**ASSIGNMENT #1:**

**Object Oriented Programming**

****

|  |  |
| --- | --- |
| Submitted to: | **Sir Shahid Bhatti** |
| Submitted by: | **Muhammad Ahmed** |
| Roll No: | **Sp24-bse-066** |
| Section: | **B** |

**1. Overview of the Messaging Application**

This is a simple messaging application built using Java, containing three main classes: SMS, MessagingApp, and Driver. The app allows users to add contacts, send messages, view chat history, delete specific messages, and clear chat histories.

**2. Class Breakdown**

**Class 1: SMS**

The SMS class represents individual text messages. It includes information like the message content, sender, receiver, message ID, status (read/unread), and timestamp.

**Attributes:**

msg: The content of the message.

sender & receiver: Represent who sends and receives the message.

messageId: A unique ID for each message.

status: Marks the message as "unread" or "read."

date: Stores the timestamp of when the message is created.

**Methods:**

1. markAsRead(): Changes the status of a message to "read."
2. toString(): Formats and returns the message details for easy viewing.

**Class 2: MessagingApp**

The MessagingApp class handles the application's core features like managing contacts and sending, viewing, and deleting messages.

**Attributes:**

chats[][]: A 2D array to store the SMS messages for each contact.

contacts[][]: Stores contact names and phone numbers.

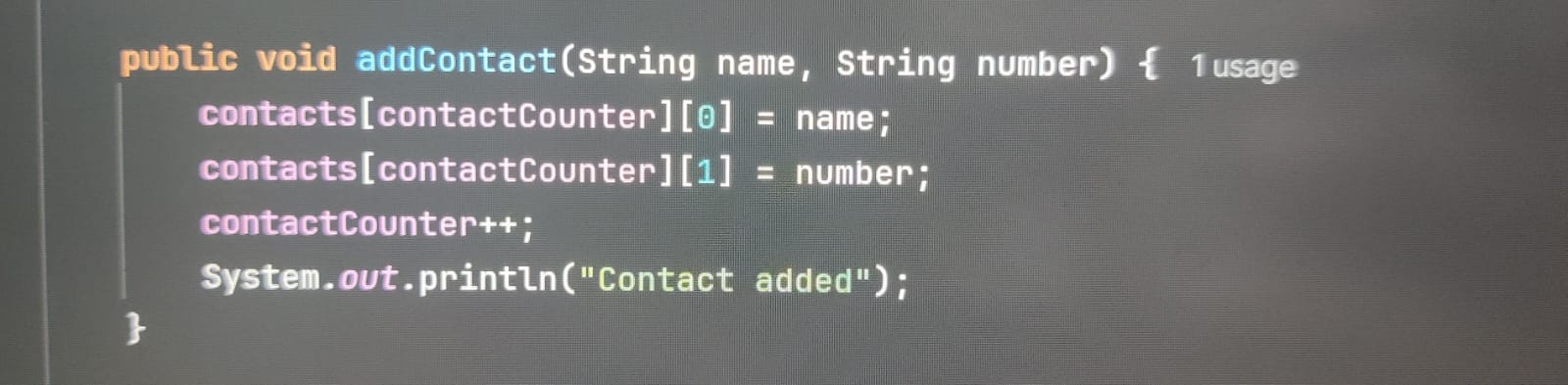
contactCounter: Tracks the number of contacts.

chatCounter[]: Keeps track of the number of messages exchanged with each contact.

