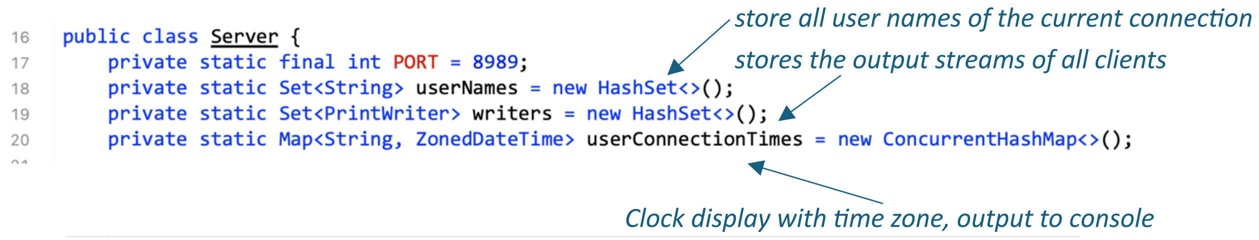
CSCI 4311

Program Assignment 1 Report

Shijun Jiang

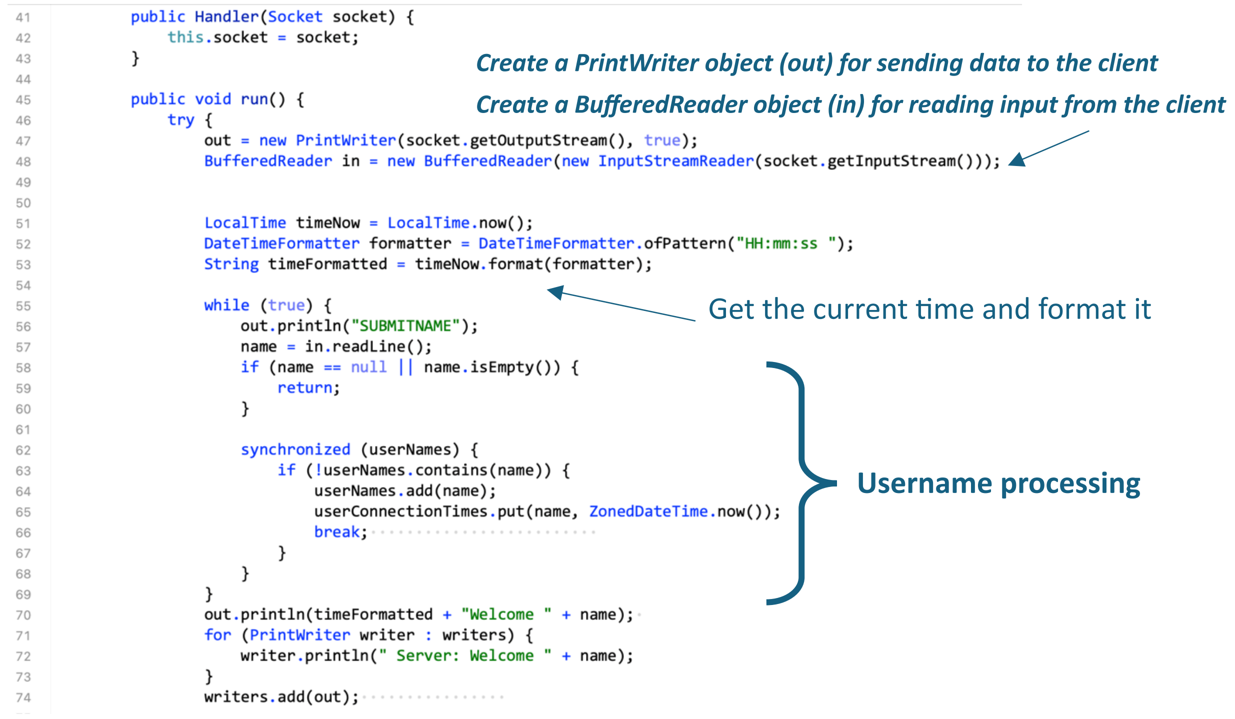
Code Overview

Server.java implements a multi-threaded server for handling multiple client connections and message delivery. The server listens to a specified port through ServerSocket and waits for the client's connection. The server maintains two key data structures: Set<String> userNames and Set<PrintWriter> writers. userNames is used to ; the writers collection stores the output streams of all clients, allowing the server to broadcast messages to all connected clients. The Handler class implements the Runnable interface and represents the processing logic for each client connection. It reads messages sent by clients and broadcasts them to all other clients.



```java
public class Server {
    private static final int PORT = 8989;
    private static Set<String> userNames = new HashSet<>();
    private static Set<PrintWriter> writers = new HashSet<>();
    private static Map<String, ZonedDateTime> userConnectionTimes = new ConcurrentHashMap<>();
```

