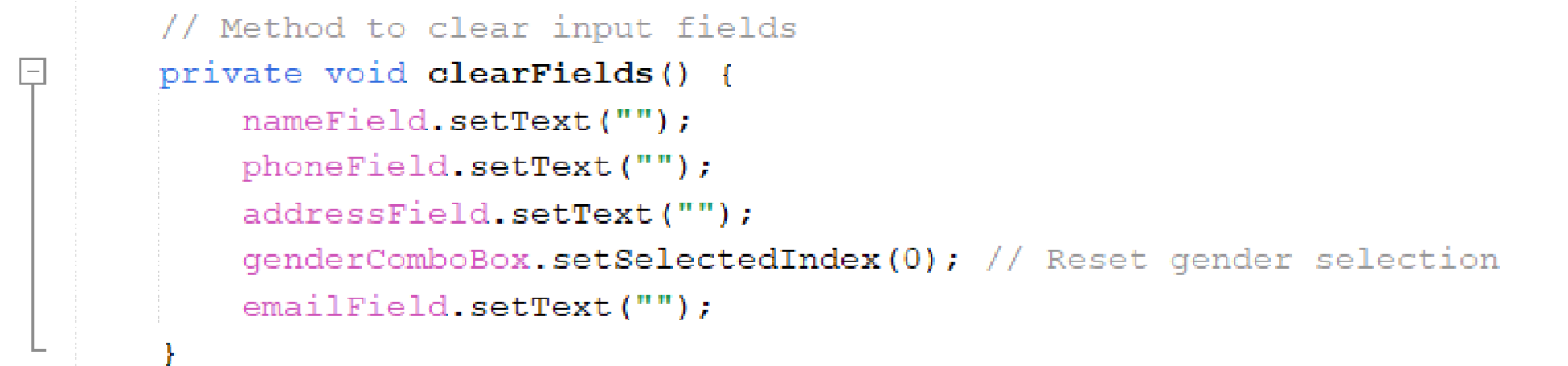
Method to Sort Contacts Alphabetically by Name

The sortContacts method sorts the contacts alphabetically by name.



