DSA PROJECT

GROUP 83

Implementation of a phonebook App system

|  |  |
| --- | --- |
| **GROUP MEMBER** | **STUDENT NUMBER** |
| LUCIO SERAFIM | 224095048 |
| Tk Hambira | 220997755 |
| Martin Nginyengwa | 223038245 |
| Kauarive Katjaita | 223077267 |
| Amilkar Manuel | 224090232 |
| Vitjiukua Kaereho | 223053317 |

**SECTION A**

- **Data Structure:** The phonebook application will use a linked list data structure. A linked list will allow us to dynamically manage contacts, making it easier to insert and delete contacts without worrying about fixed sizes.

**\*PhoneBook App System**

**Module/function Pseudocode Representation**

1. *Insert Contact*

***-*** Allows us to add a new contact to the phonebook

***FUNCTION InsertContact(head, newContact)***

***IF head IS NULL THEN***

***head = newContact***

***ELSE***

***current = head***

***WHILE current.next IS NOT NULL DO***

***current = current.next***

***ENDWHILE***

***current.next = newContact***

***ENDIF***

***RETURN head***

***END FUNCTION***

1. *Search Contact*

*-*Finds a contact by name

***FUNCTION SearchContact(head, name)***

***current = head***

***WHILE current IS NOT NULL DO***

***IF current.name == name THEN***

***RETURN current***

***ENDIF***

***current = current.next***

***ENDWHILE***

***RETURN "Contact not found"***

***END FUNCTION***

1. *Display All Contacts*

*-*Shows all the contacts in the phonebook

***FUNCTION DisplayContacts(head)***

***current = head***

***WHILE current IS NOT NULL DO***

***PRINT current.name + ": " + current.phoneNumber***

***current = current.next***

***ENDWHILE***

***END FUNCTION***

1. *Delete Contact*

*-*Removes a contact from the phonebook

***FUNCTION DeleteContact(head, name)***

***IF head IS NULL THEN***

***PRINT "Phonebook is empty"***

***RETURN head***

***ENDIF***

***IF head.name == name THEN***

***head = head.next***

***RETURN head***

***ENDIF***

***current = head***

***WHILE current.next IS NOT NULL DO***

***IF current.next.name == name THEN***

***current.next = current.next.next***

***RETURN head***

***ENDIF***

***current = current.next***

***ENDWHILE***

***PRINT "Contact not found"***

***RETURN head***

***END FUNCTION***

1. *Update Contact*

*-*Changes the details of an existing contact

***FUNCTION UpdateContact(head, name, newContact)***

***current = head***

***WHILE current IS NOT NULL DO***

***IF current.name == name THEN***

***current.name = newContact.name***

***current.phoneNumber = newContact.phoneNumber***

***RETURN***

***ENDIF***

***current = current.next***

***ENDWHILE***

***PRINT "Contact not found"***

***END FUNCTION***

1. *Sort Contacts (optional)*

*-*Organizes the contacts in alphabetical order by name

***FUNCTION SortContacts(head)***

***IF head IS NULL OR head.next IS NULL THEN***

***RETURN head***

***sorted = NULL***

***current = head***

***WHILE current IS NOT NULL DO***

***next = current.next***

***sorted = SortedInsert(sorted, current)***

***current = next***

***ENDWHILE***

***RETURN sorted***

***END FUNCTION***

***FUNCTION SortedInsert(sorted, newContact)***

***IF sorted IS NULL OR sorted.name >= newContact.name THEN***

***newContact.next = sorted***

***RETURN newContact***

***ENDIF***

***current = sorted***

***WHILE current.next IS NOT NULL AND current.next.name < newContact.name DO***

***current = current.next***

***ENDWHILE***

***newContact.next = current.next***

***current.next = newContact***

***RETURN sorted***

***END FUNCTION***

1. *Analyze Search Efficiency*

*-*Measures how long the search operation takes

***FUNCTION AnalyzeSearchEfficiency(head, name)***

***startTime = CURRENT\_TIME***

***SearchContact(head, name)***

***endTime = CURRENT\_TIME***

***PRINT "Search Time: " + (endTime - startTime)***

***END FUNCTION***

**Efficiency Analysis**

* Insertion: O(n) (average) since we traverse to the end of the list.*

* Search: O(n) (linear) since we may need to check each contact.*

* Display: O(n) as we go through the entire list.*

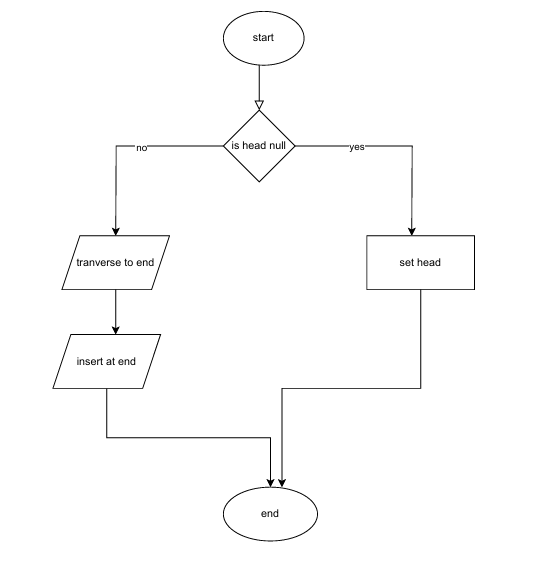
* Deletion: O(n) for searching.*

* Update: O(n) to find the contact.*

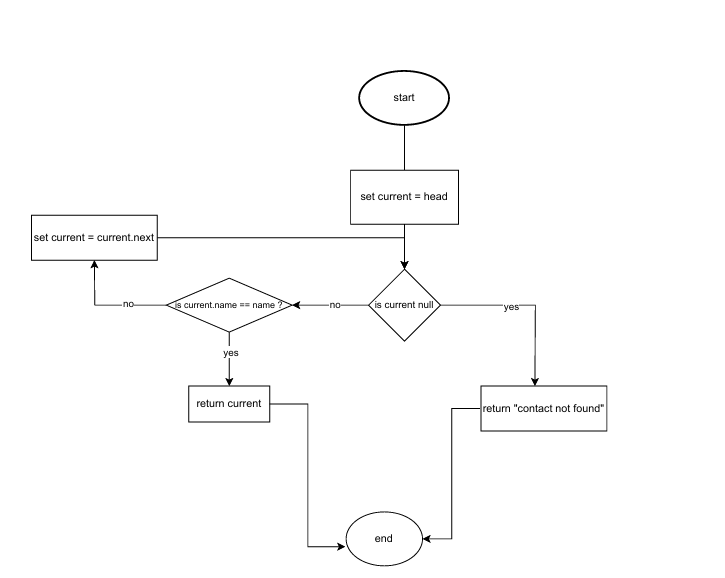
* Sort: O(n^2) (insertion sort) for linked list; could use merge sort for better efficiency.*

