**Here’s an overview of each component and its functionality:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

A screenshot of a computer

Description automatically generated

**Client Class**

- Attributes:

- ClientMessages[]: Stores messages sent from "Samar".

- messages[]: Stores all messages "Samar" sends, organized by recipient.

- contacts[]: Holds contacts, allowing only three for this client setup.

- messageCount: Tracks the total messages sent by "Samar".

**Functionalities (Menu Options):**

- 1. **Send Message**: Allows "Samar" to send a message to "Samra." Messages are also stored in ClientMessages[]. The client sends the message via a socket connection and receives a response, which is stored as well.

- 2. **Delete Messages**: Deletes a specific message by specifying the contact's name and the message to be deleted.

- 3. **Add Contacts:** Allows adding contacts with a name and phone number.

- 4. **Sending Messages:** Enables sending messages to any added contact in `contacts[]`. The program verifies the contact's existence before sending.

- 5. **Display Messages:** Shows all messages in `ClientMessages[]`.

- 6. **Modify Messages**: Allows "Samar" to modify a previously sent message by specifying the contact and the message content.

- 7**. Display Sent Messages**: Displays messages that have been sent, stored in `messages[]`.

- 8. **Sort Messages**: Sorts messages by `timeStamp` in ascending order.

- 9**. Exit:** Ends the chat by closing the socket connection.

**Server Class**

- Attributes:

**- serverMessages[]**: Stores messages sent from "Samra."

**- messageCount:** Counts the messages sent by the server.

**- Functionality:**

- Listens for incoming messages from "Samar." When a message is received, it prints it and prompts "Samra" to respond. The response is sent back to the client and stored in `serverMessages[]`.

**Contact Class**

- Attributes:

- name: Name of the contact.

- phoneNumber: Contact's phone number.

This class is used to manage the contacts that "Samar" can send messages to.

**MessagingApp Interface**

Defines the methods that the `Client` class implements, such as sendMessage`, displayMessages`, `modifyMessages`, `deleteMessages`, `addContact`, `sendingMessages`, `sentMessages`, and `sortMessages`.

**Messages Class**

- Attributes:

- sender and receiver: The sender and receiver of each message.

- content: The message content.

- timeStamp: The timestamp when the message was created.

- status: Tracks whether the message is "SEEN" or "UNSEEN."

This class encapsulates message details and allows setting and retrieving message content.

**Status Enum**

Defines two possible message states: SEEN and UNSEEN.

Overall Flow

1. The server starts and waits for a connection.

2. The client connects and can choose various options to interact.

3. Messages can be sent, received, modified, deleted, and sorted.

4. The application uses socket communication for real-time messaging, allowing "Samar" and "Samra" to exchange messages until the connection is terminated.

This setup simulates a basic client-server messaging environment with functionality for contact management, message sorting, and message modification.