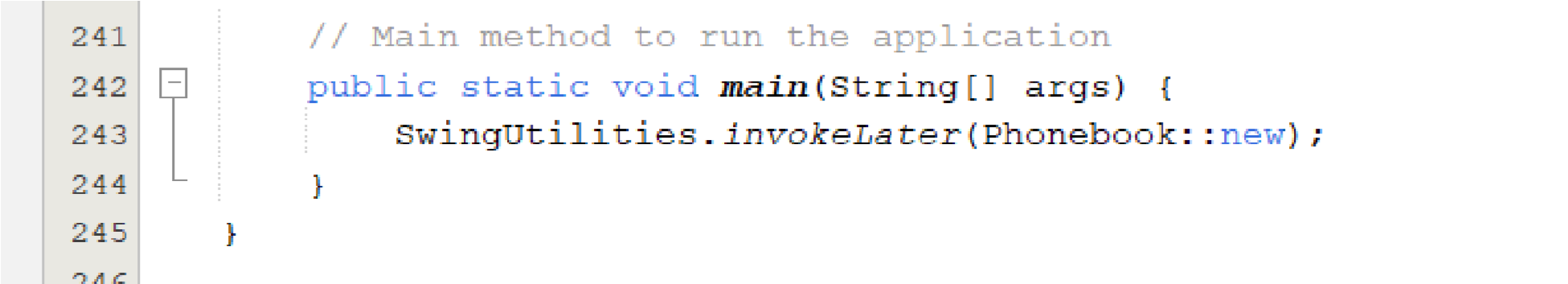
**Method to Clear Input Fields**

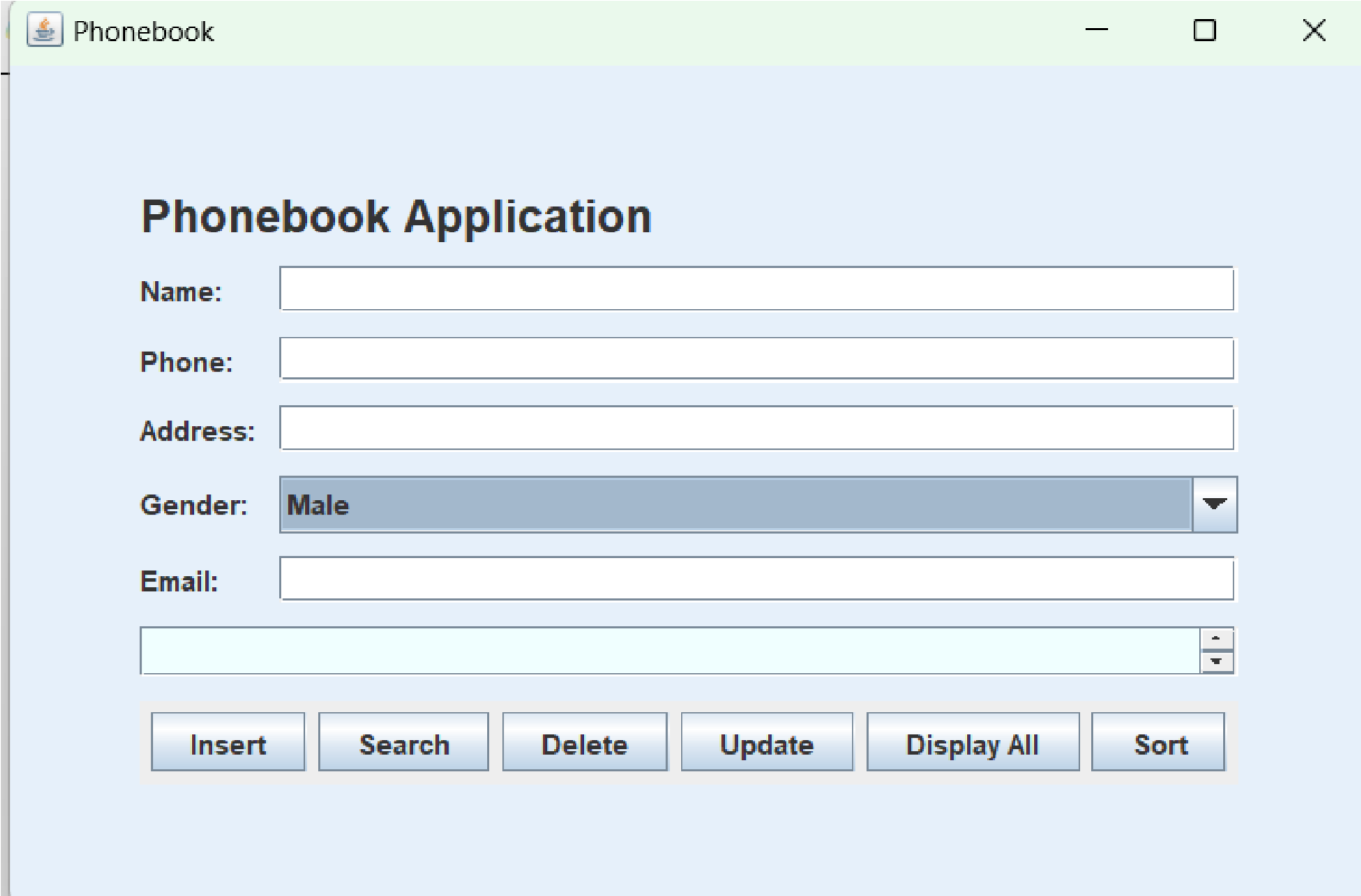
The clearFields method clears all the input fields in the form.



**Main Method to Run the Application**

The main method is the entry point of the application. It uses SwingUtilities.invokeLater to ensure that the GUI creation and updates are performed on the Event Dispatch Thread.





