Report of the Server

Server side codes are mostly in the Server.java file, which contains a Server class.

1. The work principle of server

As a bonus version, the Server class enable group chatting and bidirectional message sending. Besides, each message can be multiple lines.

* How to achieve bidirectional message sending (sender don’t need to wait)?

In order to achieve group chatting, the server cannot directly join the chatting, and the way to enable server to join the chatting will be introduced in the 4th bullet point.

* How does server wait for client and receive the message from client?

There is a serve thread (not the main thread) created by server to keep on waiting for the incoming connect socket from new client. Each time a new client connects to the server, the server thread will create a new thread to keep on listening the message sent by the client.

* How does server achieve group chatting? (most important part)

The role of server is a broadcaster. Each time, an arbitrary client sends a message to server, the server broadcast the message to all clients (stored in a hash map). Then the client just displays the message received from server.

* How does server side user join the chatting?

This is done by main thread. If a user choose server, then a server will be set up. At the same time, a local client side application will run to achieve the server chatting. (detail can be seen in the main thread part)

1. The implementation detail of server

Because the implementation is a little complex, I will just explain some important parts.

* Socket initialization and bind the IP and port

This is done by constructor of Server. Which accept 2 variables(IP and port) and throws BindException.

try {

                this.serverSocket = new ServerSocket(port,MAX\_USER,InetAddress.getByName(ip));

           } catch (BindException e) {

                throw e;

           }

* listen and accept connections (bonus part)

In constructor, the thread wait for connections from clients will be called

Thread t = new Thread(() -> addClient());

this.waitForConnection = t;

t.start();

In the addClient() method, there will be a statement wait for connection and block the thread.

Socket incoming = serverSocket.accept();

* Store the client (bonus part)

The client sockets will be stored in a concurrent hash map (thread safe data structure). Key is the socket, value is the IP address of the socket

* Receive the text (bonus part)

There will be a new thread created in the addCient() method once a new client connects to the server to keep on listening the message from client.

* Send the text (bonus part)

Each time server received a message, it will send the message to all of the clients. (Use the iterator of hash map)

* Quit the program

If client side says “Bye”, server side will remove it from the hash map and close the socket.

If server side says “Quit”, main thread will use the public method stop() to finalize the server thread.

In the stop() method, the server thread waiting for connection will be interrupted, all sockets in the hash map will be deleted