

```
package myChat;

/**
 *
 * @author Patrick Neubauer
 * @version 1.0 alpha
 *
 * Course: Distributed Systems
 * Project P1: Internet Chat System
 *
 * Team members:
 * Tenzin Dakpa, Patrick Neubauer
 *
 */

import java.net.*;
import java.io.*;
import java.util.Scanner; // for provinding console input

public class TCPClient implements Runnable {

    // defining private properties
    private Socket socket = null;

    private Thread thread = null;
    private TCPClientThread client = null;

    private DataInputStream streamIn = null; // streamIn is the console scanner that reads the input
    private DataOutputStream streamOut = null;
    private String nickname = "";

    /* constructor that creates a TCPClient
     */
    public TCPClient(String serverName, int serverPort, String _nickname) {
        System.out.println("Trying to establish connection to server " + serverName + " on port " + serverPort + ".");
        System.out.println("Please wait ...");
        try {
            socket = new Socket(serverName, serverPort);
            nickname = _nickname;
            System.out.println("Perfect, connection to server " + serverName + " properly established!\n");
            start();
        } catch (UnknownHostException uhe) {
            System.out.println("Host " + serverName + "is unknown, maybe wrong IP entered? Error: \n" + uhe.getMessage());
        } catch (IOException ioe) {
            System.out.println("Input/Output exception: \n" + ioe.getMessage());
        }
    }

    /* creating new data streams and a new client thread
     */
    public void start() throws IOException {
        streamIn = new DataInputStream(System.in);
        streamOut = new DataOutputStream(socket.getOutputStream());

        if (thread == null) {
            client = new TCPClientThread(this, socket);
            thread = new Thread(this);

            // writing to output stream that the user joined the chat
            streamOut.writeUTF(nickname + " joined the chat!");
            streamOut.flush(); // flush output stream
        }
    }
}
```

```

        thread.start();
    }
}

/* run client thread
 */
public void run() {
    while (thread != null) {
        try {
            // read from console (scanning input) and write it to DataOutputStream
            System.out.print("send: ");
            // writing the nickname and the message written by client to
            // the output stream
            streamOut.writeUTF(nickname + ": " + streamIn.readLine());
            streamOut.flush();
        } catch (IOException ioe) {
            System.out.println("Error while sending message: " + ioe.getMessage());
            stop();
        }
    }
}

/* Handles client message.
 * Writes it his own screen and handles exit command "/exit".
 */
public void handle(String message) {
    if (message.contains("/exit")) {
        System.out.println("You decided to leave the chat server, thank you for visiting us.");
        System.out.println("Press return to exit ...");
        stop(); // stopping client thread
    }
    System.out.println(message+"\n");
}

/* Stops client thread. Closes data streams and socket.
 * Can be called after an error or on properly leaving from server.
 */
public void stop() {
    if (thread != null) {
        thread.stop();
        thread = null;
    }
    // trying to close data streams and socket
    try {
        if (streamIn != null) streamIn.close();
        if (streamOut != null) streamOut.close();
        if (socket != null) socket.close();
    } catch (IOException ioe) {
        System.out.println("Error while closing data streams and server socket. Error: \n" + ioe.getMessage());
    }
    client.close();
    client.stop();
}

public static void main (String args[]) {

    String serverName, nickName;
    int serverPort = 7896;
    Scanner sc = new Scanner(System.in);

    System.out.println("Welcome to CommandLineChat 0.1 alpha");

```

```
System.out.println("=====\\n");
System.out.print("Enter server IP: ");
serverName = sc.nextLine();
System.out.print("Choose your nickname: ");
nickName = sc.nextLine();

TCPClient client = null;

client = new TCPClient(serverName, serverPort, nickName);
}
```