**Methods:**

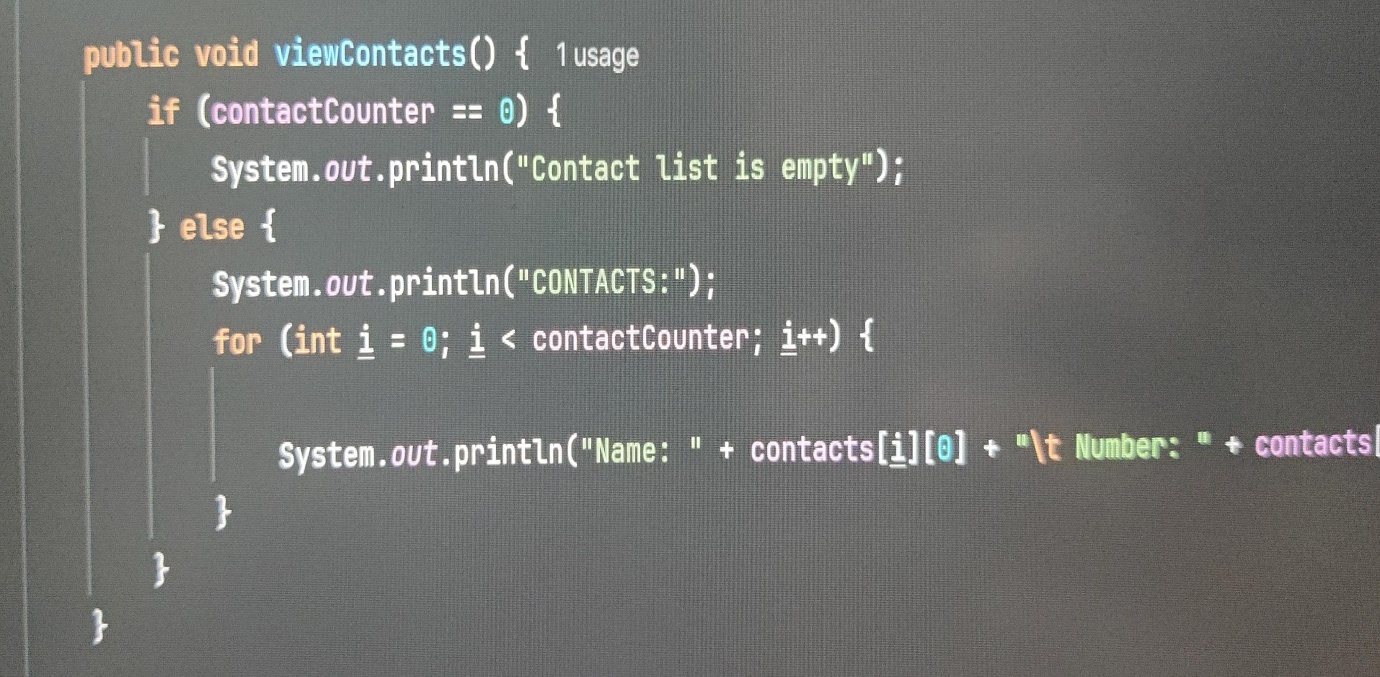
**1. addContact(String name, String number)**

This method adds a new contact to the contacts array using the provided name and number. It also increments the contactCounter to track the total number of contacts.



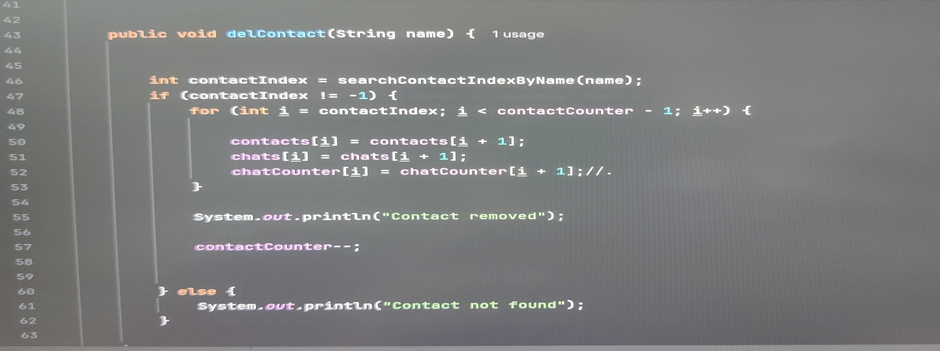
**2. viewContacts():**

Displays all the stored contacts by iterating through the contacts array. If no contacts exist, it prints a message stating that the contact list is empty.



