**OBJECT ORIENTED PROGRAMMING**

**13-11-2024**

**SUBMITTED TO:**

**Muhibba Fatima (SP24-BSE-B-138)**

**Sir Shahid Bhatti**

**Date:**

**SUBMITTED BY:**

**SUBJECT:**

**ASSIGNMENT # 2**

**Explanation of the Code:**

Client and Server Connect using Sockets over the Network:

**Server Setup:**

At the server side, it listens on a specified port which has been identified for listening. The server then waits for a client to connect.

**Client Connection:**

The client initiates communication by creating a socket to the server's IP address and port. When a client requests a connection, the server accepts it and thus opens a communication line to the client.

**Server Code:**

The server starts with a ServerSocket listening on a certain port to wait for a client to connect. On a successful client connection, the server accepts the connection and moves on to create input and output streams. The server then offers a user the right to send a message or to leave. Whenever the server sends a message, it's recorded in MessagingApp. Another thread, ServerRunnable, listens for incoming client messages and logs them in real-time. The server has implemented a menu showing messages, searching for messages, deleting messages, and shuffling messages.

**Client Code:**

The client establishes its connection with the server using a given IP address and port and establishes an input and output stream to communicate. In the client program, it asks the user to send or quit. All sent messages are logged in MessagingApp. The client starts running a new thread called ClientRunnable; this will receive message from the server and log them as well. Similarly, the same as the server, the client contains the functionalities to view, search, delete, and shuffle messages.

**Data Exchange:**

Once connected, a client and server use input and output streams to exchange messages. The client sends the messages to the server and server sends back the messages to the client.

**Termination of Connection:**

The connection can be terminated by either the client or the server by closing the socket, which degrades the communication link.

**Explanation of the output:**

**Explanation of the client output:**

.

**Explanation:**

At first, we compile and run the server class and then client class. Then client connected through sockets. The client selects the option 1 and send the message to the server. The server in return sends the message to client. And this communication continues. Then client selects the option 2 and exits.

**Server Menu Options:**

**Explanation:**

The client displays some options like display, search, delete, shuffle, display seen, unseen messages and exit. In the above output, the user selects option 1 and displays all the messages between client and server with message ID, content, status, and time stamp.

**Searching for a message:**

**Explanation:**

The client selects the option 2 and search for a message with ID 2. The output is the message with ID 2, with its details.

**Deleting a message:**

**Explanation:**

The user selects the option 3 that will delete a message. The user enters the message ID 4. The client deletes that message (“I am fine”).

**Displaying unseen messages:**

**Explanation:**

The user selects option 4 and the client will display only unseen messages. The client messages are displayed as unseen in its own class.

**Displaying seen messages:**

**Explanation:**

The user selects the option 5. The client will now display seen messages. The client messages are displayed as seen in server class and server messages are displayed as seen in client class.

**Shuffling Messages:**

**Explanation:**

The user now selects the option 6 and the client will display messages in a random order. This is shuffling of messages.

**Client Shutdown:**

**Explanation:**

Upon choosing option 7, the client shuts down, closing all the connections and exiting the application.

**Explanation of the Server Output**:

**Explanation:**

We’ll compile and run the server class first. Then it will start waiting for a client. Once the client is connected, it can then receive the messages from the client. And it can also send the messages to the client. The server sends the messages to the client by choosing option 1 and then the client sends the exit and selects option 2 to close the connection. The server acknowledges with “Server exiting” and proceeds to disconnect.

**Server messages Operations:**

**Explanation:**

The server displays all stored messages. The list include message with ID, content, status, and timestamp. Messages sent by the client are marked as seen and messages from server are initially unseen.

**Search a Message:**

**Explanation:**

The server selects the option 2 and search for a message with ID 2. The output is the message with ID 2, with its details.

**Deleting a Message:**

**Explanation:**

The user selects the option 3 that will delete a message. The user enters the message ID 4. The server deletes that message (“I am fine”).

**Shuffle Messages:**

**Explanation:**

The user now selects the option 4 and the server will display messages in a random order. This is shuffling of messages.

**Displaying Unseen Messages:**

**Explanation:**

The user selects option 5 and the server will display only unseen messages. The server messages are displayed as unseen in its own class.

**Displaying seen Messages:**

**Explanation:**

The user selects the option 6. The server will now display seen messages. The client messages are displayed as seen in server class and server messages are displayed as seen in client class.

**Server Exit and socket Exception:**

**Explanation:**

The server chooses option 7 to shut down, closing all connections and exiting the application. This exception is thrown because the server has closed the socket, which stops the server from reading data from the client’s socket. This exception is expected when shutting down network connections and is typically handled to end the program gracefully.