Ulvi Bajarani

SID 20539914

CSCI 6345 – Advanced Computer Networks

Summer II

Assignment 4

The socket programming: The chat program in Java.

**//The code of Handler.java to handle Read threads with ReadHandler class, Write threads with WriteHandler class, and User threads with ThreadHandler class.**

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class Handler extends Thread {

public static class ReadHandler extends Handler {

private BufferedReader ReadHandler;

private Socket theSocket;

private ClientClass theClient;

public ReadHandler(Socket theSocket, ClientClass client) {

this.theSocket = theSocket;

this.theClient = client;

try {

InputStream inputFromTheUser = theSocket.getInputStream();

ReadHandler = new BufferedReader(new InputStreamReader(inputFromTheUser));

} catch (Exception e) {

e.printStackTrace();

}

}

public void run() {

while (true) {

try {

String response = ReadHandler.readLine();

System.out.println("\n" + response);

if (theClient.getTheUserName() != null) {

System.out.print("[" + theClient.getTheUserName() + "]: ");

}

} catch (Exception e) {

e.printStackTrace();

break;

}

}

}

}

public static class WriteHandler extends Handler {

private PrintWriter WriteHandler;

private Socket theSocket;

private ClientClass theClient;

public WriteHandler(Socket theSocket, ClientClass theClient) {

this.theSocket = theSocket;

this.theClient = theClient;

try {

OutputStream OutputForTheUser = theSocket.getOutputStream();

WriteHandler = new PrintWriter(OutputForTheUser, true);

} catch (Exception e) {

e.printStackTrace();

}

}

public void run() {

Console newConsole = System.console();

String theUserName = newConsole.readLine("Please, enter the user name: ");

theClient.setTheUserName(theUserName);

WriteHandler.println(theUserName);

String outputText;

do {

outputText = newConsole.readLine("[" + theUserName + "]: ");

WriteHandler.println(outputText);

} while (!outputText.equals(""));

try {

theSocket.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

public static class ThreadHandler extends Handler {

private Socket theSocket;

private ServerClass theServer;

private PrintWriter WriteHandler;

public ThreadHandler(Socket theSocket, ServerClass theServer) {

this.theSocket = theSocket;

this.theServer = theServer;

}

public void run() {

try {

InputStream inputFromTheUser = theSocket.getInputStream();

BufferedReader theBufferedReader = new BufferedReader(new InputStreamReader(inputFromTheUser));

OutputStream outputFromTheUser = theSocket.getOutputStream();

WriteHandler = new PrintWriter(outputFromTheUser, true);

String userName = theBufferedReader.readLine();

theServer.userNameAdder(userName);

String theMessageFromServer = "We have the new user: " + userName;

String theMessageFromClient;

do {

theMessageFromClient = theBufferedReader.readLine();

theMessageFromServer = "[" + userName + "]:" + theMessageFromClient;

theServer.broadcast(theMessageFromServer, this);

} while (!theMessageFromClient.equals(""));

theServer.removeUser(userName, this);

theSocket.close();

theMessageFromServer = userName + " has quitted.";

theServer.broadcast(theMessageFromServer, this);

} catch (Exception e) {

e.printStackTrace();

}

}

public void sendMessage(String message) {

WriteHandler.println(message);

}

}

}

**//The code of TheServerAndClientChatApp.java that contains the main function.**

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class TheServerAndClientChatApp {

public static void main(String args[]) {

System.out.println("Enter either \"Client\" or \"Server\". If it is the first execution, type \"Server\".");

Scanner roleInput = new Scanner(System.in);

String role = roleInput.nextLine();

try {

if (role.equalsIgnoreCase("server")) {

System.out.println("Enter the port number: ");

Scanner portInput = new Scanner(System.in);

int port = portInput.nextInt();

ServerClass start = new ServerClass(port);

start.execute();

} else if (role.equalsIgnoreCase("client")) {

System.out.printf("Enter the hostname: %n");

Scanner hostInput = new Scanner(System.in);

String hostname = hostInput.nextLine();

System.out.printf("Enter the port number: %n");

Scanner portInput = new Scanner(System.in);

int port = portInput.nextInt();

ClientClass start = new ClientClass(hostname, port);

start.execute();

} else {

System.out.printf("Execute the program again with right inputs.%n");