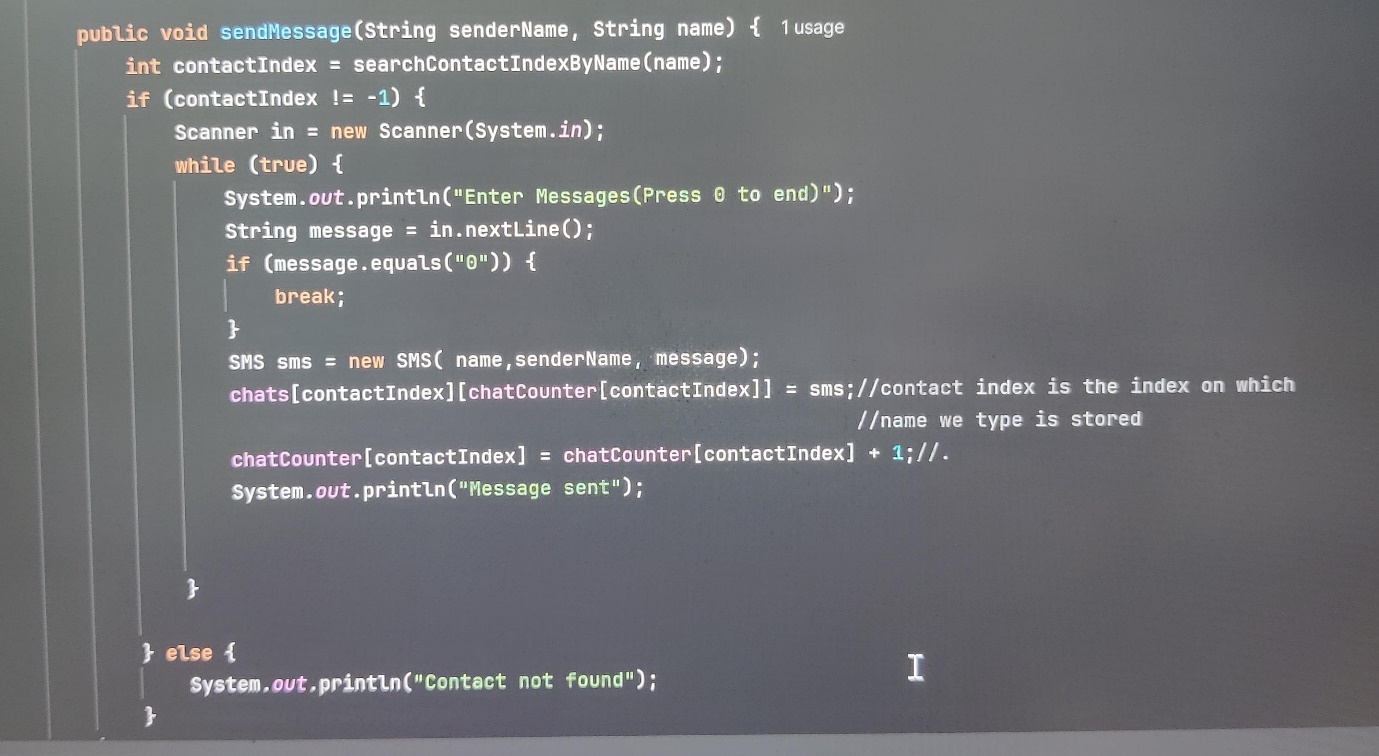
3. delContact(String name)

Removes a contact by name. It finds the contact’s index, shifts all subsequent contacts and their chats, and decrements the contactCounter to update the contact list.



