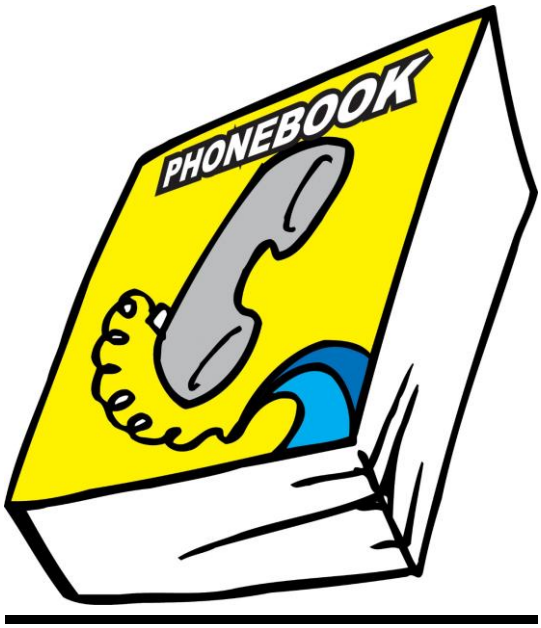
Data Structures and Algorithms 1(DSA521S) : PhoneBook Assignment



Student number	Student name	Contribution to assignment
224046128	Chenai Nyengeterai Chaurura	Pseudocode
219068259	Vijanda Jaruka	No contribution
224080881	Ndasilwohenda Nandiinotya	Pseuodocde
223113263	Donnay Willemse	Java code
223111988	Salom Mwiihangele	Pseuodocde

Pseudocode

```
START
 INITIALIZE ArrayList contacts to store contact details as HashMaps
   METHOD insertContact(name, phone)
      CREATE a new HashMap contact
      SET "name" in contact to name
      SET "phone" in contact to phone
      ADD contact to contacts
   END METHOD
   METHOD searchContact(name)
      FOR EACH contact IN contacts
         IF contact's "name" equals name
            RETURN contact
      END FOR
      RETURN null
   END METHOD
   METHOD displayAllContacts()
      IF contacts is empty
         PRINT "No contacts available in your phonebook."
         RETURN
      END IF
      FOR EACH contact IN contacts
         PRINT "Name: " + contact's "name" + ", Phone: " + contact's "phone"
      END FOR
   END METHOD
   METHOD deleteContact(name)
      FOR i FROM 0 TO size of contacts - 1
         IF contact at index i's "name" equals name
```

```
    REMOVE contact at index i
            RETURN true
    END FOR
    RETURN false
END METHOD
METHOD updateContact(name, newName, newPhone)
    FOR EACH contact IN contacts
        IF contact's "name" equals name (case-insensitive)
            SET "name" in contact to newName
            SET "phone" in contact to newPhone
            RETURN true
    END FOR
    RETURN false
END METHOD
METHOD sortContacts()
    SET n to size of contacts
    FOR i FROM 0 TO n - 2
        FOR j FROM 0 TO n - i - 2
            IF name of contact at j is greater than name of contact at j + 1
                SWAP contact at j with contact at j + 1
        END FOR
    END FOR
END METHOD
METHOD main()
    CREATE Phonebook object phonebook
    CREATE Scanner object scanner
    DECLARE integer choice
    DO
```

```
PRINT menu options (Insert, Search, Display, Delete, Update, Sort, Analyze, Exit)
GET
    Input from user
SWITCH choice
   CASE 1:
      PROMPT user for name and phone number
      CALL phonebook.insertContact with user input
      PRINT "Contacts are  inserted into your phonebook."
   CASE 2:
      PROMPT user for name to search
      CALL phonebook.searchContact with user input
      IF contact found
         PRINT "Contact found in your phonebook: Name and Phone"
      ELSE
         PRINT "Contact not found in your phonebook."
   CASE 3:
      CALL phonebook.displayAllContacts
   CASE 4:
      PROMPT user for name to delete
      CALL phonebook.deleteContact with user input
      IF contact deleted
         PRINT "Contact deleted."
      ELSE
         PRINT "Contact not found."
   CASE 5:
      PROMPT user for name, new name, and new phone number
      CALL phonebook.updateContact with user input
      IF contact updated
```

```
                PRINT "Contact updated in your phonebook."
            ELSE
                PRINT "Contact not found in your phone."
        CASE 6:
            CALL phonebook.sortContacts
            PRINT "Contacts in your phonebook is sorted."
        CASE 7:
            CALL phonebook.analyzeSearchEfficiency
        CASE 8:
            PRINT "Exit."
        DEFAULT:
            PRINT "Invalid input  Please try again."
    END SWITCH
  WHILE choice is not 8
  CLOSE scanner
END METHOD

END
```

Java source code and link to Github:

https://github.com/groupmaste/phonebook

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 */

package com.mycompany.phonebook;