store all user names of the current connection
stores the output streams of all clients
Clock display with time zone, output to console



```java
        public Handler(Socket socket) {
            this.socket = socket;
        }

        public void run() {
            try {
                out = new PrintWriter(socket.getOutputStream(), true);
                BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));


                LocalTime timeNow = LocalTime.now();
                DateTimeFormatter formatter = DateTimeFormatter.ofPattern("HH:mm:ss ");
                String timeFormatted = timeNow.format(formatter);

                while (true) {
                    out.println("SUBMITNAME");
                    name = in.readLine();
                    if (name == null || name.isEmpty()) {
                        return;
                    }

                    synchronized (userNames) {
                        if (!userNames.contains(name)) {
                            userNames.add(name);
                            userConnectionTimes.put(name, ZonedDateTime.now());
                            break;
                        }
                    }
                }
                out.println(timeFormatted + "Welcome " + name);
                for (PrintWriter writer : writers) {
                    writer.println(" Server: Welcome " + name);
                }
                writers.add(out);
```

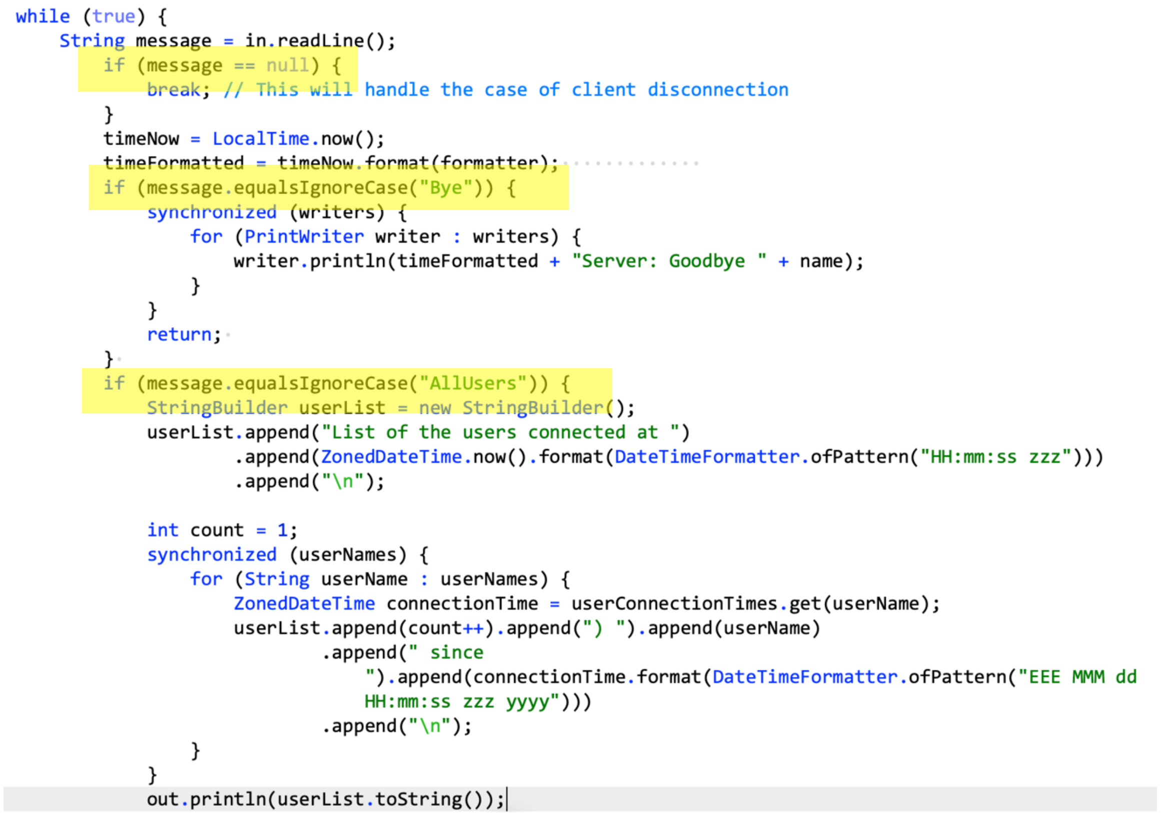
Create a PrintWriter object (out) for sending data to the client
Create a BufferedReader object (in) for reading input from the client
Get the current time and format it
Username processing

4 branch situations and their processing:

(1) If the read message is **null**, it means that the client has disconnected, and the loop will jump out through break to end the current client's processing.

