TECHNICAL REPORT

SUBJECT

TCP sockets

STUDENT

Topcii Daniil

TEACHER

Crafa Nicola

Conte Alberto

TCP SOCKETS

For the creation of a TCP socket it was proposed as an exercise the creation of a chat that had a server that manages the connections and counts the number of iterations per user, when the client disconnects it prints the number of iterations on the screen, the client side instead it must allow authentication via nickname, two files have been created to carry out the exercise:

SERVERS and CLIENTS.

SERVER:

- The run() method executes when the server thread is started. Create a ServerSocket object on port 8000 and iteratively accept client connections. For each connection, a new ClientThread thread is started that handles client interactions.

CODE:

```
public voidrun() {
 try(ServerSocketserverSocket =newServerSocket(8000)) {
 // Accept client connections iteratively
// Create an object of class ChatClient
 while(true) {
 SocketsclientSocket = serverSocket.accept();
 // Start a new thread to handle client interactions
 newClientThread(clientSocket).start();
}
}catch(IOExceptionAnd) {
e.printStackTrace();
}
}
```

- The ClientThread class represents a thread that manages a client's interactions with the server. The run() method executes when the client thread is started. Receives the client's nickname and adds it to the list of used nicknames. If the nickname has already been used, send an error message to the client and kill the thread. Otherwise, it adds the client's nickname to the map that maps each connected client to its nickname, and sends all connected clients the client connect message. Receives and sends messages iteratively as long as the client is connected. When the client is disconnected, it removes the nickname from the list of used nicknames and from the nickname map, sends the disconnection message to all connected clients and prints the client iterations on the screen.

- The broadcast(String message) method sends the message to all connected clients. To do this, it scans the nickname map and sends the message using the output stream of each client's socket.

CODE:

```
private voidbroadcast(Stringmessage) {
 for(Map.Entry<Sockets,String> entries:clientNicknames.entrySet()) {
 SocketsclientSocket = entry.getKey();
 try{
 PrintStreamout =newPrintStream(clientSocket.getOutputStream());
out.println(message);
}catch(IOExceptionAnd) {
e.printStackTrace();
}
}
}
```

- The main(String[] args) method creates a Server object and starts its thread.

CLIENTS:

- The Client class extends JFrame and defines some instance variables for the window, the message panel, the text field for sending messages and the button for sending messages.
- In the constructor of the Client class, the chat window is created and panels for messaging and sending messages are added.
- It also shows a dialog where the user can enter his nickname.
- A new thread is then started which takes care of connecting the client to the server and managing the receipt and sending of messages.
- When the "Submit" button is pressed, the code inside the ActionListener sends the text of the text field to the server.