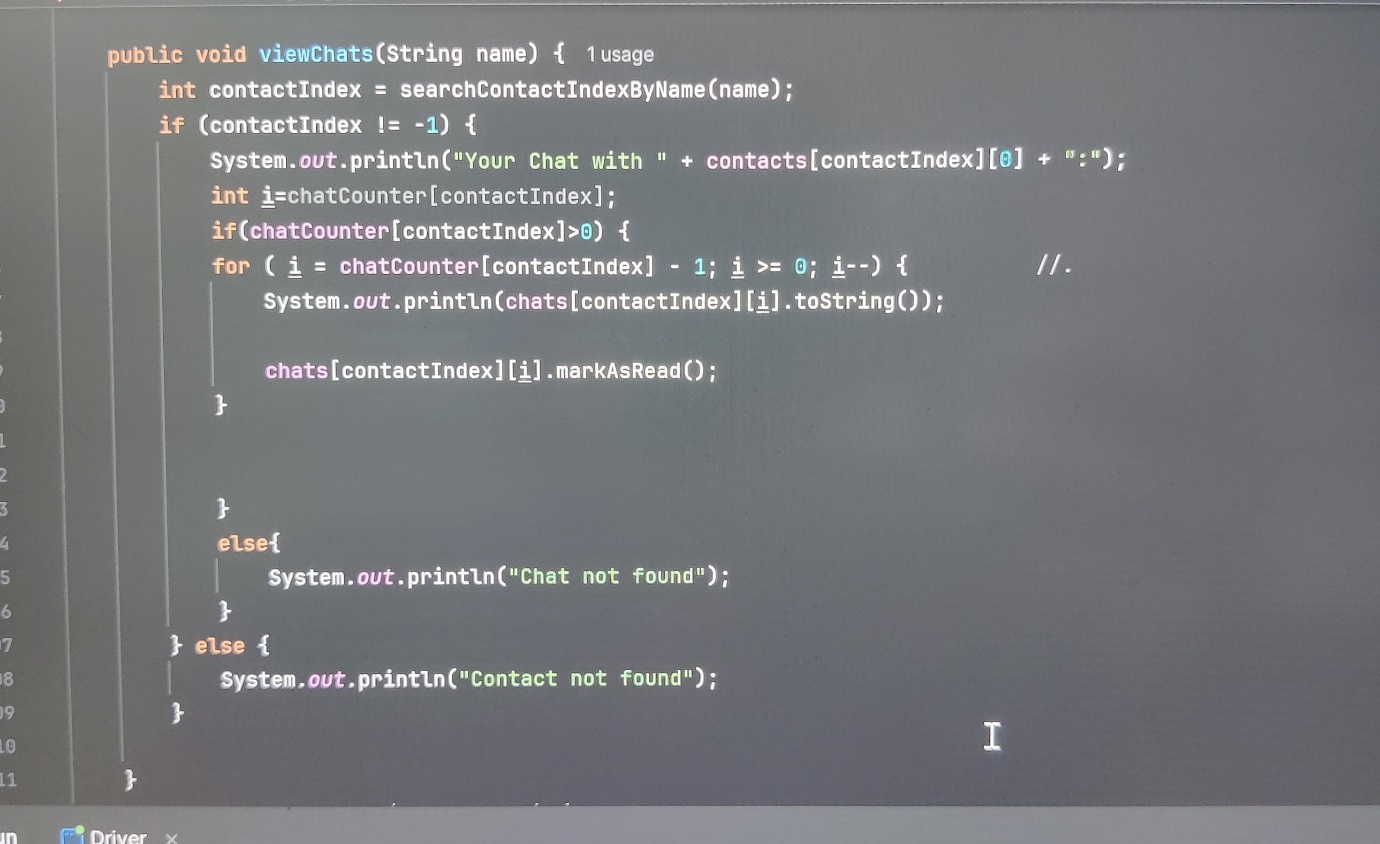
4. sendMessage(String senderName, String name)

Allows the user to send messages to a specific contact by entering the message content. It creates an SMS object, stores it in the appropriate chat array, and updates the chatCounter for that contact.



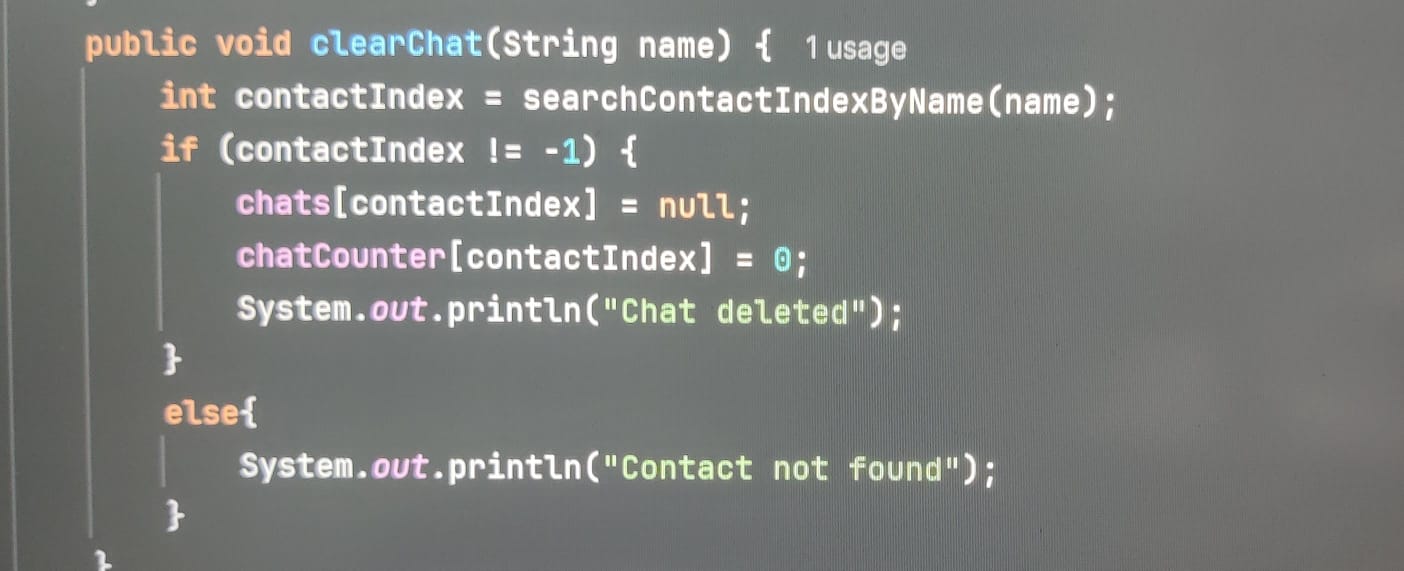
5. viewChats(String name)