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

**//The code of ClientClass.java to have the client details and functions.**

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class ClientClass {

private String HostName;

private int ClientPort;

private String userName;

public ClientClass(String HostName, int ClientPort) {

this.ClientPort = ClientPort;

this.HostName = HostName;

}

public void execute() {

try {

Socket theClientSocket = new Socket(HostName, ClientPort);

System.out.printf("The connection is occured in Hostname %s and port %d%n", HostName, ClientPort);

new Handler.ReadHandler(theClientSocket, this).start();

new Handler.WriteHandler(theClientSocket, this).start();

} catch (Exception e) {

e.printStackTrace();

}

}

public String getTheUserName() {

return this.userName;

}

public void setTheUserName(String userName) {

this.userName = userName;

}

}

**//The code of ServerClass.java to have the server details and functions.**

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class ServerClass {

private int ServerPort;

private static Set<String> ListOfRegisteredUsers = new HashSet<>();

private static Set<Handler.ThreadHandler> ListOfRegisteredUserThreads = new HashSet<>();

public ServerClass(int ServerPort) {

this.ServerPort = ServerPort;

}

public void execute() {

try (ServerSocket TheNewServerSocket = new ServerSocket(ServerPort)) {

System.out.printf("The session is started on port %d%n", ServerPort);

while (true) {

Socket theSocket = TheNewServerSocket.accept();

System.out.println("New user connected");

Handler.ThreadHandler newEnteredUser = new Handler.ThreadHandler(theSocket, this);

ListOfRegisteredUserThreads.add(newEnteredUser);

newEnteredUser.start();

System.out.printf("The connection by the user is started on port %d%n", ServerPort);

}

} catch (Exception e) {

e.printStackTrace();

}

}

public void broadcast(String message, Handler.ThreadHandler excludeUser) {

for (Handler.ThreadHandler someUser : ListOfRegisteredUserThreads) {

if (someUser != excludeUser) {

someUser.sendMessage(message);

}

}

}

public void removeUser(String userName, Handler.ThreadHandler aUser) {

if (ListOfRegisteredUsers.remove(userName) == true) {

ListOfRegisteredUserThreads.remove(aUser);

System.out.println("The user " + userName + " quitted");

}

}

public void userNameAdder(String userName) {

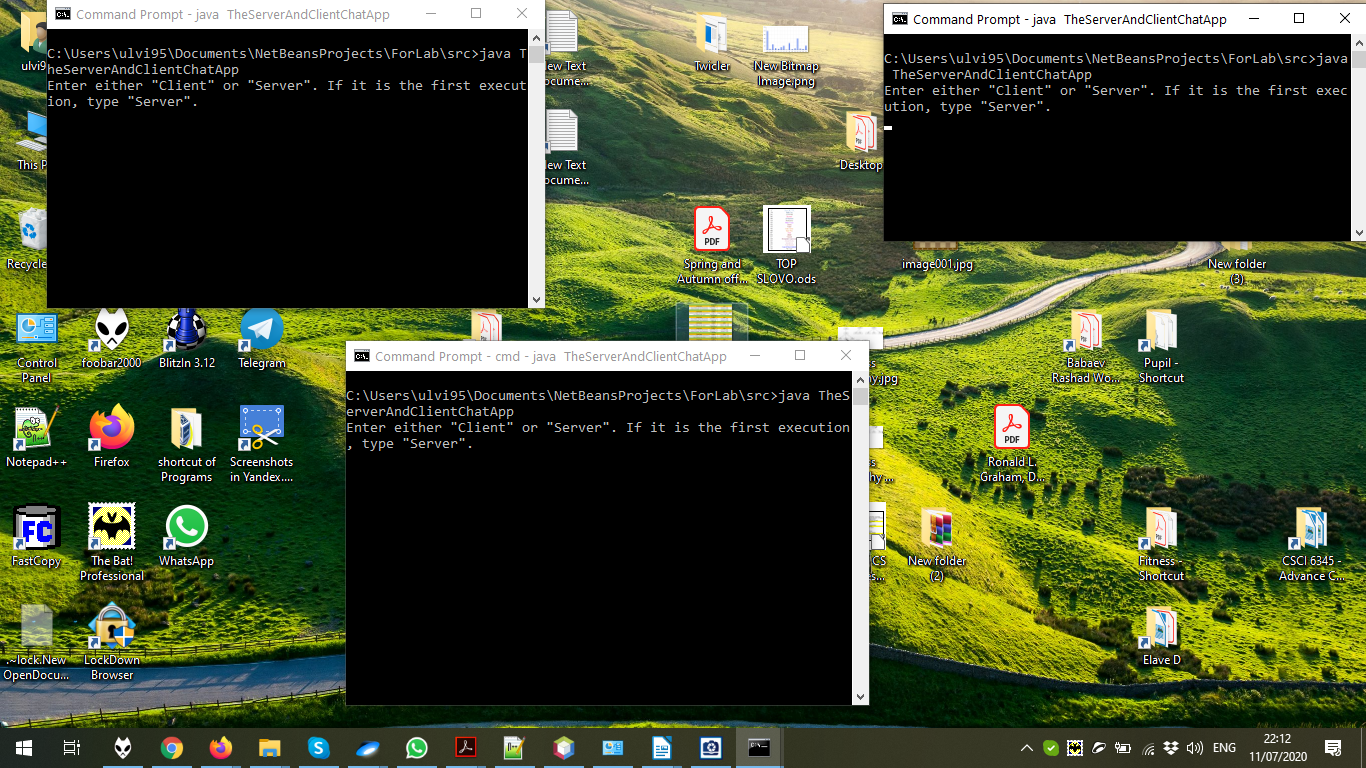
ListOfRegisteredUsers.add(userName);