(2) If the message sent by the client is **Bye(bye)**, the server will broadcast the user's away message to all clients and end the current thread.

(3) If the client requests a list of all currently connected users (by sending the **AllUsers** command), the server will build and send a message containing a list of all currently connected users and their connection times. This list is constructed by synchronously accessing the userNames collection and the userConnectionTimes map.

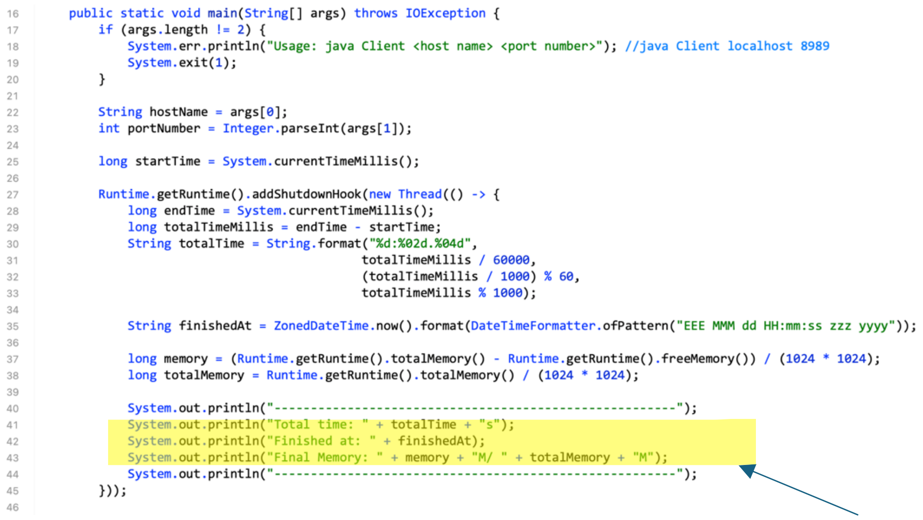


```
while (true) {
    String message = in.readLine();
        if (message == null) {
            break; // This will handle the case of client disconnection
        }
        timeNow = LocalTime.now();
        timeFormatted = timeNow.format(formatter);
        if (message.equalsIgnoreCase("Bye")) {
            synchronized (writers) {
                for (PrintWriter writer : writers) {
                    writer.println(timeFormatted + "Server: Goodbye " + name);
                }
            }
            return;
        }
        if (message.equalsIgnoreCase("AllUsers")) {
            StringBuilder userList = new StringBuilder();
            userList.append("List of the users connected at ")
                    .append(ZonedDateTime.now().format(DateTimeFormatter.ofPattern("HH:mm:ss zzz")))
                    .append("\n");

            int count = 1;
            synchronized (userNames) {
                for (String userName : userNames) {
                    ZonedDateTime connectionTime = userConnectionTimes.get(userName);
                    userList.append(count++).append(") ").append(userName)
                            .append(" since
                                ").append(connectionTime.format(DateTimeFormatter.ofPattern("EEE MMM dd
                                HH:mm:ss zzz yyyy")))
                            .append("\n");
                }
            }
            out.println(userList.toString());
```

(4)If the message received is **NOT** the special command above, the server broadcasts the message to all connected clients. This is accomplished by looping through the writers collection and sending a formatted message to each PrintWriter.

Client.java implements client logic for connecting to the server and sending and receiving messages. The client establishes a connection with the server through Socket and communicates with the server using input and output streams.

There are two main threads: one that reads the user's input and sends it to the server, and the other that continuously receives messages from the server and prints them to the console. In order to achieve real-time user interaction, the client does not need to wait for the server's response before sending a message to the server.

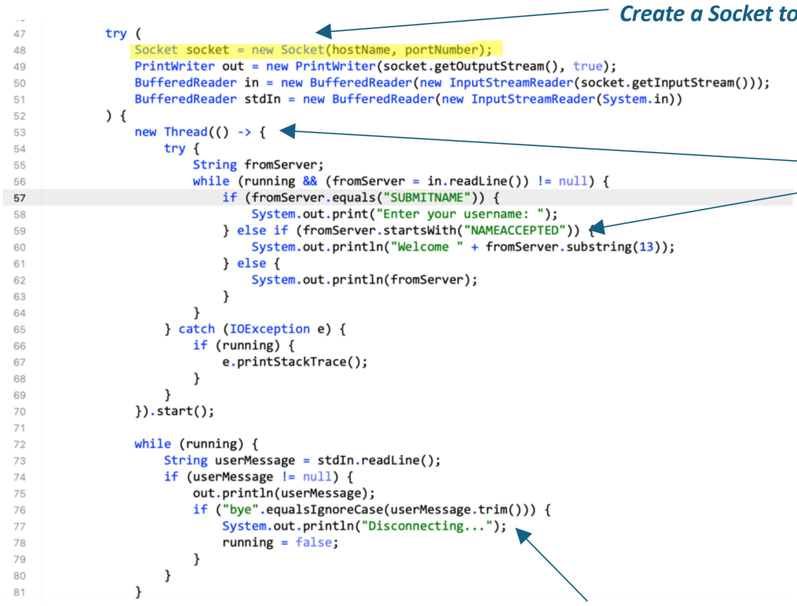


```java
    public static void main(String[] args) throws IOException {
        if (args.length != 2) {
            System.err.println("Usage: java Client <host name> <port number>"); //java Client localhost 8989
            System.exit(1);
        }

        String hostName = args[0];
        int portNumber = Integer.parseInt(args[1]);

        long startTime = System.currentTimeMillis();

        Runtime.getRuntime().addShutdownHook(new Thread(() -> {
            long endTime = System.currentTimeMillis();
            long totalTimeMillis = endTime - startTime;
            String totalTime = String.format("%d:%02d.%04d",
                                            totalTimeMillis / 60000,
                                            (totalTimeMillis / 1000) % 60,
                                            totalTimeMillis % 1000);

            String finishedAt = ZonedDateTime.now().format(DateTimeFormatter.ofPattern("EEE MMM dd HH:mm:ss zzz yyyy"));

            long memory = (Runtime.getRuntime().totalMemory() - Runtime.getRuntime().freeMemory()) / (1024 * 1024);
            long totalMemory = Runtime.getRuntime().totalMemory() / (1024 * 1024);

            System.out.println("-------------------------------------------------------");
            System.out.println("Total time: " + totalTime + "s");
            System.out.println("Finished at: " + finishedAt);
            System.out.println("Final Memory: " + memory + "M/ " + totalMemory + "M");
            System.out.println("-------------------------------------------------------");
        }));
```

Used to print the total running time, end time, final used memory and other information at the end of the program

Create a Socket to connect to the server

Start a new thread to read messages from the server asynchronously

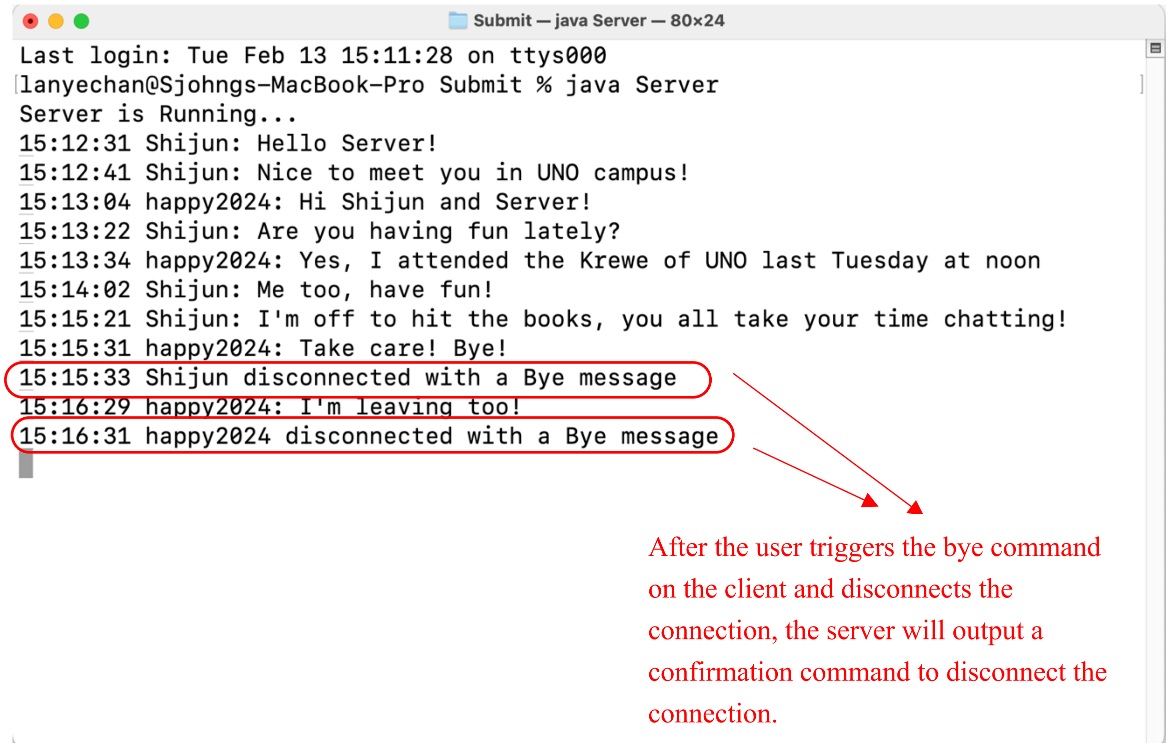


```java
        try (
            Socket socket = new Socket(hostName, portNumber);
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in))
        ) {
            new Thread(() -> {
                try {
                    String fromServer;
                    while (running && (fromServer = in.readLine()) != null) {
                        if (fromServer.equals("SUBMITNAME")) {
                            System.out.print("Enter your username: ");
                        } else if (fromServer.startsWith("NAMEACCEPTED")) {
                            System.out.println("Welcome " + fromServer.substring(13));
                        } else {
                            System.out.println(fromServer);
                        }
                    }
                } catch (IOException e) {
                    if (running) {
                        e.printStackTrace();
                    }
                }
            }).start();

            while (running) {
                String userMessage = stdIn.readLine();
                if (userMessage != null) {
                    out.println(userMessage);
                    if ("bye".equalsIgnoreCase(userMessage.trim())) {
                        System.out.println("Disconnecting...");
                        running = false;
                    }
                }
            }
```

If the user enters bye, print a disconnect message and set the running flag to false to end the main loop.

Output:

1） Server.java

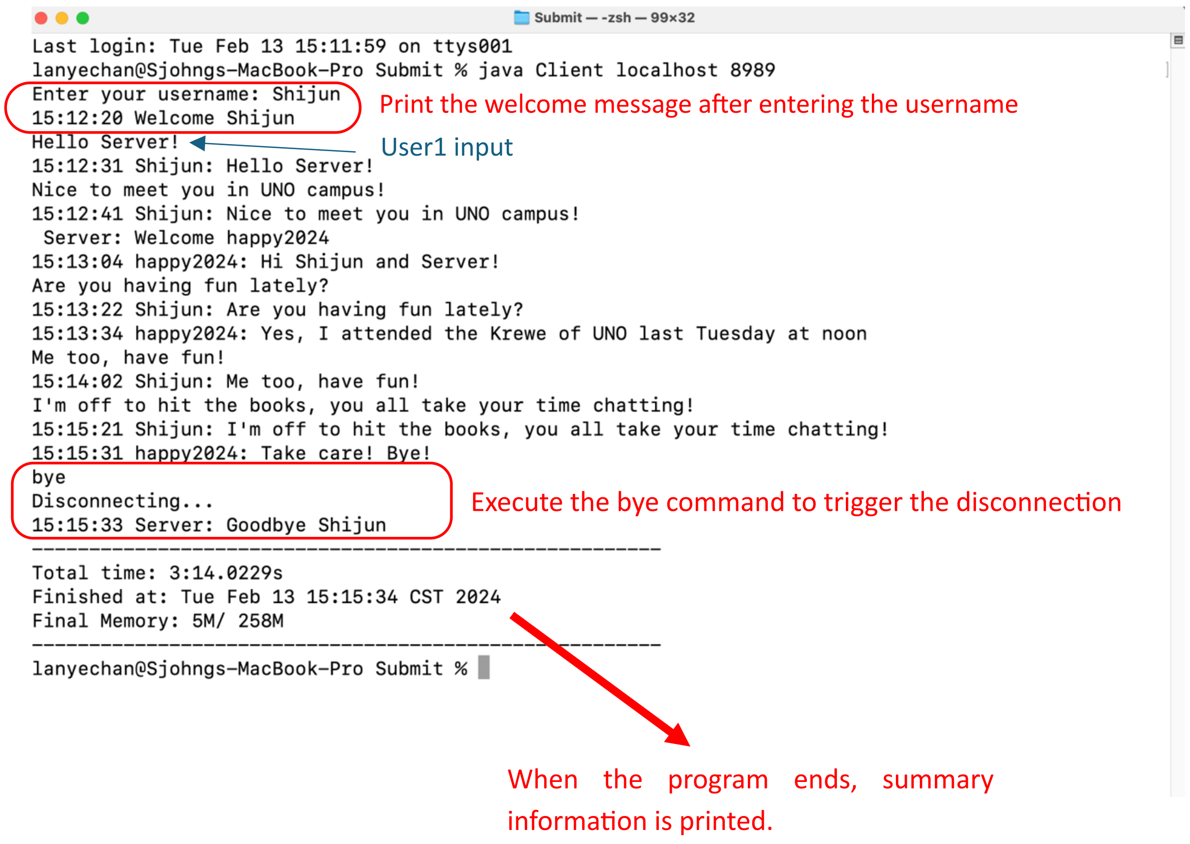

Submit — java Server — 80×24

```
Last login: Tue Feb 13 15:11:28 on ttys000
lanyechan@Sjohngs-MacBook-Pro Submit % java Server
Server is Running...
15:12:31 Shijun: Hello Server!
15:12:41 Shijun: Nice to meet you in UNO campus!
15:13:04 happy2024: Hi Shijun and Server!
15:13:22 Shijun: Are you having fun lately?
15:13:34 happy2024: Yes, I attended the Krewe of UNO last Tuesday at noon
15:14:02 Shijun: Me too, have fun!
15:15:21 Shijun: I'm off to hit the books, you all take your time chatting!
15:15:31 happy2024: Take care! Bye!
15:15:33 Shijun disconnected with a Bye message
15:16:29 happy2024: I'm leaving too!
15:16:31 happy2024 disconnected with a Bye message
```

After the user triggers the bye command on the client and disconnects the connection, the server will output a confirmation command to disconnect the connection.