Displays all messages exchanged with a specific contact. It retrieves the messages from the chats array and marks unread messages as read.



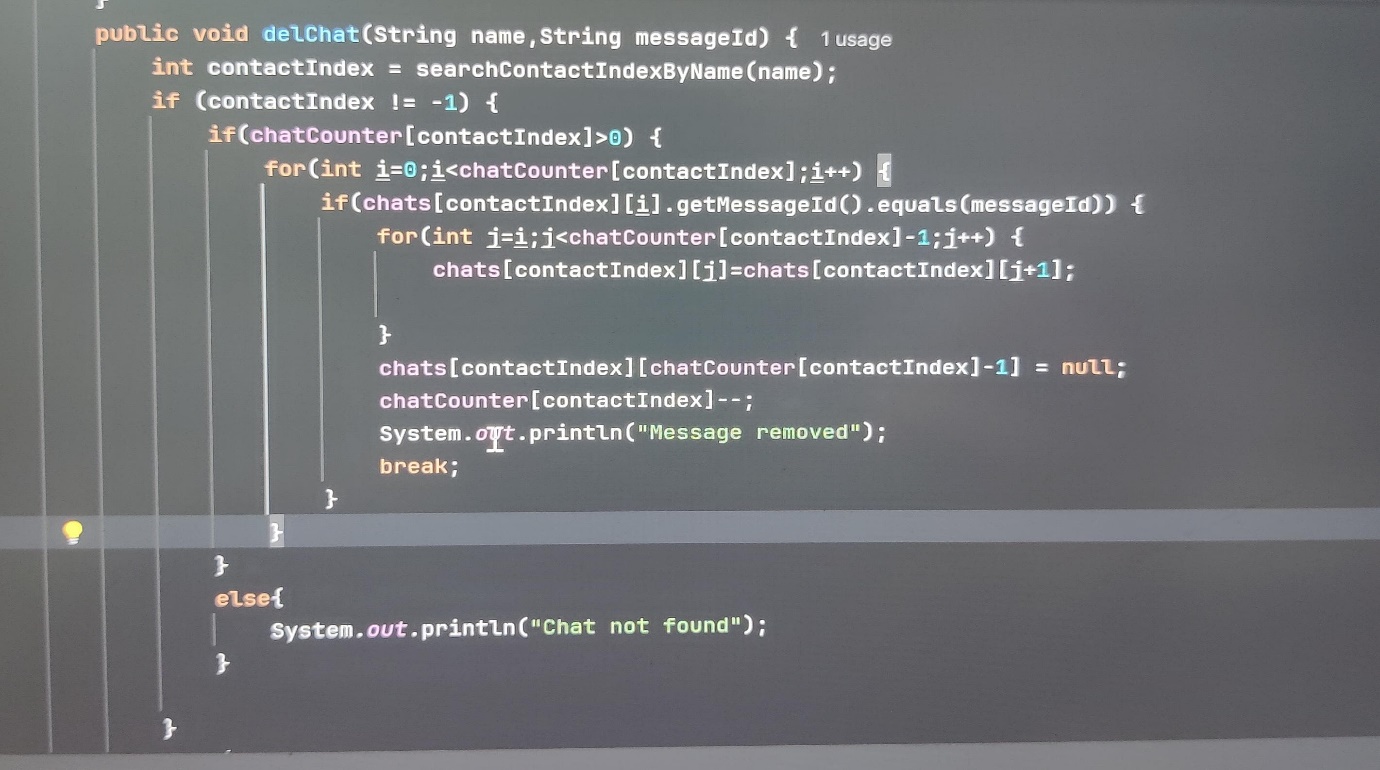
6. clearChat(String name)

