# Section B of the Phone Book application

## Source codes

## Source codes

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.EmptyBorder;
import java.util.ArrayList;
import java.util.Collections;

public class Phonebookapp implements ActionListener {

   JFrame frame;
   JTextArea nameField, phoneField, searchField;
   JList<String> contactList;
   DefaultListModel<String> contactListModel;
   JButton addButton, viewButton, searchButton, deleteButton, updateButton;

   public Phonebookapp() {
      // Frame setup
      frame = new JFrame("Phonebook - Revamped");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setSize(600, 400);
      frame.setLayout(new FlowLayout());

      // Fields and Buttons
      nameField = new JTextArea("Enter Name", 1, 15);
      phoneField = new JTextArea("Enter Phone", 1, 15);
      searchField = new JTextArea("Search Contacts", 1, 15);

      addButton = new JButton("Add");
      viewButton = new JButton("View");
      searchButton = new JButton("Search");
      deleteButton = new JButton("Delete");
      updateButton = new JButton("Update");

      // Adding action listeners
```

```
        addButton.addActionListener(this);
        viewButton.addActionListener(this);
        searchButton.addActionListener(this);
        deleteButton.addActionListener(this);
        updateButton.addActionListener(this);

        // Input validation
        nameField.addKeyListener(new KeyAdapter() {
            @Override
            public void keyTyped(KeyEvent e) {
                if (!Character.isLetter(e.getKeyChar()) && !Character.isWhitespace(e.getKeyChar()))
{
                    e.consume(); // Ignore non-letter input
                }
            }
        });

        phoneField.addKeyListener(new KeyAdapter() {
            @Override
            public void keyTyped(KeyEvent e) {
                if (!Character.isDigit(e.getKeyChar())) {
                    e.consume(); // Ignore non-digit input
                }
            }
        });

        // Contact List setup
        contactListModel = new DefaultListModel<>();
        contactList = new JList<>(contactListModel);
        contactList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        JScrollPane scrollPane = new JScrollPane(contactList);

        // Layout additions
        frame.add(nameField);
        frame.add(phoneField);
        frame.add(searchField);
        frame.add(addButton);
        frame.add(viewButton);
        frame.add(searchButton);
        frame.add(deleteButton);
        frame.add(updateButton);
```

```
        frame.add(scrollPane);

        frame.setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == addButton) {
            String name = nameField.getText().trim();
            String phone = phoneField.getText().trim();
            if (!name.isEmpty() && !phone.isEmpty()) {
                if (!isDuplicateContact(name, phone, "", "")) {
                    String contact = "Name: " + name + ", Phone: " + phone;
                    contactListModel.addElement(contact);
                    sortContacts();
                    nameField.setText("");
                    phoneField.setText("");
                    JOptionPane.showMessageDialog(frame, "Contact successfully added.");
                } else {
                    JOptionPane.showMessageDialog(frame, "This contact already exists.");
                }
            } else {
                JOptionPane.showMessageDialog(frame, "Please enter both name and phone
number.");
            }
        } else if (e.getSource() == viewButton) {
            if (contactListModel.isEmpty()) {
                JOptionPane.showMessageDialog(frame, "No contacts to display.");
            }
        } else if (e.getSource() == searchButton) {
            String search = searchField.getText().trim();
            if (!search.isEmpty()) {
                searchContact(search);
            } else {
                JOptionPane.showMessageDialog(frame, "Please enter a name or phone number
to search.");
            }
        } else if (e.getSource() == deleteButton) {
            String selectedContact = contactList.getSelectedValue();
            if (selectedContact != null) {
                int response = JOptionPane.showConfirmDialog(frame,
```

```
                "Are you sure you want to delete this contact?", "Confirm Delete",
                JOptionPane.YES_NO_OPTION);
            if (response == JOptionPane.YES_OPTION) {
              contactListModel.removeElement(selectedContact);
              JOptionPane.showMessageDialog(frame, "Contact deleted successfully.");
            }
          } else {
            JOptionPane.showMessageDialog(frame, "Please select a contact to delete.");
          }
        } else if (e.getSource() == updateButton) {
          String selectedContact = contactList.getSelectedValue();
          if (selectedContact != null) {
            String[] parts = selectedContact.split(", ");
            String currentName = parts[0].substring(6);
            String currentPhone = parts[1].substring(7);

            String newName = JOptionPane.showInputDialog(frame, "Edit Name:",
currentName);
            String newPhone = JOptionPane.showInputDialog(frame, "Edit Phone:",
currentPhone);

            if (newName != null && newPhone != null && !newName.isEmpty() &&
!newPhone.isEmpty()) {
              // Pass current contact details to avoid checking them in the duplicate check
              if (!isDuplicateContact(newName, newPhone, currentName, currentPhone)) {
                String updatedContact = "Name: " + newName + ", Phone: " + newPhone;
                contactListModel.setElementAt(updatedContact,
contactList.getSelectedIndex());
                JOptionPane.showMessageDialog(frame, "Contact updated successfully.");
              } else {
                JOptionPane.showMessageDialog(frame, "This contact already exists.");
              }
            }
          } else {
            JOptionPane.showMessageDialog(frame, "Please select a contact to update.");
          }
        }
      }

      private boolean isDuplicateContact(String name, String phone, String currentName, String
currentPhone) {
```

```
        for (int i = 0; i < contactListModel.size(); i++) {
            String contact = contactListModel.get(i);
            String[] parts = contact.split(", ");
            String existingName = parts[0].substring(6);
            String existingPhone = parts[1].substring(7);

            // Skip checking if this is the current contact being updated
            if (existingName.equalsIgnoreCase(currentName) &&
existingPhone.equals(currentPhone)) {
                continue;
            }

            // Check for duplicates
            if (existingName.equalsIgnoreCase(name) || existingPhone.equals(phone)) {
                return true;
            }
        }
        return false;
    }

    private void sortContacts() {
        ArrayList<String> contactListArray = new ArrayList<>();
        for (int i = 0; i < contactListModel.size(); i++) {
            contactListArray.add(contactListModel.get(i));
        }
        Collections.sort(contactListArray);
        contactListModel.clear();
        contactListArray.forEach(contactListModel::addElement);
    }

    private void searchContact(String search) {
        DefaultListModel<String> searchResults = new DefaultListModel<>();
        boolean found = false;
        for (int i = 0; i < contactListModel.size(); i++) {
            String contact = contactListModel.get(i);
            if (contact.toLowerCase().contains(search.toLowerCase())) {
                searchResults.addElement(contact);
                found = true;
            }
        }
        if (found) {
```

```
            contactList.setModel(searchResults);
        } else {
            JOptionPane.showMessageDialog(frame, "No matching contacts found.");
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(Phonebookapp::new);
    }
}
```