**Application Title**

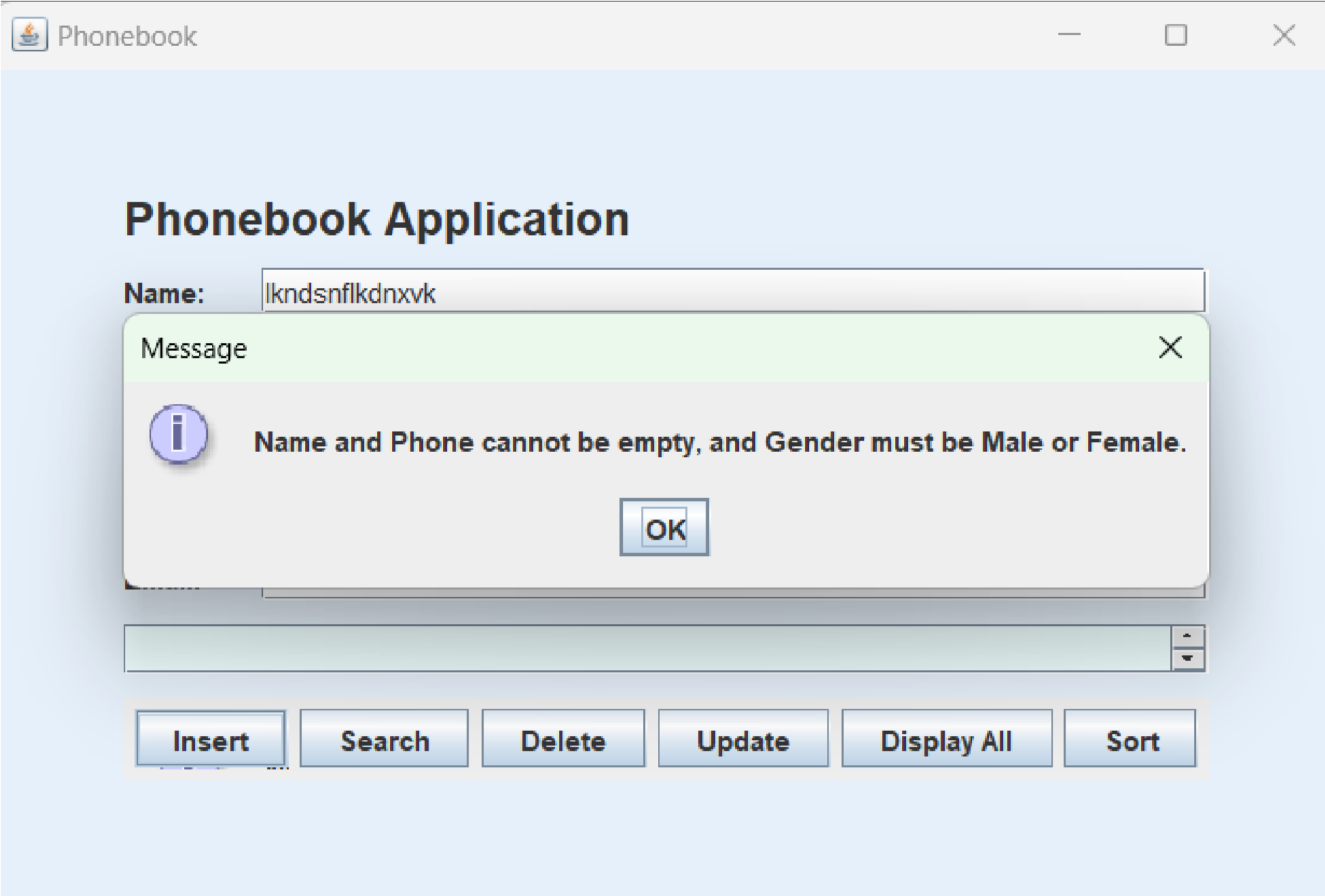
The title of the application window is **“Phonebook Application”**. **Input Fields**

The application contains several input fields for entering contact information:

* **Name**: A text field for entering the contact’s name.
* **Phone**: A text field for entering the contact’s phone number.
* **Address**: A text field for entering the contact’s address.
* **Gender**: A dropdown menu for selecting the contact’s gender (e.g., Male, Female).
* **Email**: A text field for entering the contact’s email address. Buttons

The application includes several buttons for performing different actions:

* **Insert**: Adds a new contact to the phonebook.
* **Search**: Searches for a contact by name.
* **Delete**: Deletes a contact by name.
* **Update**: Updates the information of an existing contact.
* **Display All**: Displays all contacts in the phonebook.
* **Sort**: Sorts the contacts alphabetically by name.



**Error Message**

**Application Title**

The title of the application window is **“Phonebook Application”**.

**Error Message Dialog**

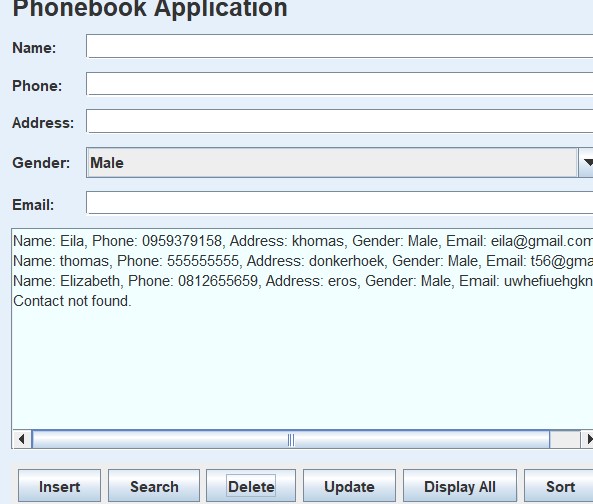
A dialog box titled **“Message”** appears within the application window. This dialog box contains an informational icon and the following text:

Name and Phone cannot be empty, and Gender must be Male or Female. **OK Button**

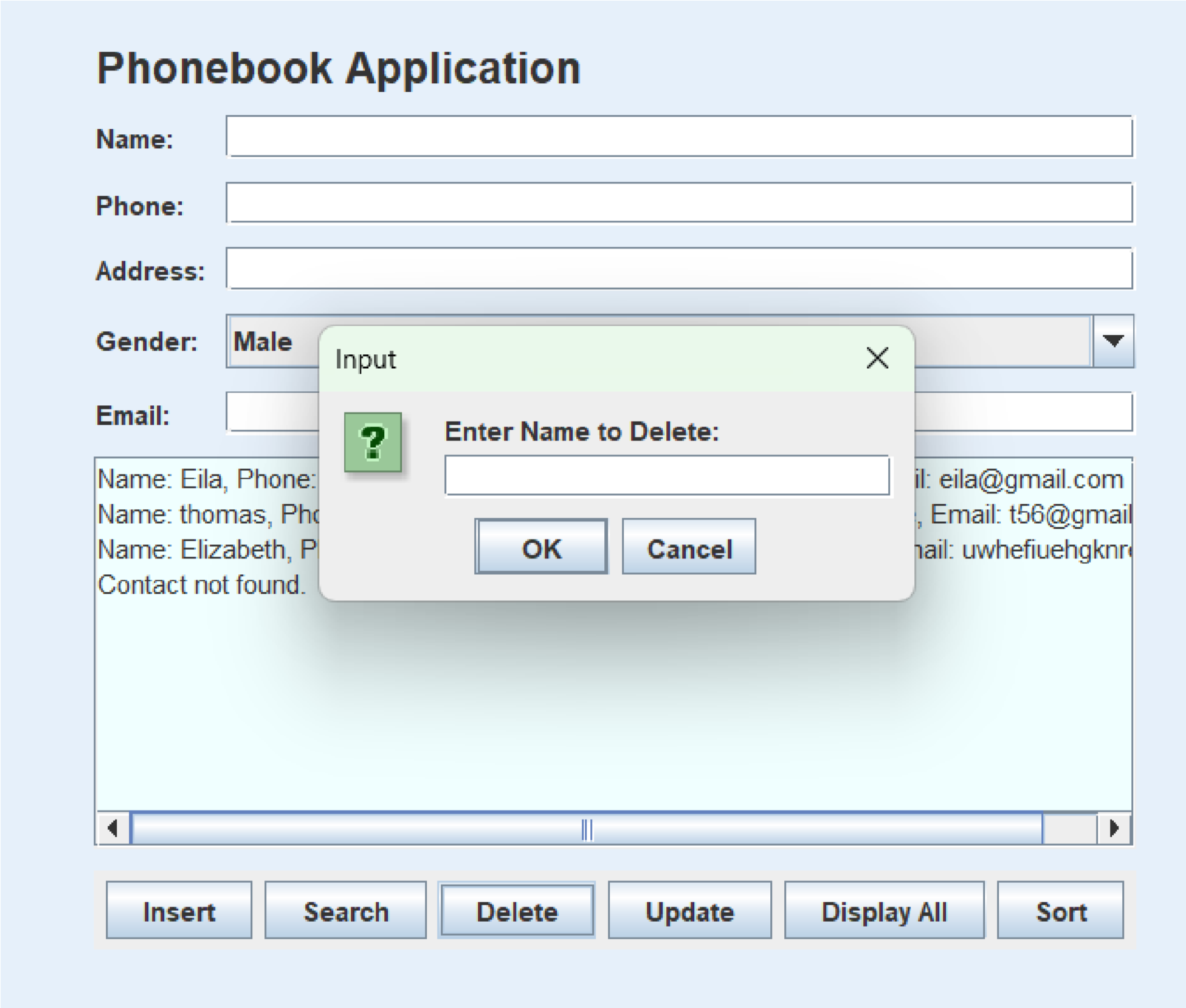
Below the error message, there is an **“OK”** button that the user can click to close the dialog box. **Context of the Error Message**

