

|  |  |
| --- | --- |
| Reg. # | SP24-BSE-116 |
| Name | Subhan Khalid |
| Course Title | OOP In Java |
| Section | B |
| Assignment # | 2 |
| Submitted to | Dr. M Shahid Bhatti |
| Date | 14/11/2024 |

**Explanation of Java**

**Client And Server**

Table of Contents

1. Text Class
2. Client Class
3. Server Class
4. Main Class

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Text Class

The Text class represents a message in the chat system, containing essential details like the message content, sender, recipient, status (read/unread), and timestamp. This class is crucial for organizing the structure of messages exchanged between the client and server.

*Attributes:*

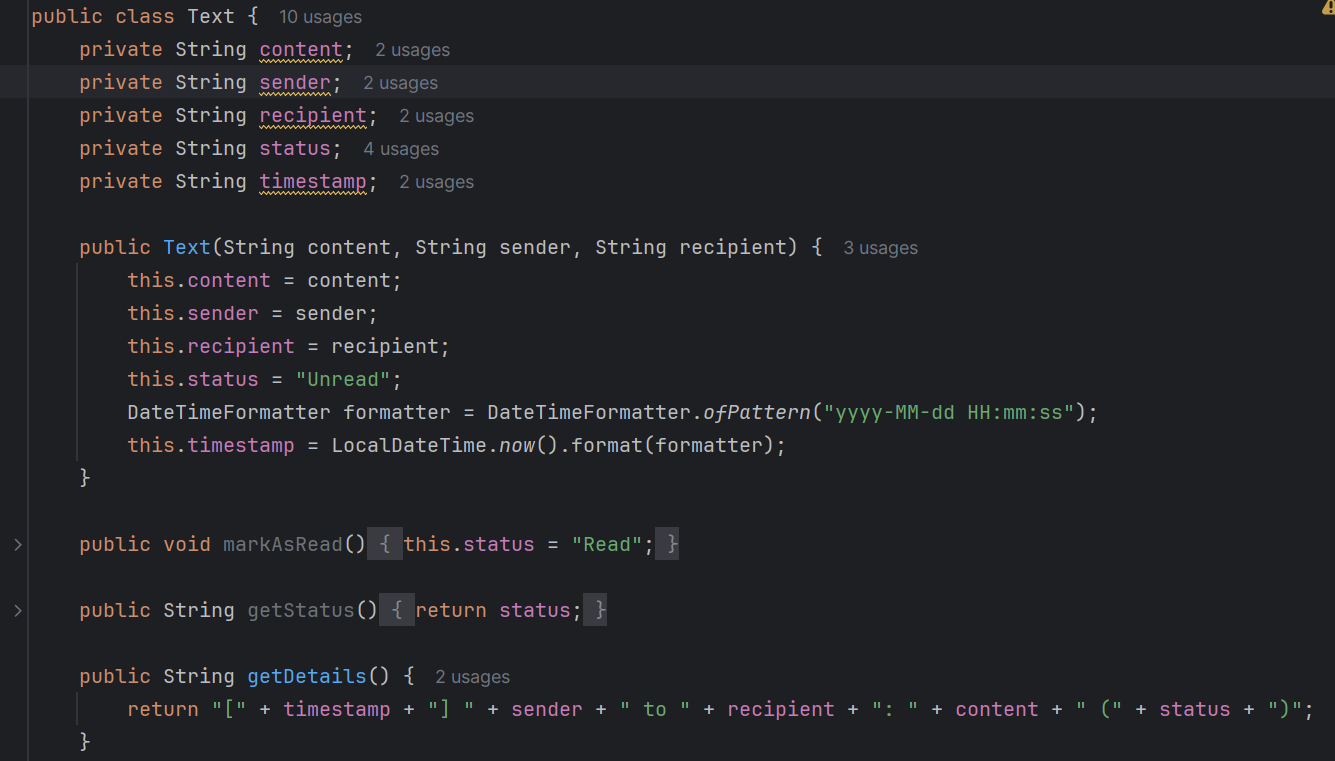
content: Stores the actual message content.

sender: Represents the person or system sending the message.

recipient: Represents the person or system receiving the message.

status: Indicates whether the message has been read or not. Initially set to "Unread."

timestamp: Records the exact time the message was sent, formatted as "yyyy-MM-dd HH:mm

" using LocalDateTime.