/**
 *
 * @author D.Willemse
 */
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;

public class Phonebook {
    // ArrayList to hold contacts (using HashMap for each contact's details)
    private final ArrayList<HashMap<String, String>> contacts;

    public Phonebook() {
        contacts = new ArrayList<>();
    }
```

```
// 1. Insert Contact
public void insertContact(String name, String phone) {
    HashMap<String, String> contact = new HashMap<>();
    contact.put("name", name);
    contact.put("phone", phone);
    contacts.add(contact);
}

// 2. Search Contact
public HashMap<String, String> searchContact(String name) {
    for (HashMap<String, String> contact : contacts) {
        if (contact.get("name").equalsIgnoreCase(name)) {
            return contact;
        }
    }
    return null;
}

// 3. Display All Contacts
public void displayAllContacts() {
    if (contacts.isEmpty()) {
        System.out.println("No contacts available.");
        return;
    }
```

```
        for (HashMap<String, String> contact : contacts) {
            System.out.println("Name: " + contact.get("name") + ", Phone: " +
contact.get("phone"));
        }
    }

    // 4. Delete Contact
    public boolean deleteContact(String name) {
        for (int i = 0; i < contacts.size(); i++) {
            if (contacts.get(i).get("name").equalsIgnoreCase(name)) {
                contacts.remove(i);
                return true;
            }
        }
        return false;
    }

    // 5. Update Contact
    public boolean updateContact(String name, String newName, String newPhone)
{
        for (HashMap<String, String> contact : contacts) {
            if (contact.get("name").equalsIgnoreCase(name)) {
                contact.put("name", newName);
                contact.put("phone", newPhone);
                return true;
            }
```

```
    }
    return false;
}

// 6. Sort Contacts (Bubble Sort by Name)
public void sortContacts() {
    int n = contacts.size();
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            String name1 = contacts.get(j).get("name");
            String name2 = contacts.get(j + 1).get("name");
            if (name1.compareToIgnoreCase(name2) > 0) {
                // Swap contacts
                HashMap<String, String> temp = contacts.get(j);
                contacts.set(j, contacts.get(j + 1));
                contacts.set(j + 1, temp);
            }
        }
    }
}

// 7. Analyze the Efficiency of the Search Algorithm
// This method provides information about the search algorithm's efficiency
public void analyzeSearchEfficiency() {
    System.out.println("The search algorithm used is linear search.");
```

```
        System.out.println("Time Complexity: O(n) in the worst case, where n is the number of contacts.");
        System.out.println("Best Case: O(1) if the contact is the first element in the list.");
    }

    // Main method to interact with the Phonebook
    public static void main(String[] args) {
        Phonebook phonebook = new Phonebook();
        Scanner scanner = new Scanner(System.in);
        int choice;
        do {
            System.out.println("\nPhonebook Application");
            System.out.println("1. Insert Contact");
            System.out.println("2. Search Contact");
            System.out.println("3. Display All Contacts");
            System.out.println("4. Delete Contact");
            System.out.println("5. Update Contact");
            System.out.println("6. Sort Contacts");
            System.out.println("7. Analyze Search Efficiency");
            System.out.println("8. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            switch (choice) {
```

```
            case 1:
                System.out.print("Enter name: ");
                String name = scanner.nextLine();
                System.out.print("Enter phone number: ");
                String phone = scanner.nextLine();
                phonebook.insertContact(name, phone);
                System.out.println("Contact inserted.");
                break;
            case 2:
                System.out.print("Enter name to search: ");
                name = scanner.nextLine();
                HashMap<String, String> foundContact =
phonebook.searchContact(name);
                if (foundContact != null) {
                    System.out.println("Contact found: Name: " +
foundContact.get("name") + ", Phone: " + foundContact.get("phone"));
                } else {
                    System.out.println("Contact not found.");
                }
                break;
            case 3:
                phonebook.displayAllContacts();
                break;
            case 4:
                System.out.print("Enter name to delete: ");
                name = scanner.nextLine();
```

```
                boolean deleted = phonebook.deleteContact(name);
                if (deleted) {
                    System.out.println("Contact deleted.");
                } else {
                    System.out.println("Contact not found.");
                }
                break;
            case 5:
                System.out.print("Enter current name: ");
                name = scanner.nextLine();
                System.out.print("Enter new name: ");
                String newName = scanner.nextLine();
                System.out.print("Enter new phone number: ");
                String newPhone = scanner.nextLine();
                boolean updated = phonebook.updateContact(name, newName,
newPhone);
                if (updated) {
                    System.out.println("Contact updated.");
                } else {
                    System.out.println("Contact not found.");
                }
                break;
            case 6:
                phonebook.sortContacts();
                System.out.println("Contacts sorted.");
                break;
```

```
            case 7:
                phonebook.analyzeSearchEfficiency();
                break;
            case 8:
                System.out.println("Exiting the application.");
                break;
            default:
                System.out.println("Invalid choice. Please try again.");
        }
    } while (choice != 8);
    scanner.close();
  }
}
```