Phonebook Project/Assignment

Student Name	Student Number	Mode of Study	Contribution
Ndahafa Nghishoongele	223032344	FullTime	Flowchart
Tapiwa Machekera	224059483	FullTime	Code
Grace Urikos (Group Leader)	223051764	FullTime	Flowchart
Estevao Gurirab	224088149	Fulltime	GUI
Uetupa Rijarua	223066346	FullTime	Documentation
Gerson Elikana	222036489	Fulltime	Documentation

This code implements a simple phonebook application in Java. The phonebook allows the user to perform several actions, such as adding, searching, displaying, updating, and deleting contacts. The program uses arrays to store contact information, including names, addresses, and phone numbers, with a limit of 100 contacts.

Rundown of the pseudocode

The phonebook program lets users manage contacts. They can add, search for, view, delete, or update contacts, and exit the program.

Breakdown

1. **Start with an Empty Phonebook**
   - PHONEBOOK = []: This creates an empty list to store contacts.
2. **Main Loop**
   - WHILE TRUE: Keeps the program running until the user decides to exit.
3. **Show Menu**
   - Displays options like:
     - Insert Contact
     - Search Contact
     - Display All Contacts
     - Delete Contact
     - Update Contact
     - Exit
4. **User Input**
   - GET USER_CHOICE: The user selects an option.

Phonebook Project/Assignment

Options Explained

1. **Insert Contact**
    - User enters name, phone number, and email.
    - The contact is added to the phonebook.
2. **Search Contact**
    - User types a name or phone number.
    - The program looks for a match and shows the contact if found.
3. **Display All Contacts**
    - If the phonebook is empty, it shows a message.
    - Otherwise, it lists all contacts.
4. **Delete Contact**
    - User provides a name or phone number.
    - The contact is removed if found; otherwise, it shows a not found message.
5. **Update Contact**
    - User inputs a name or phone number to find the contact.
    - They enter new details to update it if the contact exists.
6. **Exit**
    - The program stops running when the user selects this option.

Invalid Input Handling

- If the user selects an option that isn't listed, it prompts them to try again.

Phonebook Project/Assignment

Breakdown of the code

Contact.java

```java
public class Contact {
    String name;
    String phoneNumber;

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

Phonebook Project/Assignment

Phonebook.java

```java
import javax.swing.*;
import java.util.Arrays;

public class Phonebook {
    private Contact[] contacts;
    private int size;

    public Phonebook(int capacity) {
        contacts = new Contact[capacity];
        size = 0;
    }

    public void insertContact(String name, String phoneNumber) {
        if (size < contacts.length) {
            contacts[size++] = new Contact(name, phoneNumber);
            JOptionPane.showMessageDialog(null, "Contact added successfully!");
        } else {
            JOptionPane.showMessageDialog(null, "Phonebook is full!");
        }
    }

    public Contact searchContact(String name) {
        for (int i = 0; i < size; i++) {
            if (contacts[i].name.equalsIgnoreCase(name)) {
                return contacts[i];
            }
        }
        return null; // Contact not found
    }

```

honebook / Phonebook / src / **Phonebook.java**

ode | Blame | 63 lines (55 loc) · 1.94 KB | Code 55% faster with GitHub Copilot

```java
public class Phonebook {
    public void displayContacts(JTextArea textArea) {

    public boolean deleteContact(String name) {
        for (int i = 0; i < size; i++) {
            if (contacts[i].name.equalsIgnoreCase(name)) {
                for (int j = i; j < size - 1; j++) {
                    contacts[j] = contacts[j + 1];
                }
                contacts[--size] = null; // Clear last position
                return true; // Contact deleted
            }
        }
        return false; // Contact not found
    }

    public boolean updateContact(String name, String newPhoneNumber) {
        Contact contact = searchContact(name);
        if (contact != null) {
            contact.phoneNumber = newPhoneNumber;
            return true; // Update successful
        }
        return false; // Contact not found
    }

    public void sortContacts() {
        Arrays.sort(contacts, 0, size, (c1, c2) -> c1.name.compareToIgnoreCase(c2.name));
    }
}
```

Phonebook Project/Assignment

PhonebookGUI.java

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class PhonebookGUI extends JFrame {
    private Phonebook phonebook;
    private JTextArea textArea;
    private JTextField nameField;
    private JTextField phoneField;

    public PhonebookGUI() {
        phonebook = new Phonebook(100);
        setTitle("Phonebook Application");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        textArea = new JTextArea();
        textArea.setEditable(false);
        JScrollPane scrollPane = new JScrollPane(textArea);
        add(scrollPane, BorderLayout.CENTER);

        JPanel inputPanel = new JPanel();
        inputPanel.setLayout(new GridLayout(3, 2));

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

Phonebook Project/Assignment

