

Computer Science 354

Assignment 1 – Multi-Client/Server Chat
Program

By Group 16

Project collaborators:

Wihan Uys (21553491) - 21553491@sun.ac.za

Nikita Smal (20720661) – 20720661@sun.ac.za

Introduction:

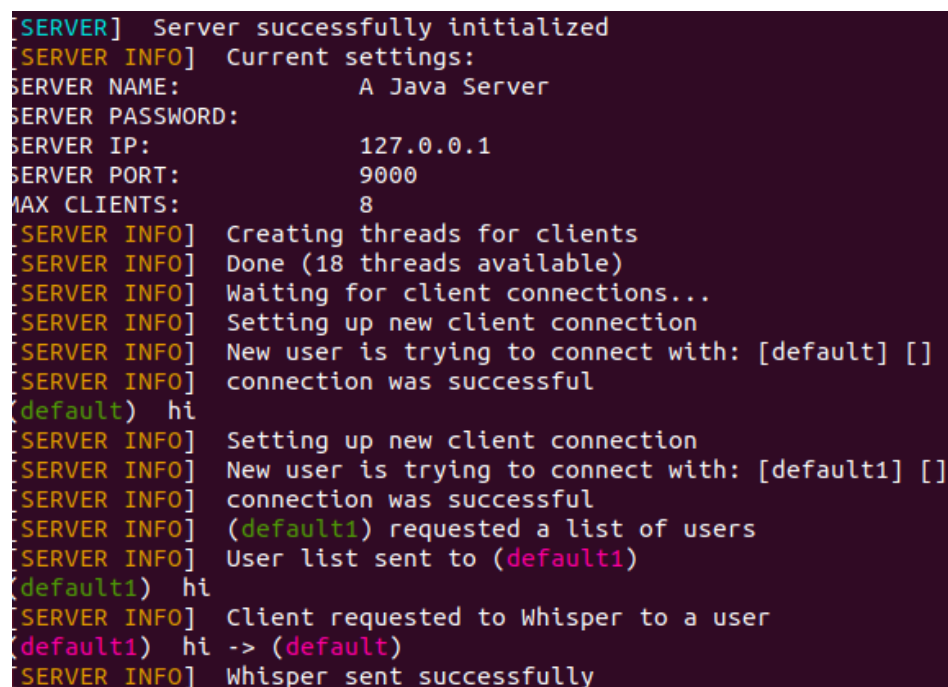
For this project the group had to develop a single server - multi client chat application. It was decided to use Java 8 as the programming language, as it was the language that had the most resources available to the group. This report will explain the features of both the server and the client as well as features that were not able to be implemented. There will also be a description of the files used in the program, the experiments and issues encountered while creating the program.

Features included in the solution:

The chat application includes multiple clients that are able to connect to a single server.

Features of the server:

- The server can be accessed by multiple clients.
- It is multi-threaded to handle requests from multiple clients.
- Server operations such as connect requests and disconnect requests are printed out by the server.
- Connections and disconnections can be handled by the server without affecting other services.
- Clients must have unique nicknames, and if the name the client is attempting to use has already been taken, the program (client GUI) will be shut down.
- The server notifies all clients when the list of connected clients changes.
- If the client does not specify port number and sever IP address when starting the server there is a default mode for the server.
 - Default mode includes:
 - Server Name: A Java Server
 - Server IP: 127.0.0.1
 - Server Password:
 - Server Port: 9000
 - Max Clients: 8
- Information about the server is colour coded in the terminal, as seen in Figure 1.



```
[SERVER] Server successfully initialized
[SERVER INFO] Current settings:
SERVER NAME:      A Java Server
SERVER PASSWORD:
SERVER IP:        127.0.0.1
SERVER PORT:      9000
MAX CLIENTS:      8
[SERVER INFO] Creating threads for clients
[SERVER INFO] Done (18 threads available)
[SERVER INFO] Waiting for client connections...
[SERVER INFO] Setting up new client connection
[SERVER INFO] New user is trying to connect with: [default] []
[SERVER INFO] connection was successful
(default) hi
[SERVER INFO] Setting up new client connection
[SERVER INFO] New user is trying to connect with: [default1] []
[SERVER INFO] connection was successful
[SERVER INFO] (default1) requested a list of users
[SERVER INFO] User list sent to (default1)
(default1) hi
[SERVER INFO] Client requested to Whisper to a user
(default1) hi -> (default)
[SERVER INFO] Whisper sent successfully
```

Figure 1: Colour code for server in terminal.