Deletes all messages exchanged with a particular contact by setting the chats array for that contact to null and resetting the chatCounter.



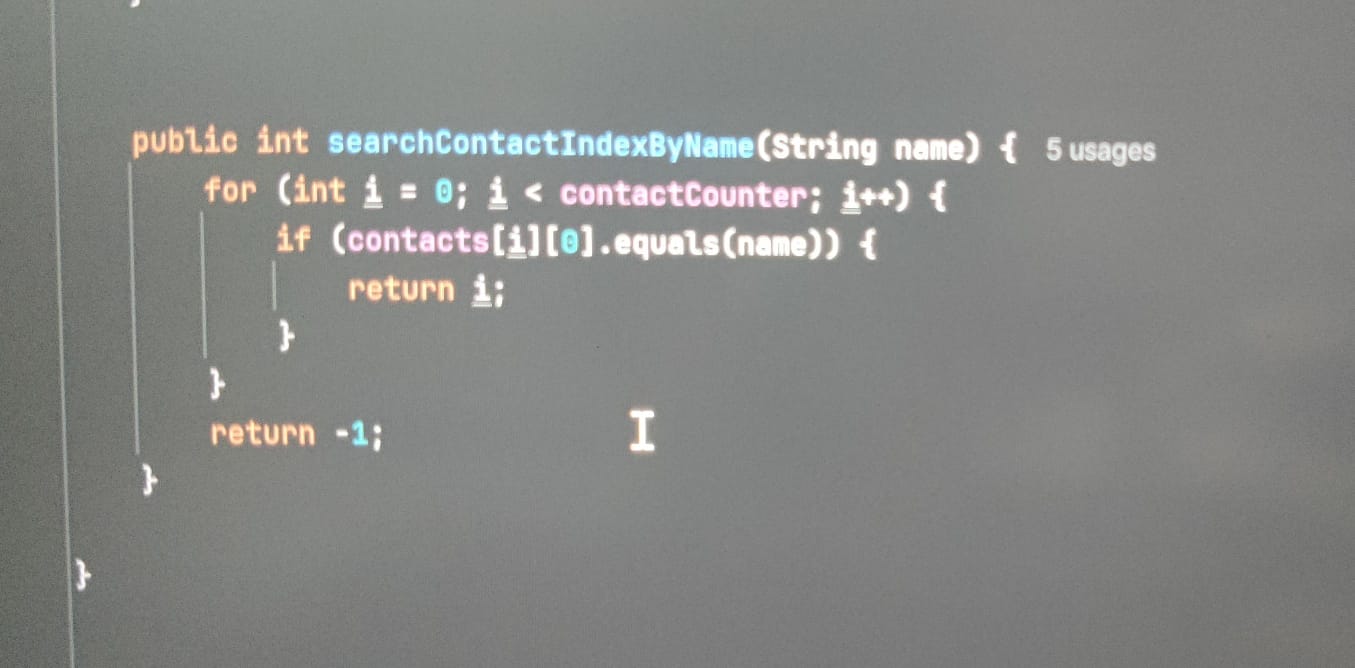
7. delChat(String name, String messageId)

Deletes a specific message in a chat by finding it using the messageId. Once found, it shifts the remaining messages and updates the chatCounter.



8. searchContactIndexByName(String name):

Searches for a contact in the contacts array by name and returns the index of the contact. If the contact is not found, it returns -1.



**Class 3: Driver**

The Driver class is the main class that interacts with the user. It offers a **menu** that allows the user to:

1. Send messages
2. View chat history
3. Add Contact
4. Remove contacts
5. Delete specific messages
6. Clear chat history
7. Exit menu

This class uses a Scanner to take user input and processes the selected options using a switch statement.

Outputs of the code:

