<u>Phonebook Class</u>

- Attributes:
    - contacts : A list of contacts.

Methods:

1. Phonebook Constructor
    - Initialize contacts as an empty list.

---

2. insertContact(name, phone)
    - Create a new contact with the given name and phone.
    - Add the new contact to the list contacts.
    - Show a message: "Contact [name] added."

---

3. searchContact(name)
    - For each contact in contacts:
        - If the contact’s name matches the given name (case-insensitive):
            - Return the contact.
    - If no contact is found, return null.

---

4. deleteContact(name)
    - Initialize toRemove as null.
    - For each contact in contacts:
        - If the contact’s name matches the given name (case-insensitive):
            - Assign the contact to toRemove.
            - Break out of the loop.
    - If toRemove is not null, remove the contact from contacts.
        - Show a message: "Contact [name] deleted."

    - Otherwise, show a message: "Contact [name] not found."

---

5. updateContact(name, newName, newPhone)
    - For each contact in contacts:
        - If the contact’s name matches the given name (case-insensitive):
            - If newName is not empty or null, update the contact's name.
            - If newPhone is not empty or null, update the contact's phone number.
            - Show a message: "Contact [name] updated."
            - Exit the method.
    - If no matching contact is found, show a message: "Contact [name] not found."

---

6. getContacts()
    - Return the list of contacts.

---

7. sortContacts()
    - Sort contacts alphabetically by the contact name.
    - Show a message: "Contacts sorted."

---

8. analyzeSearchEfficiency()
    - Return the string: "The search operation is O(n), where n is the number of contacts."

The Contact class is a simple data structure to hold the contact's name and phone number. Here's the breakdown in pseudocode:

<u>Contact Class</u>

- **Attributes:**
    - name: The contact's name.
    - phone: The contact's phone number.

Methods:

1. **Contact Constructor (name, phone)**
    - Set the name attribute to the given name.
    - Set the phone attribute to the given phone.

---

2. **toString()**
    - Return a string in the format: [name]: [phone].

PhonebookApp Class (extends JFrame)

- **Attributes:**
    - phonebook: Instance of Phonebook class to store contacts.
    - displayArea: A text area to display the list of contacts.

Constructor: PhonebookApp()

1. Initialize the phonebook object.
2. Set up the application window:
    - Title: "Phonebook Application"
    - Size: 400x400 pixels
    - Default close operation: EXIT_ON_CLOSE
    - Layout: BorderLayout
3. **Set the background color of the app to aqua.**
4. **Title Panel:**
    - Create a title panel with the text "Namibian Phonebook" centered.
    - Set background color and font styling.

    - Add the title panel to the top of the window.

5. **Display Area:**
    - Create a non-editable text area to display contacts.
    - Set font and background styling.
    - Add a scroll pane to the center of the window containing the text area.
6. **Button Panel:**
    - Create a grid layout for the buttons (two columns, dynamic rows).
    - Add space around the panel using padding.

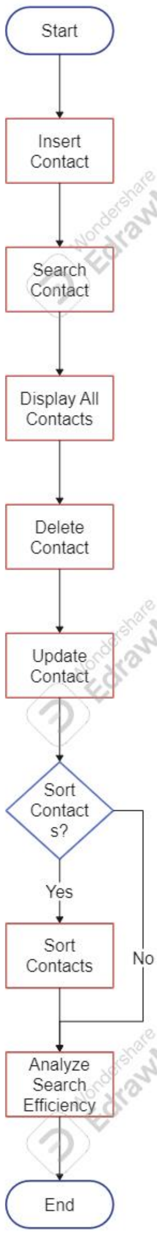
Buttons and Actions:

- **Insert Contact:**
    - Show dialogs to input contact name and phone number.
    - Insert the contact into the phonebook.
- **Search Contact:**
    - Show a dialog to input the name to search.
    - Display the contact if found, or show "Contact not found."
- **Display All Contacts:**
    - Clear the display area.
    - Retrieve all contacts from the phonebook.
    - If the phonebook is empty, display "Phonebook is empty."
    - Otherwise, display each contact.
- **Delete Contact:**
    - Show a dialog to input the name of the contact to delete.
    - Remove the contact from the phonebook.
- **Update Contact:**
    - Show dialogs to input the name of the contact to update.
    - Optionally enter a new name or phone number (leave empty to keep unchanged).

- **Sort Contacts:**
    - Sort the contacts alphabetically by name.
- **Analyze Efficiency:**
    - Display the search efficiency analysis (O(n)).

Main Method

- Use SwingUtilities.invokeLater to ensure the app's GUI is created on the Event Dispatch Thread (EDT).
- Create an instance of PhonebookApp and make it visible.



GROUP MEMBERS

Student name	Student Number
1.joel Lydia (leader)	224091417
2.Johannes Helalia	224066412
3.Mariane Karokoto	224003216
4.Moses Lasarus	224067834
5.Kapusa Kletus Kapusa	224012193
6.Sheehama Nhipandwa S	224085816