2) Client.java

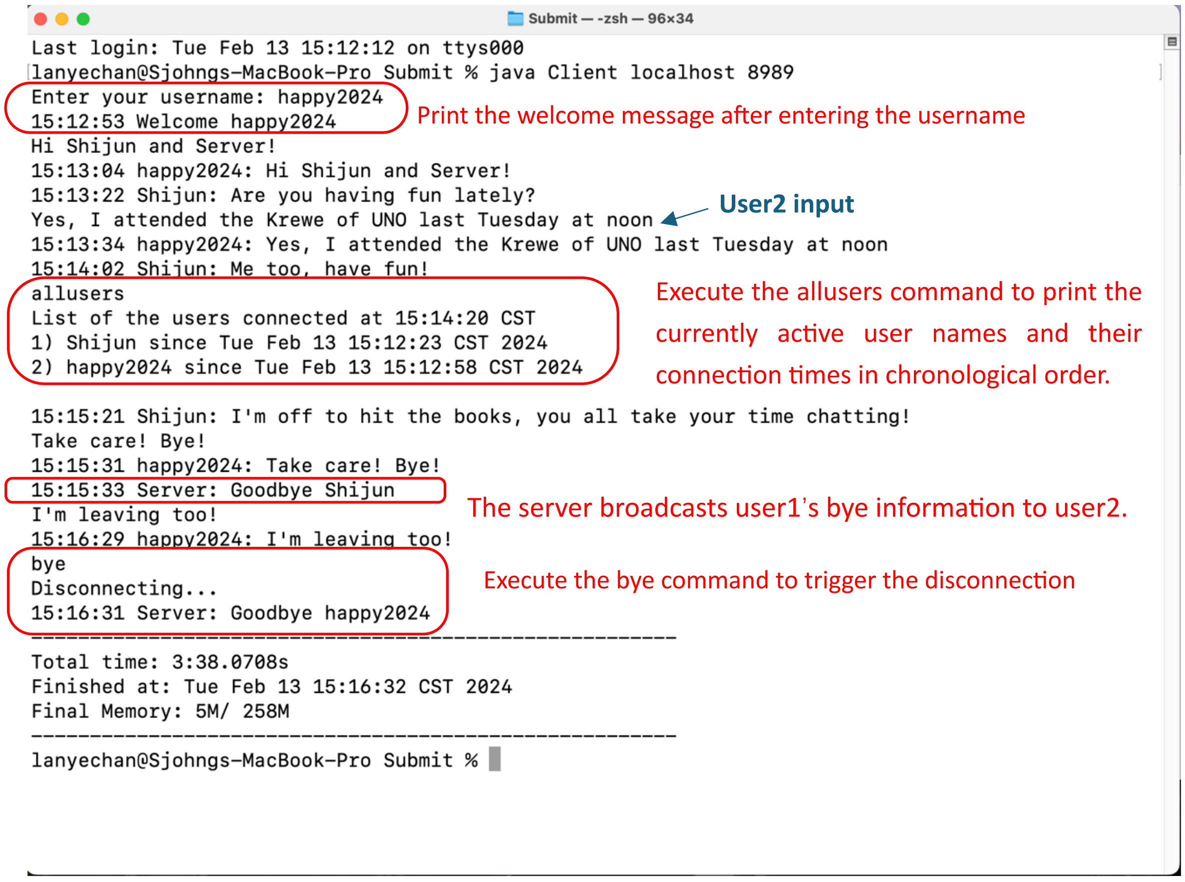
User1-Shijun



```
Last login: Tue Feb 13 15:11:59 on ttys001
lanyechan@Sjohngs-MacBook-Pro Submit % java Client localhost 8989
Enter your username: Shijun
15:12:20 Welcome Shijun
Hello Server!
15:12:31 Shijun: Hello Server!
Nice to meet you in UNO campus!
15:12:41 Shijun: Nice to meet you in UNO campus!
 Server: Welcome happy2024
15:13:04 happy2024: Hi Shijun and Server!
Are you having fun lately?
15:13:22 Shijun: Are you having fun lately?
15:13:34 happy2024: Yes, I attended the Krewe of UNO last Tuesday at noon
Me too, have fun!
15:14:02 Shijun: Me too, have fun!
I'm off to hit the books, you all take your time chatting!
15:15:21 Shijun: I'm off to hit the books, you all take your time chatting!
15:15:31 happy2024: Take care! Bye!
bye
Disconnecting...
15:15:33 Server: Goodbye Shijun
-------------------------------------------------------
Total time: 3:14.0229s
Finished at: Tue Feb 13 15:15:34 CST 2024
Final Memory: 5M/ 258M
-------------------------------------------------------
lanyechan@Sjohngs-MacBook-Pro Submit %
```

Print the welcome message after entering the username
User1 input
Execute the bye command to trigger the disconnection
When the program ends, summary information is printed.

User2-happy2024


Submit — -zsh — 96×34
Print the welcome message after entering the username
User2 input
Execute the allusers command to print the currently active user names and their connection times in chronological order.
The server broadcasts user1's bye information to user2.
Execute the bye command to trigger the disconnection

```
Last login: Tue Feb 13 15:12:12 on ttys000
lanyechan@Sjohngs-MacBook-Pro Submit % java Client localhost 8989
Enter your username: happy2024
15:12:53 Welcome happy2024
Hi Shijun and Server!
15:13:04 happy2024: Hi Shijun and Server!
15:13:22 Shijun: Are you having fun lately?
Yes, I attended the Krewe of UNO last Tuesday at noon
15:13:34 happy2024: Yes, I attended the Krewe of UNO last Tuesday at noon
15:14:02 Shijun: Me too, have fun!
allusers
List of the users connected at 15:14:20 CST
1) Shijun since Tue Feb 13 15:12:23 CST 2024
2) happy2024 since Tue Feb 13 15:12:58 CST 2024

15:15:21 Shijun: I'm off to hit the books, you all take your time chatting!
Take care! Bye!
15:15:31 happy2024: Take care! Bye!
15:15:33 Server: Goodbye Shijun
I'm leaving too!
15:16:29 happy2024: I'm leaving too!
bye
Disconnecting...
15:16:31 Server: Goodbye happy2024
-------------------------------------------------------
Total time: 3:38.0708s
Finished at: Tue Feb 13 15:16:32 CST 2024
Final Memory: 5M/ 258M
-------------------------------------------------------
lanyechan@Sjohngs-MacBook-Pro Submit % 
```