This error message is likely triggered when the user attempts to insert a new contact without providing a name or phone number, or if the gender field is not set to either “Male” or “Female”. This validation ensures that essential contact information is provided and that the gender field contains a valid value.

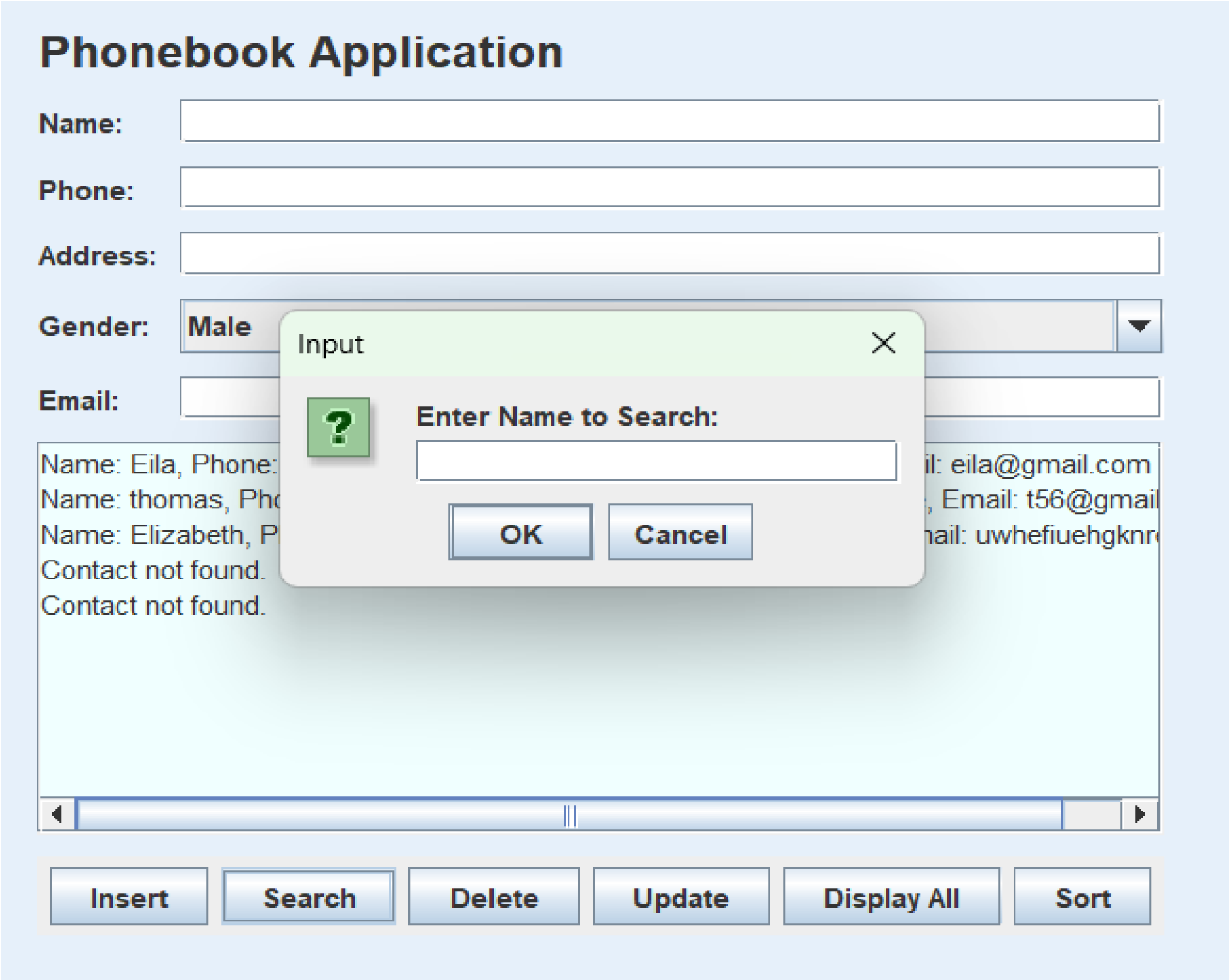
**The display all button function**



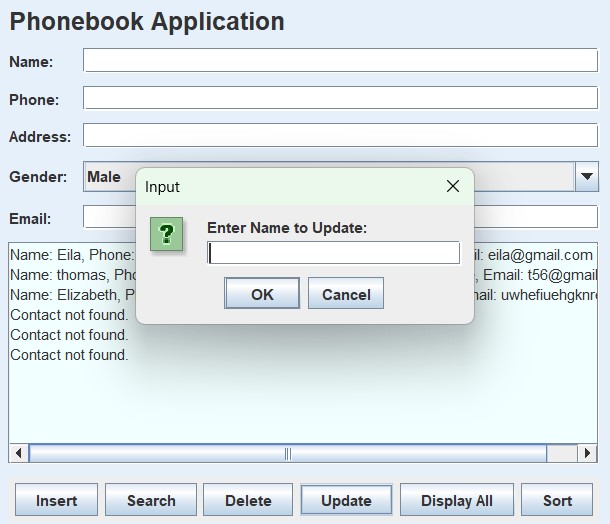
**The delete button**



**The search button**



**The update button**



**SOURCE CODE**

package phonebooksystem;  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.\*;  
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.Comparator;  
  
*// Class to represent a contact*class Contact {  
 String name;  
 String phoneNumber;  
 String address;  
 String gender;  
 String email;  
  
 *// Constructor to initialize contact details* Contact(String name, String phoneNumber, String address, String gender, String email) {  
 this.name = name;  
 this.phoneNumber = phoneNumber;  
 this.address = address;  
 this.gender = gender;  
 this.email = email;  
 }  
  
 *// Override toString method to display contact details* @Override  
 public String toString() {  
 return String.*format*("Name: %s, Phone: %s, Address: %s, Gender: %s, Email: %s",  
 name, phoneNumber, address, gender, email);  
 }  
}  
  