**Flowchart Representation**

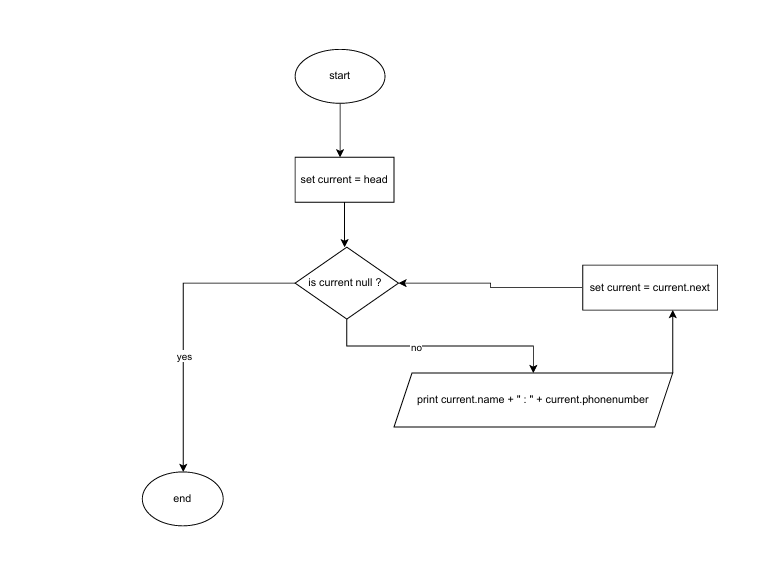
1. ***Insert Contact***

******

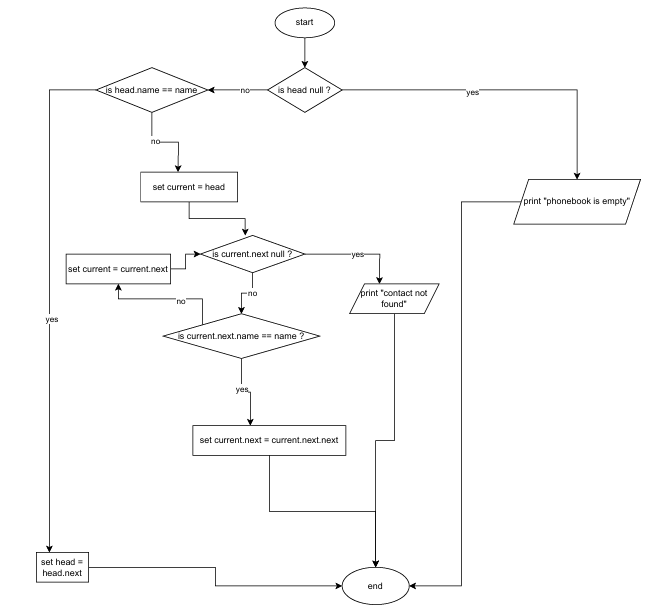
1. ***Search Contact***

******

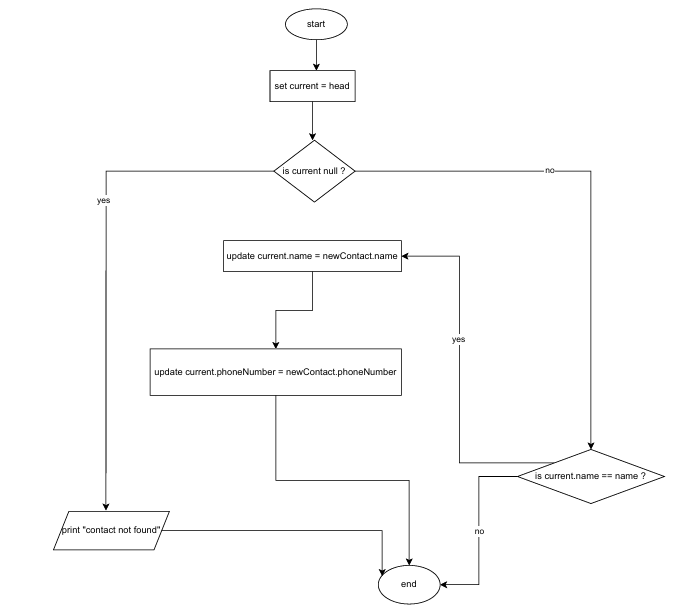
1. ***Display All Contacts***



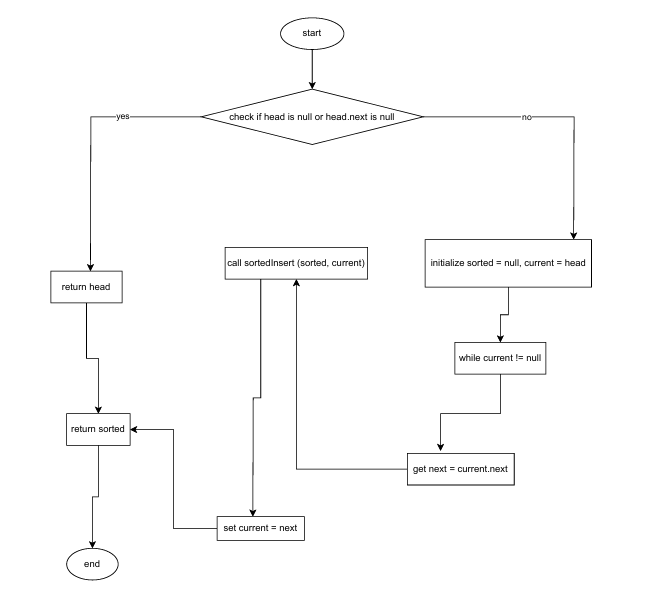
1. ***Delete Contact***



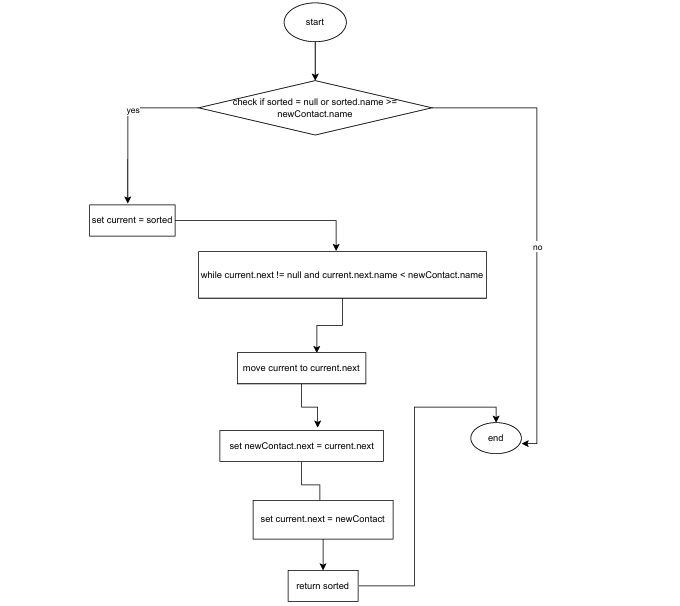
1. **Update Contact**

****

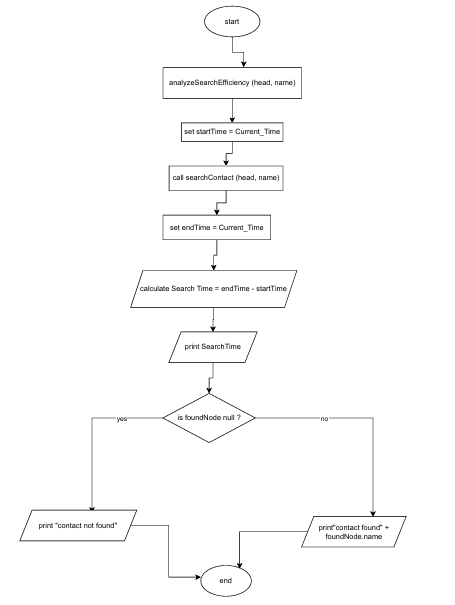
1. ***Sort Contacts (optional)***
2. **For sorted contacts(lone diagram)**

****

1. **For SortedInsert**

******

***7.Analyze Search Efficiency***

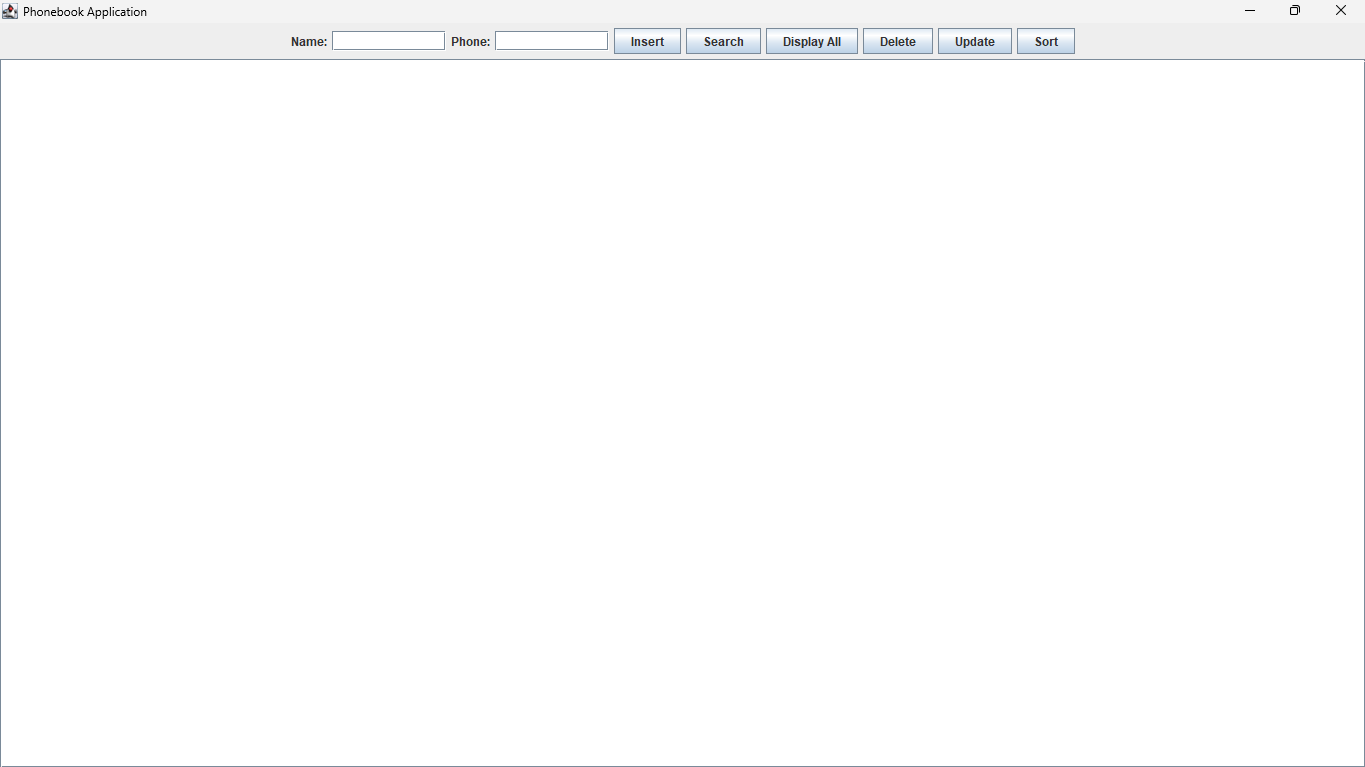


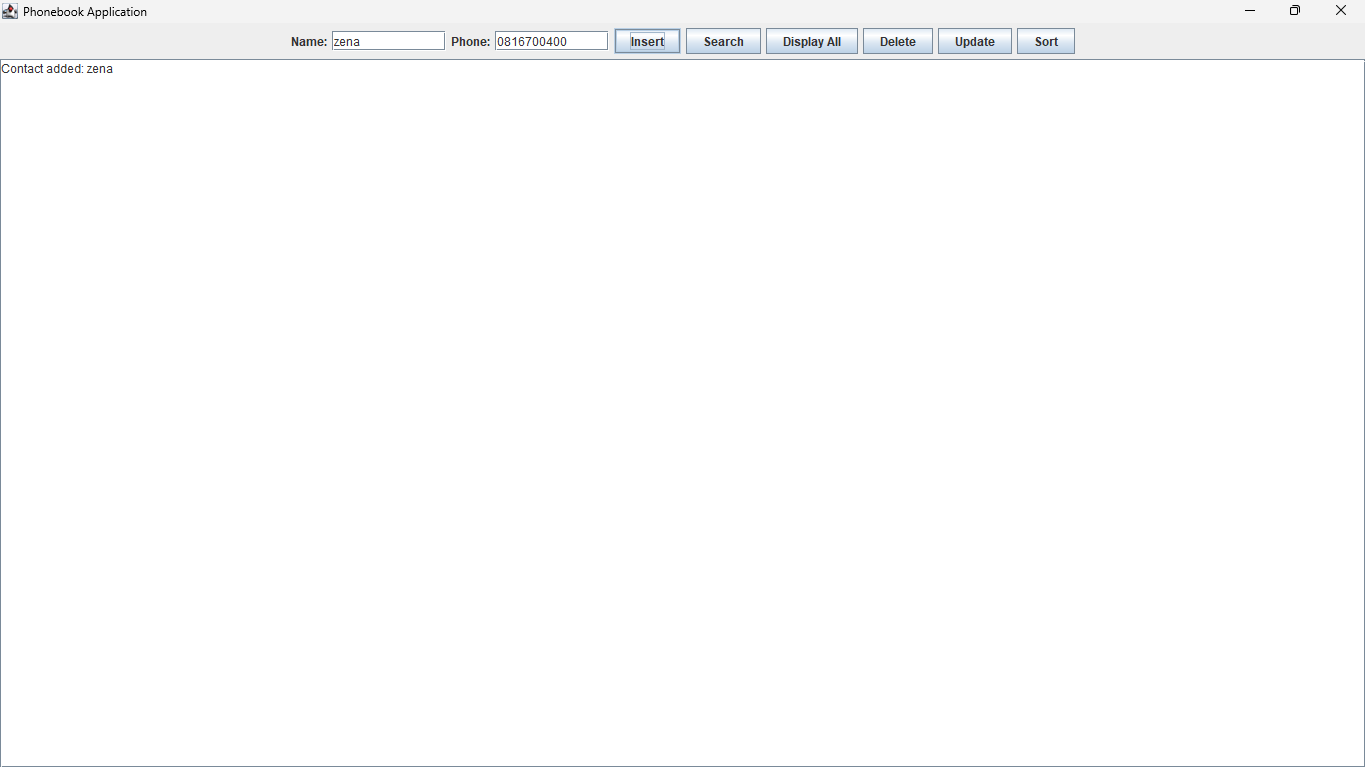
**SECTION B**

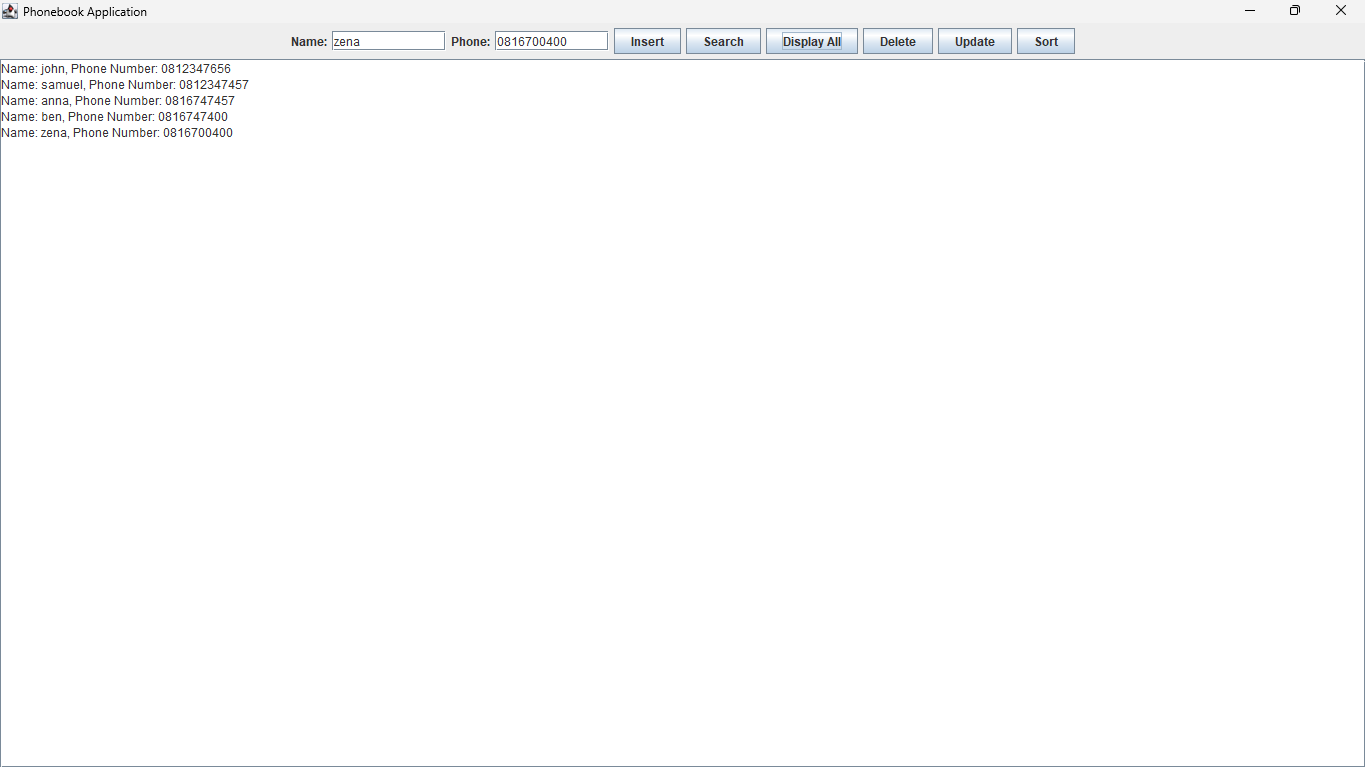
***- Java implementation of the program designed in section A.***

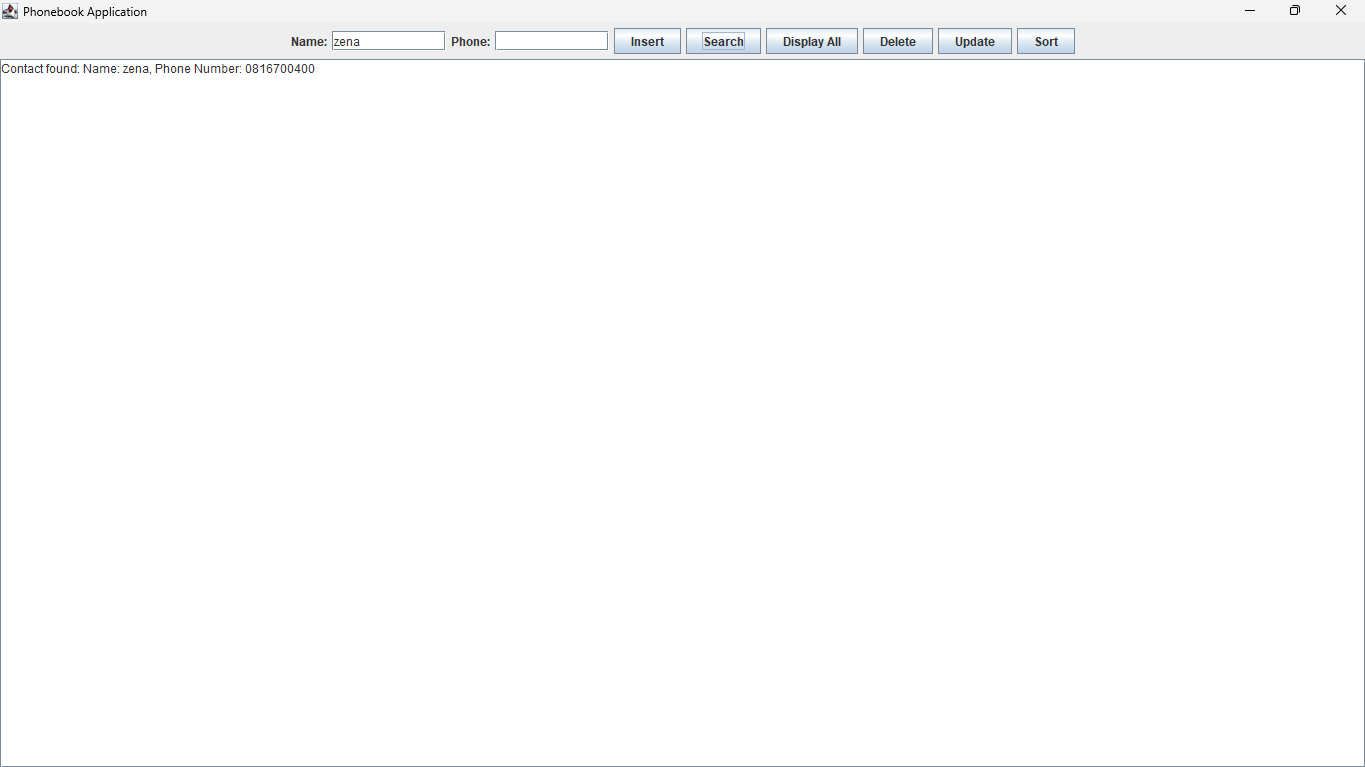
import javax.swing.\*; // Importing Swing components for GUI  
import java.awt.\*; // Importing AWT for layout managers  
import java.awt.event.\*; // Importing event handling classes  
import java.util.LinkedList; // Importing LinkedList for dynamic array  
import java.util.List; // Importing List interface  
  
public class PhoneBookApp extends JFrame {  
  
 // Contact Class: Represents an individual contact in the phonebook  
 private static class Contact {  
 private String name; // Name of the contact  
 private String phoneNumber; // Phone number of the contact  
  
 // Constructor to initialize the contact  
 public Contact(String name, String phoneNumber) {  
 this.name = name;  
 this.phoneNumber = phoneNumber;  
 }  
  
 // Getters and setters for name and phone number  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getPhoneNumber() {  
 return phoneNumber;  
 }  
  
 public void setPhoneNumber(String phoneNumber) {  
 this.phoneNumber = phoneNumber;  
 }  
  
 // toString method for displaying contact information  
 @Override  
 public String toString() {  
 return "Name: " + name + ", Phone Number: " + phoneNumber;  
 }  
 }  
  
 // PhoneBook class: Manages a collection of contacts  
 private List<Contact> contacts; // List to store contacts  
  
 // Constructor for PhoneBookApp  
 public PhoneBookApp() {  
 contacts = new LinkedList<>(); // Initialize the list of contacts  
 initializeUI(); // Set up the user interface  
 }  
  
 // Method to initialize the user interface  
 private void initializeUI() {  
 setTitle("Phonebook Application"); // Set the title of the window  
 setSize(400, 400); // Set the size of the window  
 setDefaultCloseOperation(*EXIT\_ON\_CLOSE*); // Exit the application when the window is closed  
 setLayout(new BorderLayout()); // Use a border layout  
  
 // Input Panel: For user input and buttons  
 JPanel inputPanel = new JPanel(); // Create a new panel for input  
 JTextField nameField = new JTextField(10); // Text field for entering names  
 JTextField phoneField = new JTextField(10); // Text field for entering phone numbers  
 JTextArea displayArea = new JTextArea(); // Text area to display output  
 displayArea.setEditable(false); // Make display area non-editable  
  
 // Create buttons for various functionalities  
 JButton insertButton = new JButton("Insert");  
 JButton searchButton = new JButton("Search");  
 JButton displayButton = new JButton("Display All");  
 JButton deleteButton = new JButton("Delete");  
 JButton updateButton = new JButton("Update");  
 JButton sortButton = new JButton("Sort");  
  
 // Action listeners for button clicks  
 insertButton.addActionListener(e -> addContact(nameField.getText(), phoneField.getText(), displayArea));  
 searchButton.addActionListener(e -> searchContact(nameField.getText(), displayArea));  
 displayButton.addActionListener(e -> displayAllContacts(displayArea));  
 deleteButton.addActionListener(e -> deleteContact(nameField.getText(), displayArea));  
 updateButton.addActionListener(e -> updateContact(nameField.getText(), phoneField.getText(), displayArea));  
 sortButton.addActionListener(e -> sortContacts(displayArea));  
  
 // Add components to the input panel  
 inputPanel.add(new JLabel("Name:")); // Label for name input  
 inputPanel.add(nameField); // Add name field  
 inputPanel.add(new JLabel("Phone:")); // Label for phone number input  
 inputPanel.add(phoneField); // Add phone field  
 inputPanel.add(insertButton); // Add insert button  
 inputPanel.add(searchButton); // Add search button  
 inputPanel.add(displayButton); // Add display button  
 inputPanel.add(deleteButton); // Add delete button  
 inputPanel.add(updateButton); // Add update button  
 inputPanel.add(sortButton); // Add sort button  
  
 // Add panels to the frame  
 add(inputPanel, BorderLayout.*NORTH*); // Add input panel to the top  
 add(new JScrollPane(displayArea), BorderLayout.*CENTER*); // Add display area to the center  
 }  
  
 // 1. Method to add a contact  
 private void addContact(String name, String phoneNumber, JTextArea displayArea) {  
 if (name.isEmpty() || phoneNumber.isEmpty()) {  
 displayArea.setText("Name and Phone Number cannot be empty."); // Error message if fields are empty  
 return;  
 }  
 contacts.add(new Contact(name, phoneNumber)); // Create and add new contact  
 displayArea.setText("Contact added: " + name); // Feedback message  
 }  
  
 // 2. Method to search for a contact  
 private void searchContact(String name, JTextArea displayArea) {  
 for (Contact contact : contacts) {  
 if (contact.getName().equalsIgnoreCase(name)) {  
 displayArea.setText("Contact found: " + contact); // Display found contact  
 return;  
 }  
 }  
 displayArea.setText("Contact not found."); // Error message if not found  
 }  
  
 // 3. Method to display all contacts  
 private void displayAllContacts(JTextArea displayArea) {  
 if (contacts.isEmpty()) {  
 displayArea.setText("Phonebook is empty."); // Message if there are no contacts  
 } else {  
 StringBuilder builder = new StringBuilder();  
 for (Contact contact : contacts) {  
 builder.append(contact).append("\n"); // Append each contact to the display  
 }  
 displayArea.setText(builder.toString()); // Show all contacts  
 }  
 }  
  
 // 4. Method to delete a contact  
 private void deleteContact(String name, JTextArea displayArea) {  
 for (Contact contact : contacts) {  
 if (contact.getName().equalsIgnoreCase(name)) {  
 contacts.remove(contact); // Remove the found contact  
 displayArea.setText("Contact deleted: " + name); // Feedback message  
 return;  
 }  
 }  
 displayArea.setText("Contact not found."); // Error message if not found  
 }  
  
 // 5. Method to update a contact's phone number  
 private void updateContact(String name, String newPhoneNumber, JTextArea displayArea) {  
 for (Contact contact : contacts) {  
 if (contact.getName().equalsIgnoreCase(name)) {  
 contact.setPhoneNumber(newPhoneNumber); // Update phone number  
 displayArea.setText("Contact updated: " + contact); // Feedback message  
 return;  
 }  
 }  
 displayArea.setText("Contact not found."); // Error message if not found  
 }  
  
 // 6. Method to sort contacts  
 private void sortContacts(JTextArea displayArea) {  
 contacts.sort((c1, c2) -> c1.getName().compareToIgnoreCase(c2.getName())); // Sort contacts alphabetically  
 displayArea.setText("Contacts sorted."); // Feedback message  
 }  
  
 // Main method: Entry point of the application  
 public static void main(String[] args) {  
 SwingUtilities.*invokeLater*(() -> {  
 PhoneBookApp app = new PhoneBookApp(); // Create instance of the application  
 app.setVisible(true); // Make the application window visible  
 });  
 }  
}

**Screenshoot of the application**

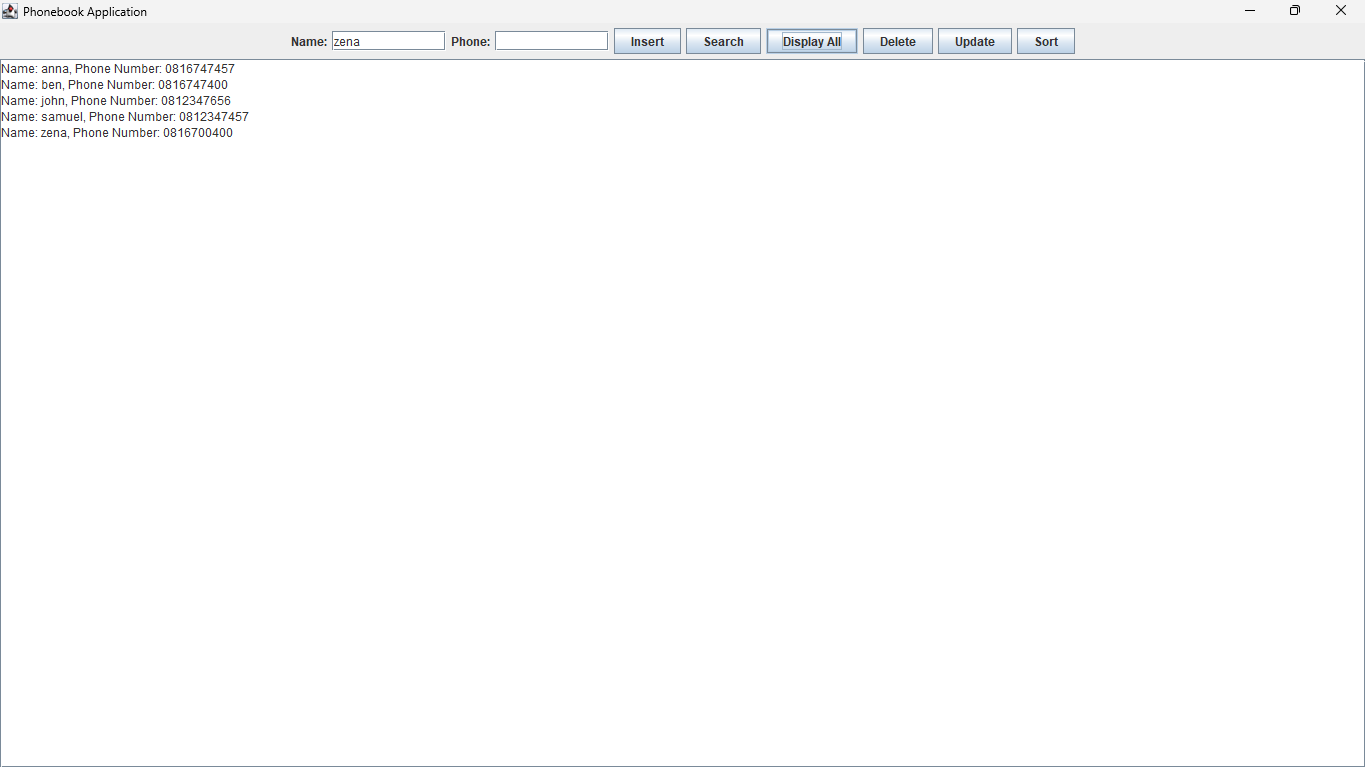
****







**Sorted contacts screenshoot**

****