

Phi Phan

CSCI 4311

### Project Assignment 1: Socket Programming

**Terminal Used:** GitBash

**Code Editor Used:** Microsoft Visual Studio Code

**Goal of the assignment** In this assignment, we build a simple group chat application. The protocol between the client and the server is as follows.

- You can choose to use either TCP or UDP in your implementation. (TCP preferred)

(Server.java) Line 31 – 36: Server Socket

```
30 // this opens the server on the port
31 ServerSocket serverSocket = new ServerSocket(port);
32 System.out.println("Scanning for projects...");
33
34 //server listens for clients that connect to port
35 while (true) {
36     Socket client = serverSocket.accept();
37 }
```

(Client.java) Line 20- 22: Client Socket

```
20 public Client(String address, int port){
21     //start of client socket
22     try (Socket socket = new Socket(address, port)) {
23 }
```

(Server.java) Line 35-44: Allows new clients to connect.

```
34 //server listens for clients that connect to port
35 while (true) {
36     Socket client = serverSocket.accept();
37
38     //create new clients object
39     ClientHandler clientSocket = new ClientHandler(client, clientsList);
40
41     //Thread begins to handle client
42     clientsList.add(clientSocket);
43     pool.execute(clientSocket);
44 }
```

- The server is first started on a known port.

(Server.java) Line 120-122 Server is created and starts on port 8989

```
119
120  ✓ public static void main(String[] args) {
121      |     Server server = new Server(8989);
122      | }
123 }
```

- The client program is started (server IP and port are provided on the command line).

(Client.java) Line 25-26 If Client program successfully starts, display serverIp and port

```
21      //start of client socket
22  ✓ try (Socket socket = new Socket(address, port)) {
23
24      //if connection is successful, displays serverIp and Port
25      System.out.println("Server IP: " + socket.getInetAddress()); //grabs and print IP
26      System.out.println("Server port: " + socket.getPort()); //grabs and print port
27 }
```

(Client.java) Line 53-55 End case: If there is no server or connection fails, prompt the client that there is no server connection

```
51
52      //end case if there is no server
53  } catch (IOException e) {
54      System.out.println("No server connection.");
55  }
56 }
57 }
```

- The client connects to the server. The server asks the user for input. The user types the username in the following format "username = ComNet" message on the terminal.
- After the user enters a username, the server broadcast to everyone "Server: Welcome username".

**This picture is for the last two bullet points**

(Server.java) Line 71 – 81: Prompts user to enter username and if client provides username, welcomes client

```

67     public void run(){
68         try {
69             //prompts Client to enter username store in username
70             out.println("Enter the username using the following format 'username = Bob'");
71             String splituser = in.readLine();
72             //grabs username
73             String[] username = splituser.split(" = ", 0);
74             if (username[0].equals("username")){
75                 userList.add(username[1]);
76                 //welcomes clients
77                 broadcastAll("Welcome " + username[1]);
78                 System.out.println(username[1] + " has connected.");
79             }
80         }
81     }
82 }

```

- If the user doesn't provide a username, the server doesn't accept the user's messages.

(Server.java) Line 104: Prompts user to provide a username if client doesn't provide a username and closes.

```

102
103     //if client doesn't provide username, prompts user
104     out.println("Username was not entered. Please enter a username.");
105     clientSocket.close();

```

- After that, the user can send messages (e.g., "Hi", "Bye", "How are you"). The user's input is sent to the server via the connected socket.

(Client.java) Line 37- 49: Allows Client to type in message and prompt it to server to display. If client inputs Bye, disconnect Client. In client inputs AllUsers, display current user

```

36     //messages are displayed on servers
37     String line = "";
38
39     //prompts user commands so they can use Bye = exit, AllUsers = see current users
40     System.out.println("Some Commands: type \Bye\ to exit the program or \AllUsers\ to display all users.");
41
42     //if client says Bye, disconnect Client and display Goodbye (user) in the server.
43     while (!line.equals("Bye")){
44         try {
45             line = outputs.readLine();
46             out.println(line);
47         } catch (IOException e){
48         }
49     }

```

- The server reads the user's input from the client's socket. If the user has typed "Bye", the server must broadcast to everyone with "Goodbye username" e.g. "Server: Goodbye ComNet".