```
        inputPanel.add(nameField);

        inputPanel.add(new JLabel("Phone Number:"));
        phoneField = new JTextField();
        inputPanel.add(phoneField);

        JButton insertButton = new JButton("Insert");
        insertButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String name = nameField.getText();
                String phoneNumber = phoneField.getText();
                phonebook.insertContact(name, phoneNumber);
                phonebook.displayContacts(textArea);
                clearFields();
            }
        });
        inputPanel.add(insertButton);

        JButton searchButton = new JButton("Search");
        searchButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String name = nameField.getText();
                Contact contact = phonebook.searchContact(name);
                if (contact != null) {
                    JOptionPane.showMessageDialog(null, "Found: " + contact.name + ": " + contact.phoneNumber);
```

```
                } else {
                    JOptionPane.showMessageDialog(null, "Contact not found.");
                }
            }
        });
        inputPanel.add(searchButton);

        JButton deleteButton = new JButton("Delete");
        deleteButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String name = nameField.getText();
                if (phonebook.deleteContact(name)) {
                    JOptionPane.showMessageDialog(null, "Contact deleted.");
                } else {
                    JOptionPane.showMessageDialog(null, "Contact not found.");
                }
                phonebook.displayContacts(textArea);
                clearFields();
            }
        });
        inputPanel.add(deleteButton);

        JButton updateButton = new JButton("Update");
        updateButton.addActionListener(new ActionListener() {
```

Phonebook Project/Assignment

```
81              @Override
82              public void actionPerformed(ActionEvent e) {
83                  String name = nameField.getText();
84                  String newPhoneNumber = phoneField.getText();
85                  if (phonebook.updateContact(name, newPhoneNumber)) {
86                      JOptionPane.showMessageDialog(null, "Contact updated.");
87                  } else {
88                      JOptionPane.showMessageDialog(null, "Contact not found.");
89                  }
90                  phonebook.displayContacts(textArea);
91                  clearFields();
92              }
93          });
94          inputPanel.add(updateButton);
95
96          JButton sortButton = new JButton("Sort");
97          sortButton.addActionListener(new ActionListener() {
98              @Override
99              public void actionPerformed(ActionEvent e) {
100                 phonebook.sortContacts();
101                 phonebook.displayContacts(textArea);
102                 JOptionPane.showMessageDialog(null, "Contacts sorted.");
103             }
104         });
```

```
104             });
105             inputPanel.add(sortButton);
106
107             add(inputPanel, BorderLayout.SOUTH);
108         }
109
110         private void clearFields() {
111             nameField.setText("");
112             phoneField.setText("");
113         }
114
115         public static void main(String[] args) {
116             SwingUtilities.invokeLater(() -> {
117                 PhonebookGUI gui = new PhonebookGUI();
118                 gui.setVisible(true);
119             });
120         }
121     }
```

Phonebook Project/Assignment

Important Features:

1. **Add Contact:** Allows users to input a name, address, and phone number to add a new contact. After adding a contact, the list of contacts is sorted alphabetically by name using the **Merge Sort** algorithm.
2. **Search Contact:** Enables users to search for a contact by name. If found, it displays the contact's details; otherwise, it notifies that the contact is not found.
3. **Display All Contacts:** Displays all the contacts in the phonebook, showing the name, address, and phone number of each contact.
4. **Update Contact:** Users can update the details of a contact by selecting the contact's index. After updating, the list is re-sorted alphabetically.
5. **Delete Contact:** Allows users to delete a contact by selecting its index. The contacts are shifted to fill the gap left by the deleted contact.
6. **Exit:** Allows the user to exit the program.