*// Main class for the phonebook application with GUI*public class Phonebook {  
 private ArrayList<Contact> contacts; *// List to store contacts* private JFrame frame;  
 private JTextArea displayArea;  
 private JTextField nameField, phoneField, addressField, emailField;  
 private JComboBox<String> genderComboBox; *// Combo box for gender selection  
  
 // Constructor to initialize the phonebook and GUI components* public Phonebook() {  
 contacts = new ArrayList<>();  
 createGUI();  
 }  
  
 *// Method to create the GUI* private void createGUI() {  
 frame = new JFrame("Phonebook");  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 frame.setSize(600, 400);  
 frame.setLayout(new GridBagLayout());  
 frame.getContentPane().setBackground(new Color(230, 240, 250));  
  
 GridBagConstraints gbc = new GridBagConstraints();  
 gbc.fill = GridBagConstraints.*HORIZONTAL*;  
 gbc.insets = new Insets(5, 5, 5, 5);  
  
 *// Title Label* JLabel titleLabel = new JLabel("Phonebook Application");  
 titleLabel.setFont(new Font("Arial", Font.*BOLD*, 20));  
 gbc.gridx = 0;  
 gbc.gridy = 0;  
 gbc.gridwidth = 2;  
 frame.add(titleLabel, gbc);  
  
 *// Input fields for contact details* gbc.gridwidth = 1;  
 gbc.gridy = 1;  
 frame.add(new JLabel("Name:"), gbc);  
 nameField = new JTextField(15);  
 gbc.gridx = 1;  
 frame.add(nameField, gbc);  
  
 gbc.gridx = 0;  
 gbc.gridy = 2;  
 frame.add(new JLabel("Phone:"), gbc);  
 phoneField = new JTextField(15);  
 gbc.gridx = 1;  
 frame.add(phoneField, gbc);  
  
 gbc.gridx = 0;  
 gbc.gridy = 3;  
 frame.add(new JLabel("Address:"), gbc);  
 addressField = new JTextField(15);  
 gbc.gridx = 1;  
 frame.add(addressField, gbc);  
  
 gbc.gridx = 0;  
 gbc.gridy = 4;  
 frame.add(new JLabel("Gender:"), gbc);  
 *// Combo box for gender selection* String[] genders = {"Male", "Female"};  
 genderComboBox = new JComboBox<>(genders);  
 gbc.gridx = 1;  
 frame.add(genderComboBox, gbc);  
  
 gbc.gridx = 0;  
 gbc.gridy = 5;  
 frame.add(new JLabel("Email:"), gbc);  
 emailField = new JTextField(15);  
 gbc.gridx = 1;  
 frame.add(emailField, gbc);  
  
 *// Text area to display contacts* displayArea = new JTextArea(10, 40);  
 displayArea.setEditable(false);  
 displayArea.setBackground(new Color(240, 255, 255));  
 JScrollPane scrollPane = new JScrollPane(displayArea);  
 gbc.gridx = 0;  
 gbc.gridy = 6;  
 gbc.gridwidth = 2;  
 frame.add(scrollPane, gbc);  
  
 *// Panel for buttons* JPanel buttonPanel = new JPanel();  
 buttonPanel.setLayout(new FlowLayout());  
  
 JButton insertButton = new JButton("Insert");  
 JButton searchButton = new JButton("Search");  
 JButton deleteButton = new JButton("Delete");  
 JButton updateButton = new JButton("Update");  
 JButton displayButton = new JButton("Display All");  
 JButton sortButton = new JButton("Sort");  
  
 *// Add buttons to the panel* buttonPanel.add(insertButton);  
 buttonPanel.add(searchButton);  
 buttonPanel.add(deleteButton);  
 buttonPanel.add(updateButton);  
 buttonPanel.add(displayButton);  
 buttonPanel.add(sortButton);  
  
 *// Add button panel to the frame* gbc.gridy = 7;  
 frame.add(buttonPanel, gbc);  
  
