DSA

PHONE BOOK

GROUP MEMBERS

Name	Student number
Nabot Albert	223105856
Moses filippus	224088629
Wilhelm Kafidi	224021109
Ligoleni Young	222119950
Aina Kakonda	223098663
Chriselia M H //Naobes	224062662

1. Description of the project

In this project we created a phone book app much like the that we get on our mobile devises. The phone book will do processes such as 1. Insert Contact 2. Search Contact 3. Display all contacts 4. Delete Contact 5. Update Contact 6. Sort Contacts.

2. Description of our code

Modules and functions

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
```

Imports

1. Import java.swing.*; -is a development tool used for GUI creation.
2. Import java.awt.*; - is used for creating user interfaces and painting graphics and images.
3. Import java.util.*; - provide functionality for commonly used cases which are usually encountered like scanners.

Class

```
public class PhoneBookApp extends JFrame implements ActionListener {
```

Phone book class - extends JFRAMES meaning its a type of window.

-Implements ActionListener, which allows the app to respond to button clicks.

Attribute of *PhoneBookApp*

```
private LinkedList<Contact> phoneBook;  13 usages

private JTextField nameField, phoneField, searchField;  4 usages
private JTextArea displayArea;  14 usages
private JButton addButton, searchButton, displayButton, deleteButton, updateButton, sortButton;  4 usages
```

- phoneBook: A LinkedList to store Contact objects (name and phone number).
- nameField, phoneField, searchField: Input fields for the user to enter names, phone numbers, or search queries.

- **displayArea**: A text area to display information about contacts.
- **Buttons**: These trigger actions like adding, searching, displaying, deleting, updating, and sorting contacts.

Constructor *PhoneBookApp*

```
public PhoneBookApp() {  1 usage
    phoneBook = new LinkedList<>();  // Initializes the LinkedList

    setTitle("Phone Book Application");
    setSize( width: 800,  height: 800);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLayout(new BorderLayout());
    setLocationRelativeTo(null);
    setResizable(false);

    JPanel inputPanel = new JPanel(new GridLayout( rows: 5,  cols: 2));
```

At this code, going downwards it;

- Sets up the GUI layout with buttons, input fields, and panels.

- it also helps set but action, when the button is clicked.

Method *actionPerformed(ActionEvent e)*

```
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == addButton) {
        String name = nameField.getText();
        String phone = phoneField.getText();
        if (!name.isEmpty() && !phone.isEmpty()) {
            phoneBook.add(new Contact(name, phone));
            displayArea.setText("Contact Added: " + name + " -> " + phone);
            nameField.setText("");
            phoneField.setText("");
        } else {
            displayArea.setText("Please enter both name and phone number.");
        }
    } else if (e.getSource() == searchButton) {
        String name = searchField.getText();
        Contact found = searchContactByName(name);
        if (found != null) {
            displayArea.setText("Found: " + found.getName() + " -> " + found.getPhone());
        } else {
            displayArea.setText("Contact not found.");
        }
    } else if (e.getSource() == displayButton) {
        displayAllContacts();
    } else if (e.getSource() == deleteButton) {
        String name = searchField.getText();
        Contact found = searchContactByName(name);
        if (found != null) {
            phoneBook.remove(found);
            displayArea.setText("Deleted Contact: " + name);
```

- This part of the code handles the logic of the buttons (till where it ends)

- Add Contact button: Takes input , creates a new Contact, and adds it to the phoneBook.
- Search Contact button: Searches for a contact by name.
- Display All Contacts button: Show all the contacts in the list.
- Delete Contact button: Removes a contact by name.
- Update Contact button: Updates a contact's phone number.
- Sort Contacts button: Sorts the phonebook alphabetically.

Helper methods

- Display

```
private void displayAllContacts() {  2 usages
    if (phoneBook.isEmpty()) {
        displayArea.setText("No contacts available.");
    } else {
        StringBuilder sb = new StringBuilder();
        for (Contact contact : phoneBook) {
            sb.append(contact.getName()).append(" -> ").append(contact.getPhone()).append("\n");
        }
        displayArea.setText(sb.toString());
    }
}
```

-Displays all contacts in the phonebook in the displayArea. If the list is empty, it informs the user.

- Sort

```
private void sortContacts() {  1 usage
    if (phoneBook.isEmpty()) {
        displayArea.setText("No contacts to sort.");
    } else {
        for (int i = 1; i < phoneBook.size(); i++) {
            Contact key = phoneBook.get(i);
            int j = i - 1;
            while (j >= 0 && phoneBook.get(j).getName().compareTo(key.getName()) > 0) {
                phoneBook.set(j + 1, phoneBook.get(j));
                j--;
            }
            phoneBook.set(j + 1, key);
        }
        displayAllContacts();
    }
}
```

- Sorts contacts alphabetically by name using insertion sort (we used insertion sort because it'a simple sorting algorithm that works well for small lists).

- Search

```
private Contact searchContactByName(String name) {  3 usages
    for (Contact contact : phoneBook) {
        if (contact.getName().equalsIgnoreCase(name)) {
            return contact;
        }
    }
    return null;
}
```

- Iterates through the phoneBook and compares contact names.
- Returns the Contact if found, or null if not.

- Main Method

```
    public static void main(String[] args) {
        PhoneBookApp app = new PhoneBookApp();
        app.setVisible(true);
    }
}
```

Creates an instance of PhoneBookApp and makes the application visible.

3. Code

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.util.HashMap;

public class PhoneBookApp extends JFrame implements ActionListener {
```

```
    private LinkedList<Contact> phoneBook;

    private JTextField nameField, phoneField, searchField;
    private JTextArea displayArea;
    private JButton addButton, searchButton, displayButton, deleteButton, updateButton,
sortButton;

    public PhoneBookApp() {
        phoneBook = new LinkedList<>();

        setTitle("Phone Book Application");
        setSize(800, 800);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());
        setLocationRelativeTo(null);
        setResizable(false);

        JPanel inputPanel = new JPanel(new GridLayout(5, 2));

        inputPanel.add(new JLabel("Name:"));
        nameField = new JTextField();
        inputPanel.add(nameField);

        // Add phone number input
        inputPanel.add(new JLabel("Phone Number:"));
        phoneField = new JTextField();
        inputPanel.add(phoneField);

        // Add buttons for contact actions
        addButton = new JButton("Add Contact");
        addButton.addActionListener(this);
        inputPanel.add(addButton);

        // Search section
        inputPanel.add(new JLabel("Search by Name:"));
        searchField = new JTextField();
        inputPanel.add(searchField);

        searchButton = new JButton("Search");
        searchButton.addActionListener(this);
        inputPanel.add(searchButton);
```

```
    // Add display area
    displayArea = new JTextArea();
    displayArea.setEditable(false);
    add(new JScrollPane(displayArea), BorderLayout.CENTER);

    // Add buttons for other features
    displayButton = new JButton("Display All Contacts");
    displayButton.addActionListener(this);
    deleteButton = new JButton("Delete Contact");
    deleteButton.addActionListener(this);
    updateButton = new JButton("Update Contact");
    updateButton.addActionListener(this);
    sortButton = new JButton("Sort Contacts");
    sortButton.addActionListener(this);

    // Add panels to the frame
    JPanel actionPanel = new JPanel(new GridLayout(1, 4));
    actionPanel.add(displayButton);
    actionPanel.add(deleteButton);
    actionPanel.add(updateButton);
    actionPanel.add(sortButton);
    add(inputPanel, BorderLayout.NORTH);
    add(actionPanel, BorderLayout.SOUTH);
}

// Method to handle actions when buttons are clicked
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == addButton) {
        // Add contact logic
        String name = nameField.getText();
        String phone = phoneField.getText();
        if (!name.isEmpty() && !phone.isEmpty()) {
            phoneBook.add(new Contact(name, phone));
            displayArea.setText("Contact Added: " + name + " -> " + phone);
            nameField.setText("");
            phoneField.setText("");
        } else {
            displayArea.setText("Please enter both name and phone number.");
        }
    } else if (e.getSource() == searchButton) {
```

```
        // Search contact logic
        String name = searchField.getText();
        Contact found = searchContactByName(name);
        if (found != null) {
            displayArea.setText("Found: " + found.getName() + " -> " + found.getPhone());
        } else {
            displayArea.setText("Contact not found.");
        }
    } else if (e.getSource() == displayButton) {
        // Display all contacts
        displayAllContacts();
    } else if (e.getSource() == deleteButton) {
        // Delete contact logic
        String name = searchField.getText();
        Contact found = searchContactByName(name);
        if (found != null) {
            phoneBook.remove(found);
            displayArea.setText("Deleted Contact: " + name);
        } else {
            displayArea.setText("Contact not found.");
        }
    } else if (e.getSource() == updateButton) {
        // Update contact logic
        String name = searchField.getText();
        String newPhone = phoneField.getText();
        Contact found = searchContactByName(name);
        if (found != null && !newPhone.isEmpty()) {
            found.setPhone(newPhone);
            displayArea.setText("Updated Contact: " + name + " -> " + newPhone);
        } else {
            displayArea.setText("Contact not found or phone number is empty.");
        }
    } else if (e.getSource() == sortButton) {
        // Sort contacts by name
        sortContacts();
    }
}

// Method to display all contacts
private void displayAllContacts() {
    if (phoneBook.isEmpty()) {
```

```
            displayArea.setText("No contacts available.");
        } else {
            StringBuilder sb = new StringBuilder();
            for (Contact contact : phoneBook) {
                sb.append(contact.getName()).append(" ->
").append(contact.getPhone()).append("\n");
            }
            displayArea.setText(sb.toString());
        }
    }

    // Method to sort contacts by name using insertion sort (suitable for linked lists)
    private void sortContacts() {
        if (phoneBook.isEmpty()) {
            displayArea.setText("No contacts to sort.");
        } else {
            // Perform insertion sort on the linked list
            for (int i = 1; i < phoneBook.size(); i++) {
                Contact key = phoneBook.get(i);
                int j = i - 1;
                while (j >= 0 && phoneBook.get(j).getName().compareTo(key.getName()) > 0) {
                    phoneBook.set(j + 1, phoneBook.get(j));
                    j--;
                }
                phoneBook.set(j + 1, key);
            }

            // Display sorted contacts
            displayAllContacts();
        }
    }

    // Helper method to search contact by name
    private Contact searchContactByName(String name) {
        for (Contact contact : phoneBook) {
            if (contact.getName().equalsIgnoreCase(name)) {
                return contact;
            }
        }
        return null;
    }
```

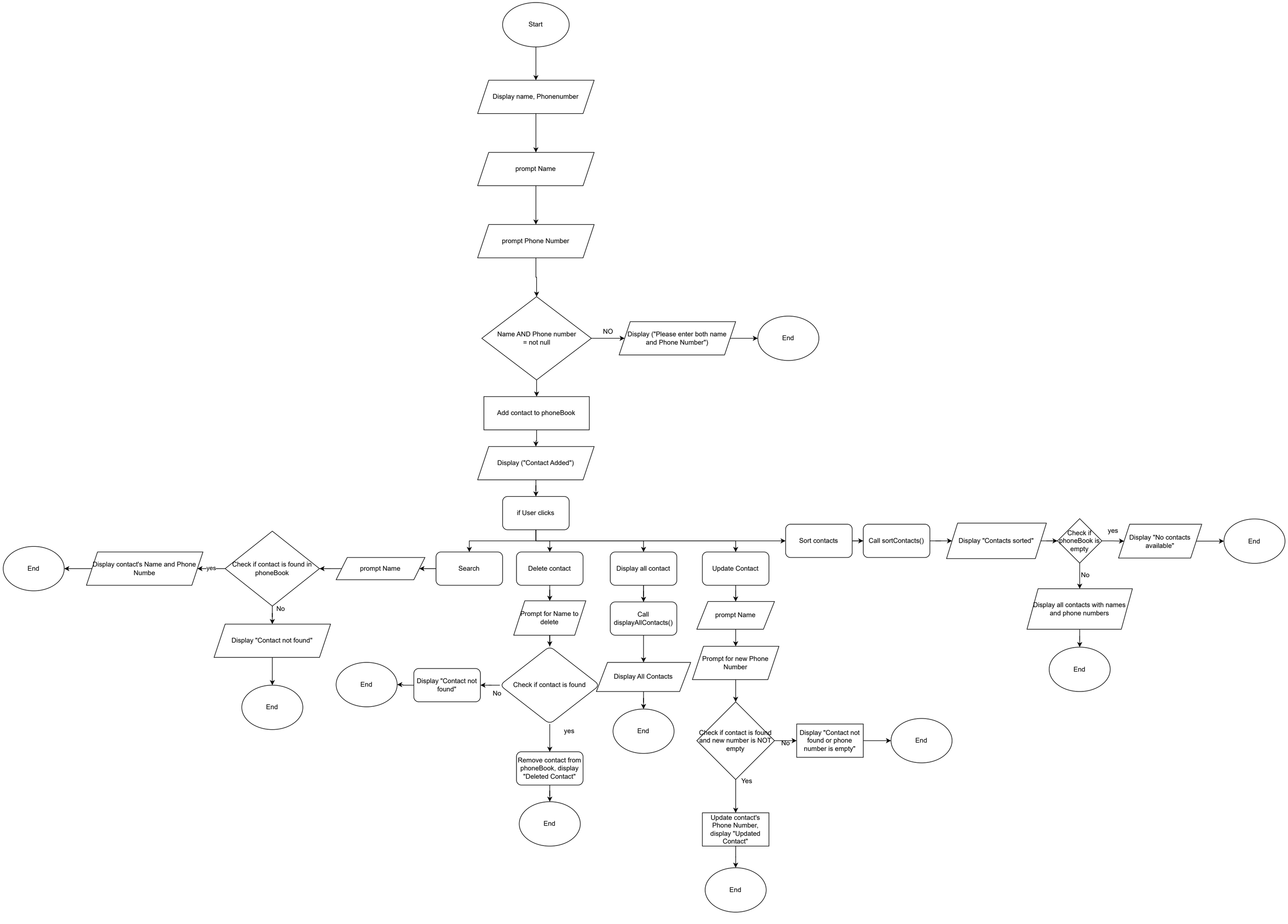
```
    // Main method to run the application
    public static void main(String[] args) {
        PhoneBookApp app = new PhoneBookApp();
        app.setVisible(true);
    }
}

// Contact class to store name and phone number
class Contact {
    private String name;
    private String phone;

    public Contact(String name, String phone) {
        this.name = name;
        this.phone = phone;
    }

    public String getName() {
        return name;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public String getPhone() {
        return phone;
    }
}
```

4. Pseudocode

```
  START
DEF onSearchButtonClick():
Prompt user for name
Get name
Prompt user for number
Get number
   IF name and phone fields are NOT empty:
      ADD contact to phoneBook
      DISPLAY "Contact Added"
   ELSE:
      DISPLAY "Please enter both name and phone number"

Prompt user for search item
   GET name from search field
   IF contact is found:
      DISPLAY contact's name and phone
   ELSE:
      DISPLAY "Contact not found"

DEF onDisplayButtonClick():
   CALL displayAllContacts()

DEF onDeleteButtonClick():
   Prompt user for name they want to delete
```

```
        get name from search
        IF contact found:
            REMOVE contact from phoneBook
            DISPLAY "Deleted Contact"
        ELSE:
            DISPLAY "Contact not found"

    DEF onUpdateButtonClick():
        Prompt user for name
        GET name from search field
        Prompt user for number
        GET number from phone field
        CALL searchContactByName(name)
        IF (name found AND number is NOT empty)
            UPDATE contact's phone
            DISPLAY "Updated Contact"
        ELSE:
            DISPLAY "Contact not found or phone number is empty"

    DEF onSortButtonClick():
        CALL sortContacts()
        DISPLAY "Contacts sorted"

DEF searchContactByName(name)
    FOR each contact in phoneBook:
        IF contact name matches name (case-insensitive):
            RETURN contact
    RETURN null
```

```
DEF displayAllContacts():
    IF phoneBook is empty:
        DISPLAY "No contacts available"
    ELSE:
        FOR each contact in phoneBook:
            DISPLAY contact's name and phone


DEF sortContacts():
    IF (phoneBook is NOT empty)
        FOR (i from 1 to size of phoneBook – 1)
            SET key to phoneBook[i]
            SET j to i - 1
            WHILE j >= 0 AND phoneBook[j].name > key.name:
                SHIFT phoneBook[j] one position to the right
                DECREMENT j
            INSERT key at position j + 1

END
```


Start
Display name, Phonenumber
prompt Name
prompt Phone Number
Name AND Phone number = not null
NO
Display ("Please enter both name and Phone Number")
End
Add contact to phoneBook
Display ("Contact Added")
if User clicks
Search
Delete contact
Display all contact
Update Contact
Sort contacts
Call sortContacts()
Display "Contacts sorted"
Check if phoneBook is empty
yes
Display "No contacts available"
End
No
Display all contacts with names and phone numbers
End
prompt Name
Check if contact is found in phoneBook
yes
Display contact's Name and Phone Numbe
End
No
Display "Contact not found"
End
Prompt for Name to delete
Check if contact is found
No
Display "Contact not found"
End
yes
Remove contact from phoneBook, display "Deleted Contact"
End
Call displayAllContacts()
Display All Contacts
End
prompt Name
Prompt for new Phone Number
Check if contact is found and new number is NOT empty
No
Display "Contact not found or phone number is empty"
End
Yes
Update contact's Phone Number, display "Updated Contact"
End