OOP- THEORY ASSIGNMENT #2

Submitted by: sara riaz

Roll no: sp24-bse-110-b

Administrator

Messaging App System –

# Assignment Instructions:

In this assignment, you are tasked with creating a control-based messaging app system using Java Object-Oriented Programming (OOP) principles.

## Requirements:

1. You must create three classes: Message, MessagingServer, and messaging client class

- Delete a message.

- Search message by ID.

- Read all messages.

- Send a message.

2. Implement a menu-driven program in the client class . The user should input their name or ID, and the system should display the appropriate functionality.

3. Each message should have a timestamp of when it was sent using the LocalDateTime class from java.time.

4.implements sockets

5.two way communication

6.reading input in different languages

### Class: ****Message****

* **Attributes:**
  + String messageId: Unique ID for each message.
  + String content: Content of the message.
  + String dateTime: Timestamp of the message.
  + String status: Status (e.g., Sent, Received).
  + static String senderName: Name of the sender ("Aleesha").
  + static String receiverName: Name of the receiver ("Sara").
  + static int id: Counter for generating unique IDs.
* **Functions:**
  + Message(String content, String status): Constructor to initialize a message with content and status.
  + String getContent(): Returns message content.
  + String getMessageId(): Returns the message ID.
  + int compareTo(Message other): Compares two messages by dateTime.
  + String toString(): Formats message details into a readable string.

### Class: ****Client****

* **Attributes:**
  + BufferedReader in: Reads server responses.
  + PrintWriter out: Sends data to the server.
  + Socket socket: Client socket for connection.
  + Scanner scanner: Reads user input.
* **Functions:**
  + Client(): Constructor to initialize and connect to the server.
  + main(String[] args): Main method with a menu for client options.
  + void sendMessage(): Sends a message to the server.
  + void readSentMessages(): Reads sent messages from the server.
  + void readReceivedMessages(): Reads received messages from the server.
  + void readAllMessages(): Reads all messages from the server.
  + void searchMessages(): Searches for messages by keyword.
  + void deleteMessage(): Deletes a message by ID.
  + void displayServerResponse(): Displays server responses.
  + void closeConnection(): Closes the connection.

### Class: ****Server****

* **Attributes:**
  + int PORT: Port number for the server (12345).
  + int MAX\_MESSAGES: Maximum number of messages (200).
  + Message[] sentMessages: Array to store sent messages.
  + Message[] receivedMessages: Array to store received messages.
  + int sentCounter: Counter for sent messages.
  + int receivedCounter: Counter for received messages.
* **Functions:**
  + main(String[] args): Main method to start the server and accept clients.
  + **Inner Class: ClientHandler**
    - **Attributes:**
      * Socket clientSocket: Client connection socket.
      * BufferedReader in: Reads client input.
      * PrintWriter out: Sends data to client.
    - **Functions:**
      * ClientHandler(Socket clientSocket): Constructor to initialize the handler.
      * void run(): Processes client requests in a loop.
      * void handleSendMessage(): Receives and saves a new message.
      * void handleReadSentMessages(): Sends sent messages to the client.
      * void handleReadReceivedMessages(): Sends received messages to the client.
      * void handleReadAllMessages(): Sends all messages to the client in sorted order.
      * void handleSearchMessage(): Searches for messages by keyword.
      * void handleDeleteMessage(): Deletes a message by ID.
      * void handleCloseConnection(): Closes the client connection.

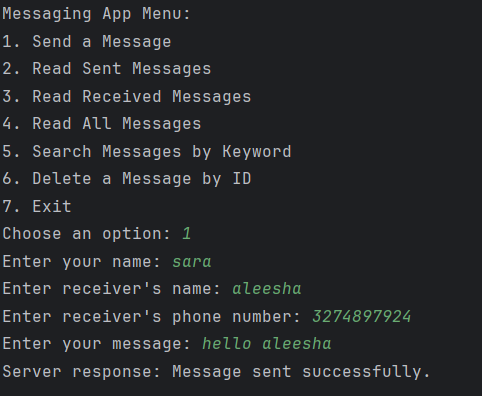
# Java Code:

Uploaded on git hub.

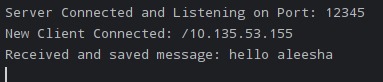
# Output Screenshots:

## Send a Message

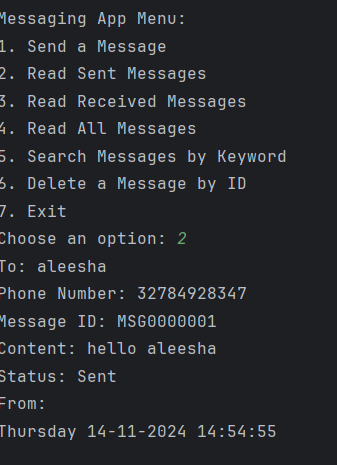
Client Side:



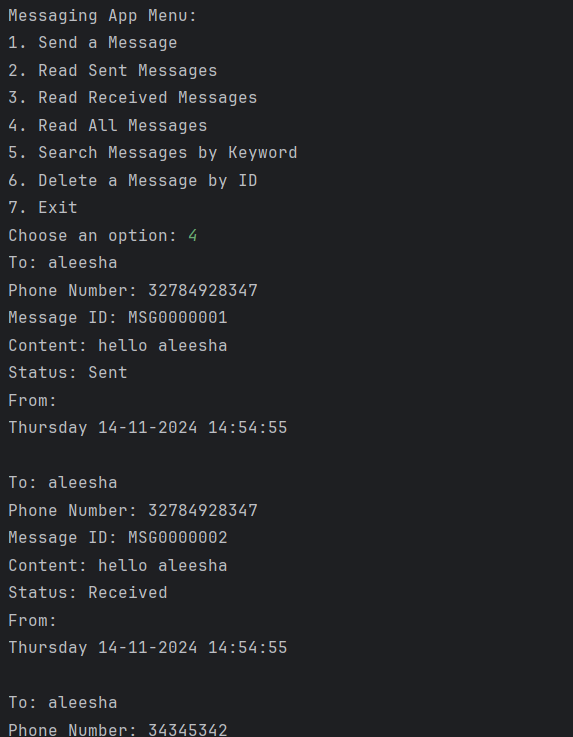
Server Side:



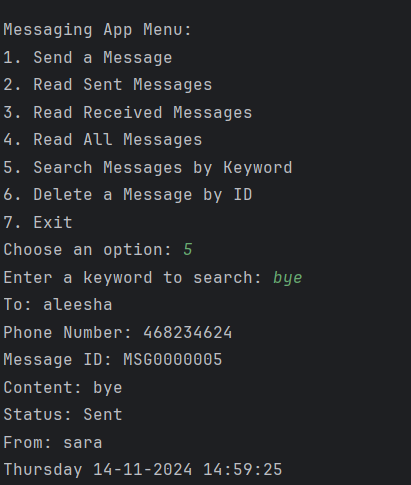
## 2.Read sent messages



## 3.ReadAllMessages



**3.Search Message by keyword:**



## 5.Delete a message by ID