Features of the client:

- The client offers a simple graphical user interface (GUI).
- Clients have the option of selecting a nickname.
- A list of online users is displayed via a command in the client GUI.
- Users' connection and disconnection actions are displayed.
- Messages from the individual who sent the message as well as messages from other clients are displayed (in other words the messages the client sends is also displayed).
- Whisper (private) messages between clients has been implemented.
- While typing a message, clients can receive messages or actions.
- Clients are able to disconnect without disrupting the server.

As shown in Figure 2 the features of the Client GUI:

- Contains a field for the client to enter
 - A unique username,
 - The server IP address,
 - The server password,
 - The server port number.
- Has a help text box, that informs the client of what they are able to do in the program.
- The client has the ability to type and send a message once connected to the server.
- There is also a text box that displays the messages from the server and the other clients using the program.

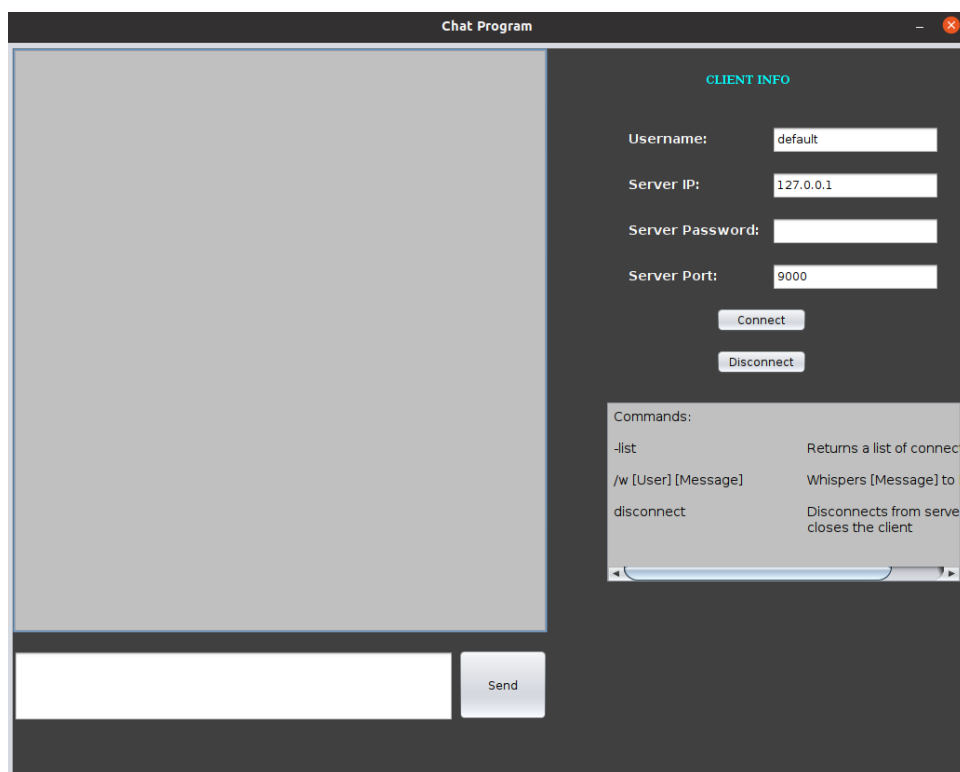


Figure 2: Client GUI.

Features not included in the solution:

All features listed in the specification document of the assignment where included.

Extra features included:

There were not any extra features implemented in the assignment.

Description of files:

The source file includes the following files:

- Client.java
 - Socket that connects to the server and provides an interface to the GUI that allows the user to communicate to the server.
- ClientGUI.java
 - Contains the code for setting up the GUI such as frame and Jpanel layout as well as the colour and text settings for each of these components.
 - Receives the information necessary to be displayed in the GUI from the client and server files.
- ClientReceiver.java
 - All Message objects sent to a particular client are received and processed for the client.
 - Handles error checking for the client.
- ClientSender.java
 - All Message objects sent to a particular client are received and processed for the client.
 - Handles error checking for the client.
- Colours.java
 - Contains special ANSI encoding for all the colours that are used for the usernames in the terminal.
- Message.java
 - This file contains the constructor to create a Message object, as well as get and set functions for the Message.
- Server.java
 - The server stores and controls the client's socket as well as their Message objects.
 - Also forms the enter point for the program through the main.
- ServerReceiver.java
 - Receives all Message objects from the ClientSender.
 - Handles error checking.
- ServerSender.java
 - Sends all Message objects to the ClientReceiver.
 - Handles error checking.