- If a user enters "AllUsers", the server needs to send all active users to that user

(Server.java) Line 84-102: Check if Client type Bye or AllUsers

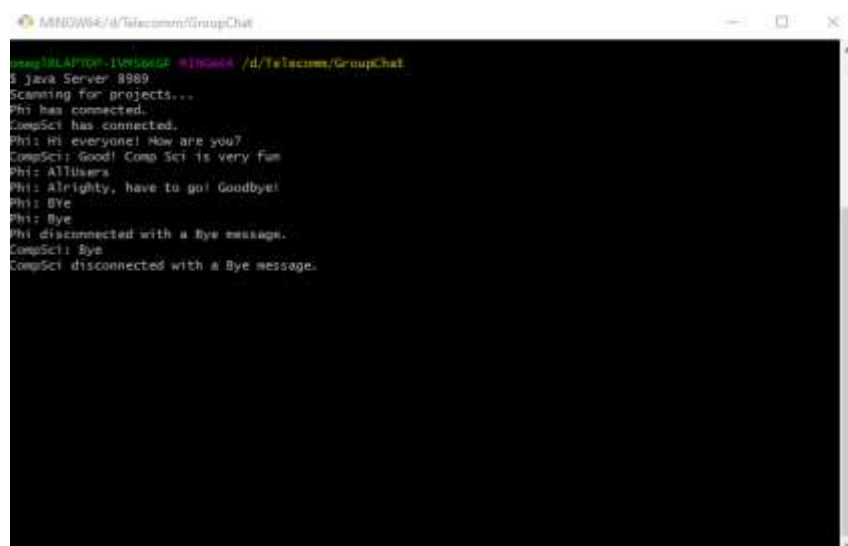
If client inputs "Bye" disconnects and prompt "Server: Goodbye 'username'" to everyone

If client inputs "AllUser", prompt active user to user

```
82
83 //if the user enters Bye, Client exits
84 String line = "";
85 while (!line.equals("Bye")) {
86     try {
87         line = in.readLine();
88
89         //if user enters AllUsers, displays all the users in the server
90         if (line.equals("AllUsers")) {
91             out.println(usersList);
92         }
93
94         broadcastAll(username[i] + ": " + line);
95         System.out.println(username[i] + ": " + line);
96     } catch (Exception e) {
97     }
98 }
99 //once Client exits, prompts Server and Clients
100 broadcastAll("Server: Goodbye " + username[i]);
101 usersList.remove(username[i]);
102
```

## Outputs:

### Server Output



```
MINGW64: d:/Telecomm/GroupChat
$ java Server 8989
Scanning for projects...
Phi has connected.
CompSci has connected.
Phi: Hi everyone! How are you?
CompSci: Good! Comp Sci is very fun
Phi: AllUsers
Phi: Alrighty, have to go! Goodbye!
Phi: Bye
Phi: Bye
Phi disconnected with a Bye message.
CompSci: Bye
CompSci disconnected with a Bye message.
```

## Client 1 Output:

```
MINGW64/d/Telecomm/GroupChat
$ java Client localhost 8888
Server IP: localhost/127.0.0.1
Server port: 8888
Type "Bye" to exit the program or "AllUsers" to see current users.
Enter the username using the following format 'username = Bob'

username = Phi
welcome Phi

welcome CompSci

Hi everyone! How are you?
Phi: Hi everyone! How are you?

CompSci: Good! Comp Sci is very fun

AllUsers
[Phi, CompSci]

Phi: AllUsers

Alrighty, have to go! Goodbye!
Phi: Alrighty, have to go! Goodbye!

Bye
Phi: Bye

Bye
Phi: Bye

root@LAPTOP-1VW56KJF: ~/d/Telecomm/GroupChat
$
```

## Client 2 Output:

```
MINGW64/d/Telecomm/GroupChat
root@LAPTOP-1VW56KJF: ~/d/Telecomm/GroupChat
$ java Client localhost 8888
Server IP: localhost/127.0.0.1
Server port: 8888
Type "Bye" to exit the program or "AllUsers" to see current users.
Enter the username using the following format 'username = Bob'

welcome Phi

username = CompSci
welcome CompSci

Phi: Hi everyone! How are you?

Good! Comp Sci is very fun
CompSci: Good! Comp Sci is very fun

Phi: AllUsers

Phi: Alrighty, have to go! Goodbye!

Phi: Bye

Phi: Bye

Server: Goodbye Phi

Bye

root@LAPTOP-1VW56KJF: ~/d/Telecomm/GroupChat
$
```