*Key Methods:*

Constructor (Text(String content, String sender, String recipient)): This method is used to create a new message, initializing the content, sender, recipient, and setting the status to "Unread." It also records the current time (timestamp) when the message is created using DateTimeFormatter and LocalDateTime.now().

markAsRead(): This method updates the message's status from "Unread" to "Read," indicating that the recipient has seen the message.

getStatus(): Returns the current status of the message (either "Read" or "Unread").

getDetails(): This method returns a formatted string containing the timestamp, sender, recipient, content, and status of the message. It helps in displaying message details in an organized format.

*Use Case:*

When a message is sent or received, a new Text object is created. For example, when a message is received from the server, the client creates a Text object to store that message's details and adds it to the message history. This class ensures that each message is timestamped and properly tracked with its read status.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Client Class

The Client class represents the client-side of the communication, where it connects to the server and allows the user to send and receive messages. It uses a socket connection to communicate with the server and maintains a history of the messages exchanged.

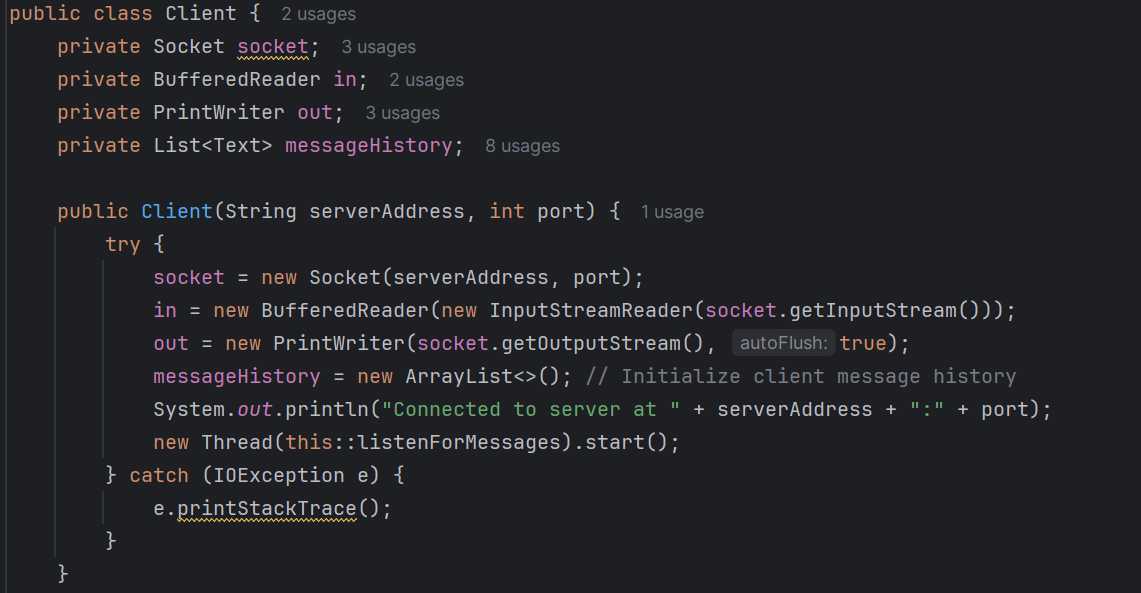
*Attributes:*

socket: A Socket object that connects the client to the server.

in: A BufferedReader for reading messages from the server.

out: A PrintWriter for sending messages to the server.

messageHistory: A list of Text objects that stores the history of messages exchanged.



*Key Features and Methods:*

Constructor (Client(String serverAddress, int port)): Establishes a connection to the server using the given server address and port number. It initializes the communication streams (in and out) and starts a new thread that listens for incoming messages from the server.

listenForMessages(): This method continuously listens for messages from the server. Whenever a message is received, it's added to the messageHistory list as a new Text object, and the content is displayed on the console.

sendMessageToServer(String message): This method sends a message to the server by writing it to the output stream (out). It also adds the message to the client’s message history.

displayMessageHistory(): Displays the entire list of messages in the client’s message history, showing the timestamp, sender, recipient, content, and status of each message.

deleteLastMessage(): Allows the client to delete the last message in the history. This feature can be useful for correcting mistakes or removing unwanted messages.

*Use Case:*

The client establishes a connection with the server and can send and receive messages. The client also keeps a record of all messages exchanged, making it easy to view the message history or delete the last message if necessary. The listenForMessages method is especially important as it ensures the client is always ready to receive messages from the server.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*3. Server Class*

The Server class represents the server-side of the communication, handling connections from clients. It listens for incoming messages, processes them, and sends responses or acknowledgments back to the client. It also maintains a history of messages exchanged with the client.

*Attributes:*

serverSocket: A ServerSocket object that listens for incoming client connections.

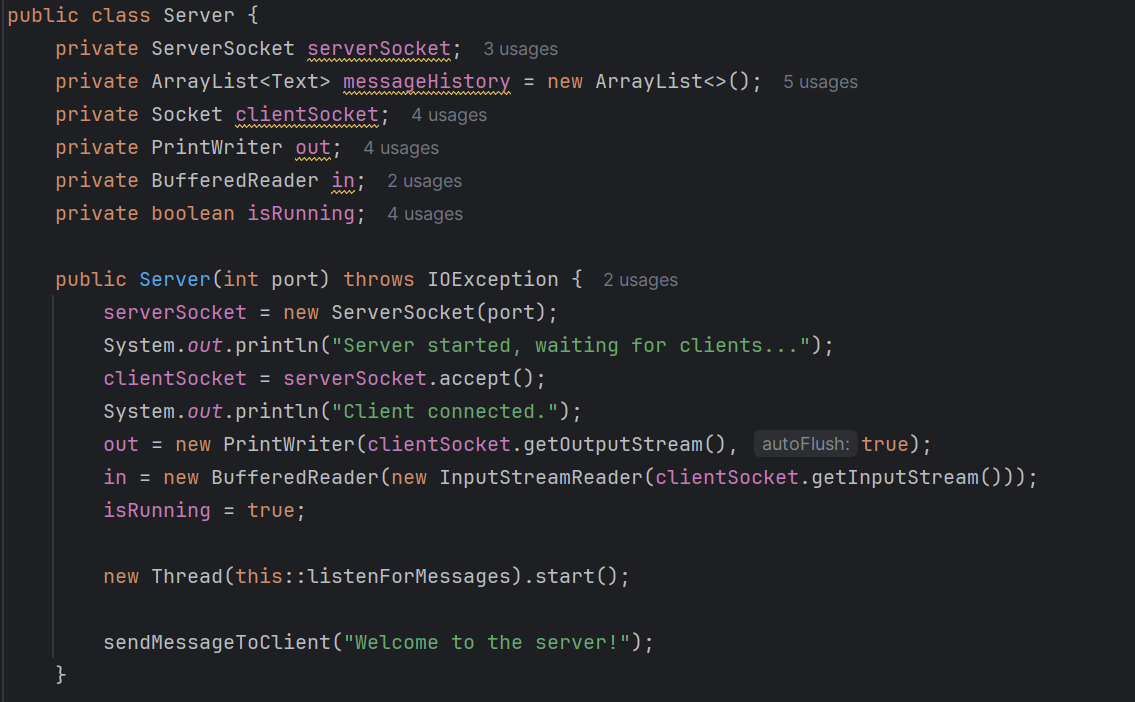
clientSocket: A Socket object representing the connection with the client.

out: A PrintWriter used to send messages to the client.

in: A BufferedReader used to read messages from the client.

messageHistory: A list of Text objects that stores the history of messages received and sent by the server.

isRunning: A boolean flag that controls the server’s operation and termination.



*Key Features and Methods:*

Constructor (Server(int port)): This method starts the server by binding it to a specific port. It listens for client connections, and when a client connects, it sets up input and output streams for communication. It also starts a new thread to listen for incoming messages from the client.

listenForMessages(): Continuously listens for incoming messages from the client. When a message is received, it’s added to the message history, and an acknowledgment message ("Parh Lia Ha.") is sent back to the client.

sendAcknowledgement(): Sends an acknowledgment message to the client, letting them know the server has received their message.

sendMessageToClient(String message): Sends a message to the connected client.

displayMessageHistory(): Displays all the messages exchanged between the client and server.

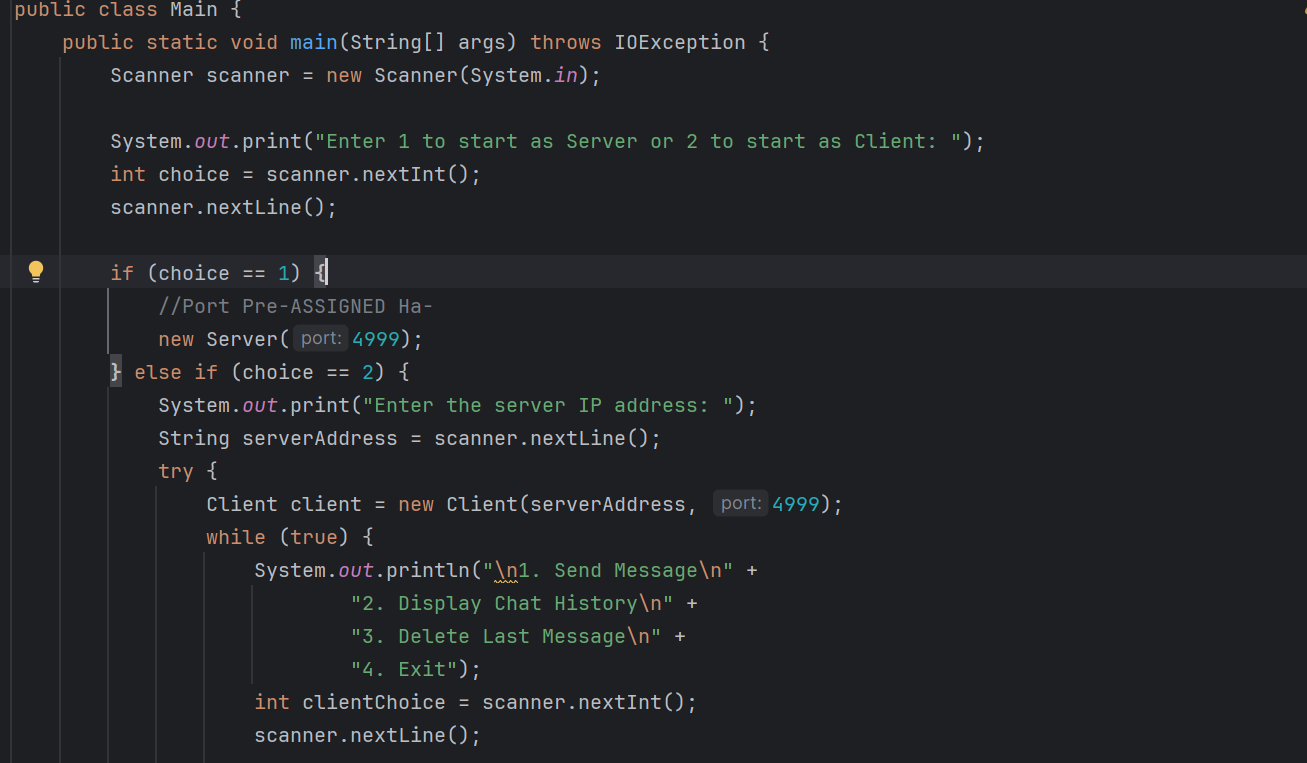
deleteLastMessage(): Removes the last message from the server’s message history.

*Use Case:*

The server listens for incoming client connections, receives and processes messages, and can send replies. It maintains a history of all messages exchanged with the client. The server can also delete the last message in case of mistakes or other reasons.

4. Main Class

The Main class is the entry point of the application. It allows the user to select whether to run the program as a server or a client. Based on the user's choice, it either initializes a Server object or a Client object and allows them to perform various actions.



*Key Features:*

User Input: The program first asks the user to input 1 to start as a server or 2 to start as a client.

Client-Side Operations: If the user chooses to run as a client, they are prompted to enter the server’s IP address and port number. The client can then send messages, display message history, or delete the last message.

Server-Side Operations: If the user chooses to run as a server, the program initializes the server, waits for a client connection, and allows the server to send messages, display message history, or delete the last message.

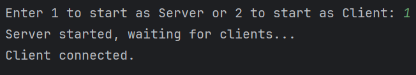
*Use Case:*

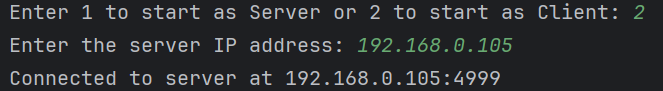
The user can interact with the system by selecting whether to operate as a client or server. Based on their choice, they can engage in real-time messaging, manage the message history, and perform various other operations.

**-Code Working-**

**Client Class Functions**

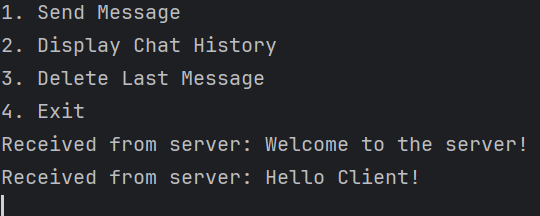
**Constructor: Client(String serverAddress, int port)**

* Connects to the server at the specified address and port, sets up communication streams, and starts listening for messages.



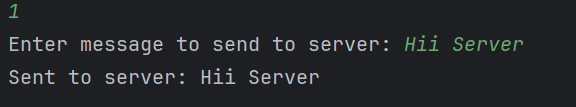
**Method: listenForMessages()**

* Continuously listens for messages from the server, stores them, and prints them to the console.



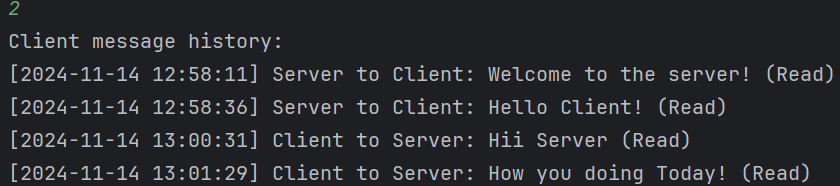
**Method: sendMessageToServer(String message)**

* Sends a message to the server and stores it in the message history.



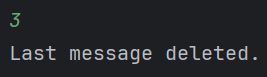
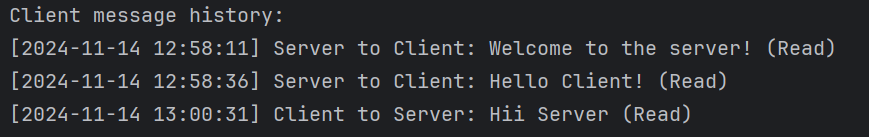
**Method: displayMessageHistory()**

* Displays the history of sent and received messages.



**Method: deleteLastMessage()**

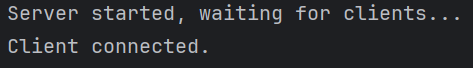
* Deletes the most recent message from the history.



**Server Class Functions**

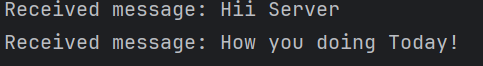
**Constructor: Server(int port)**

* Starts the server, listens for a client connection, and sets up communication streams.



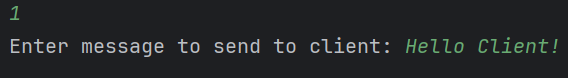
**Method: listenForMessages()**

* Listens for messages from the client, stores them

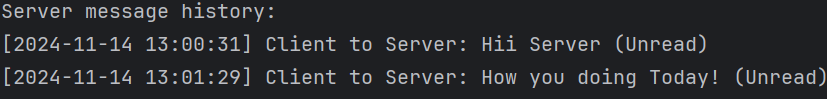


**Method: sendMessageToClient(String message)**

* Sends a message to the client.



**Method: displayMessageHistory()**

* Displays the server’s message history.

**Method: deleteLastMessage()**

* Deletes the most recent message from the server’s history.