Description of the program:

The program starts in the main function of the Server.java file, where a server is created and started. This will run for the entire duration of the program execution. A client is then connected to the server through the Client.java file, either in the terminal or the client GUI.

Once the client is connected to the server, they will be able to send a message to the server and the server will then be able to send its reply back to the client. Both the client and server are continuously listening for a message from either one. The server receives messages from the client through the file ServerReceiver.java and sends messages to the client through the file ServerSend.java. The client receives messages from the server through the file ClientReciever.java and sends messages to the server through the file ClientSender.java.

If multiple clients are connected to the server, each client will send the message to the server and the server will then send the message to all other clients if the message was public or to a particular client if the message was a whisper. Each client is run on its own thread so as not to cause blocking while waiting for a message from the server.

Experiments:

To ensure the code worked as expected experiments were run in the terminal and the GUI. These experiments included starting the server and connecting clients to the server, if this was successful the next stage could begin.

The following stage included checking that the error messages (such as duplicate names) showed up at the correct time and that all the necessary information was displayed when required.

Multiple tests were run to see how the program would run as expected, these tests included:

- Connecting multiple clients,
- Connecting a client to an online server,
- Sending whispers between clients,
- Sending public messages on the chat program,
- Connecting with the wrong username,
- Connecting with the wrong server password,
- Disconnecting clients,
- Making sure the list functionality worked,
- Connecting to Hamachi and running the program on two different machines.

Issues encountered:

One of the issues encountered was methods blocking, where the client was not able to receive messages from the server if the client hadn't sent a message yet a fix to this was to use `ObjectInput` and `ObjectOutput` streams.

Another issue that was encountered was that the GUI did not use the same colour implementation as the terminal, thus when we setup the code from the terminal to be outputted in the GUI it resulted in incorrect output showing up. To correct this, we changed the way that the output was sent to the GUI.

Compilation:

To compile the code run "make" in terminal which will automatically compile all the code that is necessary to run the program. The commands to run the program can then be executed in terminal as described below in the execution of the program.

Execution:

How to Compile and run:

1. ``git clone git@git.cs.sun.ac.za:Computer-Science/rw354/2021/project-1/group-16-project-1.git``
2. ``cd group-16-project-1``
3. ``make all``
4. ``cd out``

To run the Server:

```
`java Server` (This runs the server in default mode)
```

(Optional arguments)

```
`java Server -c [Server Name] [Server IP] [Server Password] [Server Port] [Max Clients]`
```

To run the Client:

```
`java ClientGUI` (This runs the Client GUI)
```

(Optional command line client)

```
`java Client - c [Username] [Server IP] [Server Password] [Server Port]`
```

Notes:

There is currently a server running on a VPS which the clients can connect to

```
`SERVER IP: 63.250.42.32`
```

```
`SERVER PASSWORD: 1234`
```

```
`SERVER PORT: 4999`
```

Libraries:

External libraries that were used in the implementation:

- Java Socket
 - Java socket that implements communication between two machines.
- Java ServerSocket
 - A server socket sits and waits for network requests to arrive.
 - It then returns a result to the requester after performing some operation based on that request.
- Java ObjectInputStream and Java ObjectOutputStream
 - When used with a FileOutputStream and FileInputStream, ObjectOutputStream and ObjectInputStream can provide persistent storage for graphs of objects to an application.
 - ObjectInputStream is used to recover previously serialized items.
 - Other applications include sending objects between hosts via a socket stream, as well as marshaling and unmarshaling arguments and parameters in a remote communication system.
- Java InetAddress
 - This class represents an Internet Protocol (IP) address.
- Java ExecutorService
 - An Executor with methods for managing termination and producing a Future for tracking the progress of one or more asynchronous processes.
 - Shutting off an ExecutorService will lead it to reject new jobs.

- Javax Swing
 - A library of components used to make the GUI.
- Java Serializable
 - Subtypes of non-serializable classes may acquire responsibility for saving and restoring the state of the supertype's public, protected, and (if accessible) package fields in order to allow them to be serialized.

Conclusion:

In this report, the file description of the program, the flow of the program and the libraires used in the program where giving. It was shown that all features of the assignment where implemented. Each feature of the Server, Client and clientGUI were described as well as how they communicate with each other. Finally the experiments that were run on the program were described and how the issues that were encountered while testing where overcome.