

Codecrafters Phonebook Application: Functions

1. Insert Contact

Function: Adds a new contact to the phonebook if it doesn't already exist.

Explanation:

- This function checks if the contact already exists by comparing names (ignoring case).
- If the contact doesn't exist, it creates a new `Contact` object and adds it to the list.
- Feedback is provided via a success or error message.

Code block:

```
public void insertContact(String name, String phoneNumber) {  
    for (Contact contact : contacts) {  
        if (contact.getName().equalsIgnoreCase(name)) {  
            System.out.println("Contact already exists.");  
            return;  
        }  
    }  
    Contact newContact = new Contact(name, phoneNumber);  
    contacts.add(newContact);  
    System.out.println("Contact added successfully.");  
}
```

2. Search Contact

Function: Searches for a contact by name and returns their phone number.

Explanation:

- The function loops through the phonebook, checking for a match by name.

- If found, it displays the contact's details; otherwise, it outputs "Contact not found."

Code block:

```
public Contact searchContact(String name) {  
    for (Contact contact : contacts) {  
        if (contact.getName().equalsIgnoreCase(name)) {  
            System.out.println("Contact found: " + contact.getPhoneNumber());  
            return contact;  
        }  
    }  
    System.out.println("Contact not found.");  
    return null;  
}
```

3. Display All Contacts

Function: Displays all contacts stored in the phonebook.

Explanation:

- Checks if the phonebook is empty. If it is, it displays an error message.
- Otherwise, it prints each contact's name and phone number.

Code block:

```
public void displayAllContacts() {  
    if (contacts.isEmpty()) {  
        System.out.println("Phonebook is empty.");  
    } else {  
        for (Contact contact : contacts) {  
            System.out.println(contact.getName() + ": " +  
contact.getPhoneNumber());  
        }  
    }  
}
```

```
}  
}
```

4. Delete Contact

Function: Deletes a contact from the phonebook by name.

Explanation:

- The function uses an iterator to safely remove the contact while looping through the list.
- It provides feedback if the contact is deleted or not found.

Code block:

```
public void deleteContact(String name) {  
    Iterator<Contact> iterator = contacts.iterator();  
    while (iterator.hasNext()) {  
        Contact contact = iterator.next();  
        if (contact.getName().equalsIgnoreCase(name)) {  
            iterator.remove();  
            System.out.println("Contact deleted successfully.");  
            return;  
        }  
    }  
    System.out.println("Contact not found.");  
}
```

5. Update Contact

Function: Updates an existing contact's phone number.

Explanation:

- Searches for the contact by name, and if found, updates the phone number.
- Provides feedback on whether the update was successful, or the contact wasn't found.

Code block:

```
public void updateContact(String name, String newPhoneNumber) {  
    for (Contact contact : contacts) {  
        if (contact.getName().equalsIgnoreCase(name)) {  
            contact.setPhoneNumber(newPhoneNumber);  
            System.out.println("Contact updated successfully.");  
            return;  
        }  
    }  
    System.out.println("Contact not found.");  
}
```

6. Sort Contacts

Function: Sorts the contacts alphabetically by name.

Explanation:

- This function uses the `Comparator` to sort the contacts list alphabetically by name.
- A success message is displayed once the sorting is complete.

Code block:

```
public void sortContacts() {  
    contacts.sort(Comparator.comparing(Contact::getName));  
    System.out.println("Contacts sorted successfully.");  
}
```

7. Save Contacts to File

Function: Saves all contacts to a file for persistence.

Explanation:

- Saves each contact's name and phone number to a file in CSV format.

- Handles potential IO exceptions and provides error messages if needed.

Code block:

```
public void saveContactsToFile(String filename) {  
    try (PrintWriter writer = new PrintWriter(new File(filename))) {  
        for (Contact contact : contacts) {  
            writer.println(contact.getName() + "," + contact.getPhoneNumber());  
        }  
        System.out.println("Contacts saved to file.");  
    } catch (IOException e) {  
        System.out.println("Error saving contacts to file: " + e.getMessage());  
    }  
}
```

8. Load Contacts from File

Function: Loads contacts from a file into the phonebook.

Explanation:

- Reads a file line by line and splits each line into name and phone number.
- Inserts contacts into the phonebook from the file.

Code block:

```
public void loadContactsFromFile(String filename) {  
    try (BufferedReader reader = new BufferedReader(new  
FileReader(filename))) {  
        String line;  
        while ((line = reader.readLine()) != null) {  
            String[] parts = line.split(",");  
            if (parts.length == 2) {  
                insertContact(parts[0], parts[1]);  
            }  
        }  
    }  
}
```

```
        System.out.println("Contacts loaded from file.");  
    } catch (IOException e) {  
        System.out.println("Error loading contacts from file: " +  
e.getMessage());  
    }  
}
```

9. Analyze Search Efficiency

Function: Analyzes how efficient the search operation is by measuring the time taken.

Explanation:

- Measures the time taken for the search operation by using system timestamps.
- Outputs the search time in milliseconds.

Code block:

```
public void analyzeSearchEfficiency(String name) {  
    long startTime = System.nanoTime();  
    searchContact(name);  
    long endTime = System.nanoTime();  
    long duration = (endTime - startTime) / 1000000; // convert to  
milliseconds  
    System.out.println("Search completed in " + duration + " ms.");  
}
```