 *// Button actions* insertButton.addActionListener(e -> insertContact());  
 searchButton.addActionListener(e -> searchContact());  
 deleteButton.addActionListener(e -> deleteContact());  
 updateButton.addActionListener(e -> updateContact());  
 displayButton.addActionListener(e -> displayContacts());  
 sortButton.addActionListener(e -> sortContacts());  
  
 frame.setVisible(true);  
 }  
  
 *// Method to insert a new contact* private void insertContact() {  
 String name = nameField.getText().trim();  
 String phone = phoneField.getText().trim();  
 String address = addressField.getText().trim();  
 String gender = (String) genderComboBox.getSelectedItem(); *// Get selected gender* String email = emailField.getText().trim();  
  
 if (!name.isEmpty() && !phone.isEmpty() && (gender.equals("Male") || gender.equals("Female"))) {  
 contacts.add(new Contact(name, phone, address, gender, email));  
 displayArea.append("Contact added: " + name + "\n");  
 clearFields();  
 } else {  
 JOptionPane.*showMessageDialog*(frame, "Name and Phone cannot be empty, and Gender must be Male or Female.");  
 }  
 }  
  
 *// Method to search for a contact by name* private void searchContact() {  
 String name = JOptionPane.*showInputDialog*("Enter Name to Search:");  
 Contact = contacts.stream()  
 .filter(c -> c.name.equalsIgnoreCase(name))  
 .findFirst().orElse(null);  
  
 if (contact != null) {  
 displayArea.append("Contact found: " + contact + "\n");  
 } else {  
 displayArea.append("Contact not found.\n");  
 }  
 }  
  
 *// Method to delete a contact by name* private void deleteContact() {  
 String name = JOptionPane.*showInputDialog*("Enter Name to Delete:");  
 boolean removed = contacts.removeIf(c -> c.name.equalsIgnoreCase(name));  
  
 if (removed) {  
 displayArea.append("Contact deleted: " + name + "\n");  
 } else {  
 displayArea.append("Contact not found.\n");  
 }  
 }  
  
 *// Method to update an existing contact's information* private void updateContact() {  
 String name = JOptionPane.*showInputDialog*("Enter Name to Update:");  
 Contact = contacts.stream()  
 .filter(c -> c.name.equalsIgnoreCase(name))  
 .findFirst().orElse(null);  
  
 if (contact != null) {  
 String newPhone = JOptionPane.*showInputDialog*("Enter New Phone Number:", contact.phoneNumber);  
 String newAddress = JOptionPane.*showInputDialog*("Enter New Address:", contact.address);  
 String newGender = (String) JOptionPane.*showInputDialog*(frame, "Select New Gender:", "Update Gender",  
 JOptionPane.*QUESTION\_MESSAGE*, null, new String[]{"Male", "Female"}, contact.gender);  
 String newEmail = JOptionPane.*showInputDialog*("Enter New Email:", contact.email);  
  
 contact.phoneNumber = newPhone;  
 contact.address = newAddress;  
 contact.gender = newGender;  
 contact.email = newEmail;  
 displayArea.append("Contact updated: " + contact + "\n");  
 } else {  
 displayArea.append("Contact not found.\n");  
 }  
 }  
  
 *// Method to display all contacts* private void displayContacts() {  
 displayArea.setText(""); *// Clear display area* for (Contact : contacts) {  
 displayArea.append(contact + "\n");  
 }  
 }  
  
 *// Method to sort contacts alphabetically by name* private void sortContacts() {  
 Collections.*sort*(contacts, Comparator.*comparing*(c -> c.name));  
 displayArea.append("Contacts sorted.\n");  
 }  
  
 *// Method to clear input fields* private void clearFields() {  
 nameField.setText("");  
 phoneField.setText("");  
 addressField.setText("");  
 genderComboBox.setSelectedIndex(0); *// Reset gender selection* emailField.setText("");  
 }  
  
 *// Main method to run the application* public static void main(String[] args) {  
 SwingUtilities.*invokeLater*(Phonebook::new);  
 }  
}