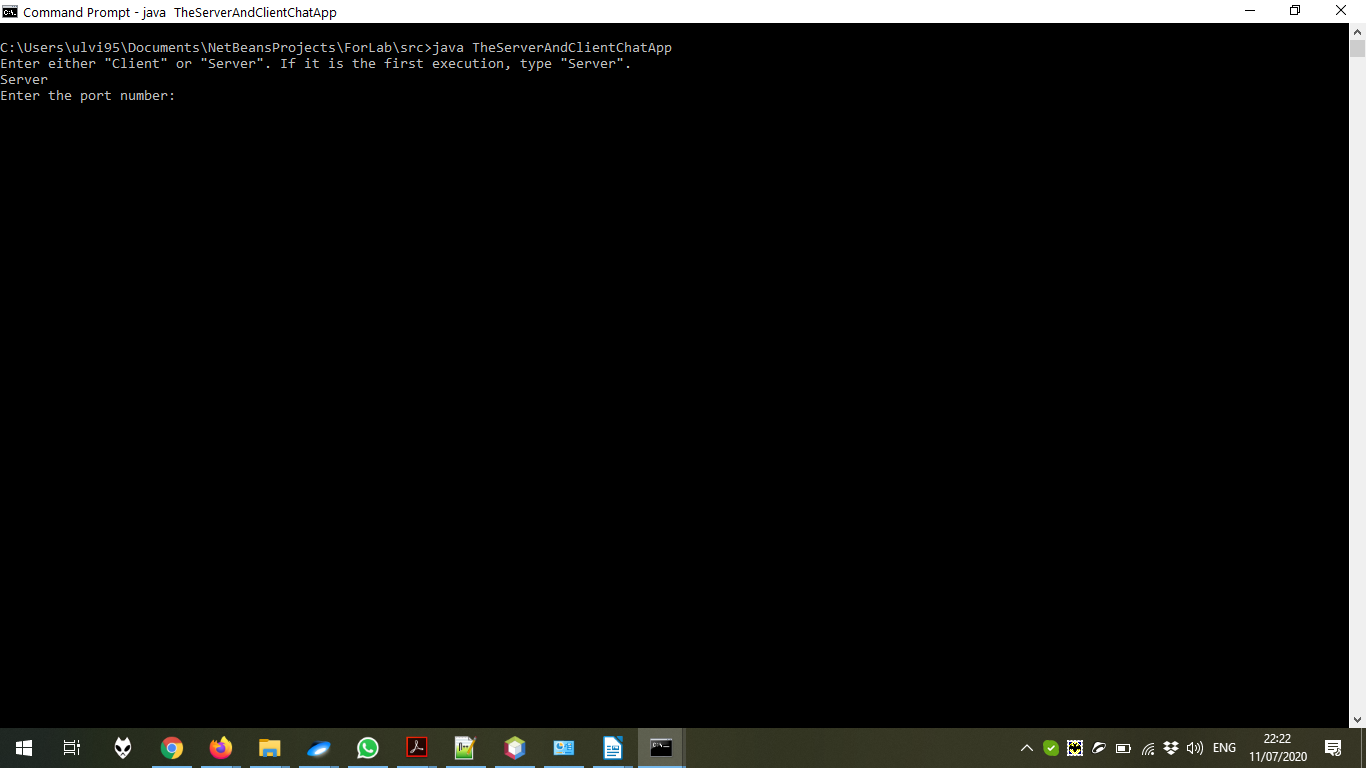
}

}

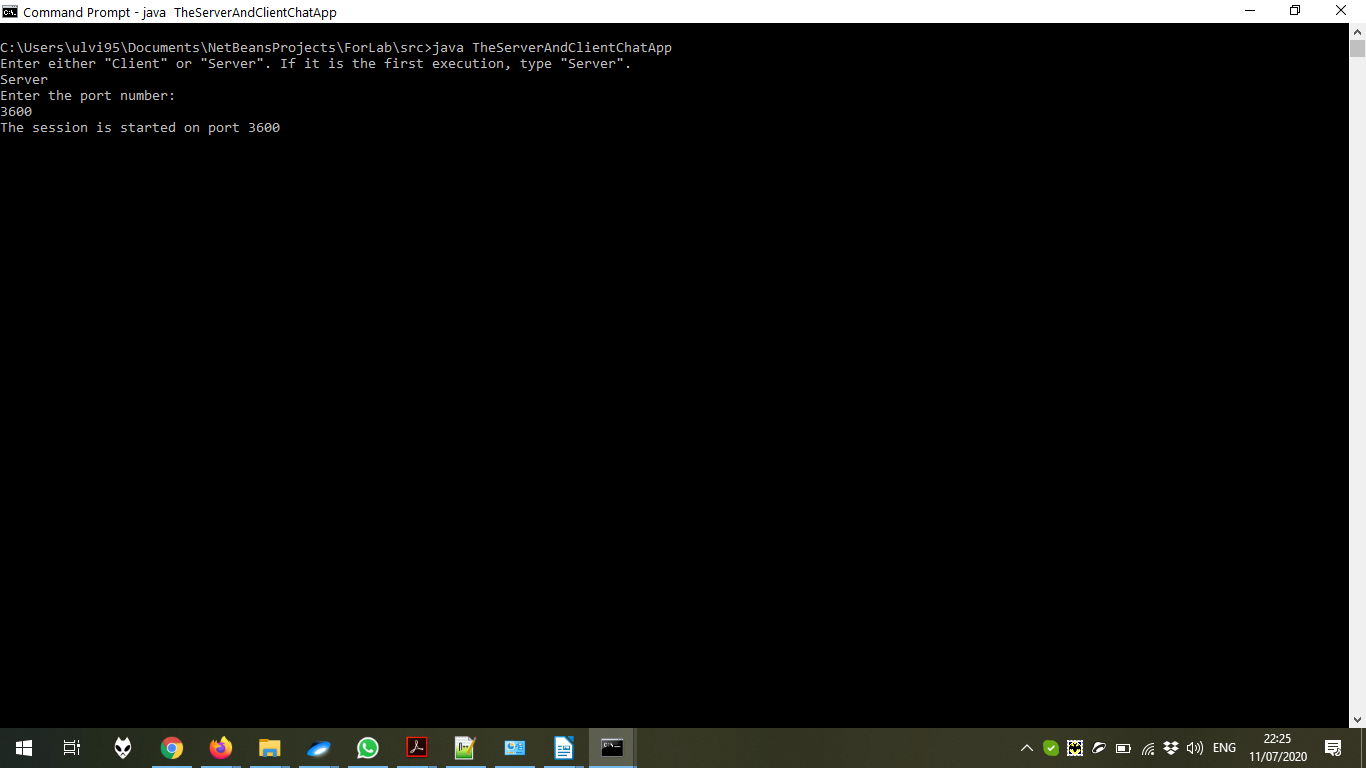
The screenshots of the program work. It is important to remember that in the first execution, the Server option should be chosen to avoid errors. The users cannot have the same name. The termination of Client session is done by “” input, while for the termination of the Server, the Ctrl+C on the keyboard should be used. All executions are tested in the localhost, and all command prompt’s are different. The Java JDK 14 is used for the program. All .java files are compiled by **javac \*.java** command, and the program is executed by **java TheServerAndClientChatApp** command. The next screenshots describes the work of the programs.

The initial execution

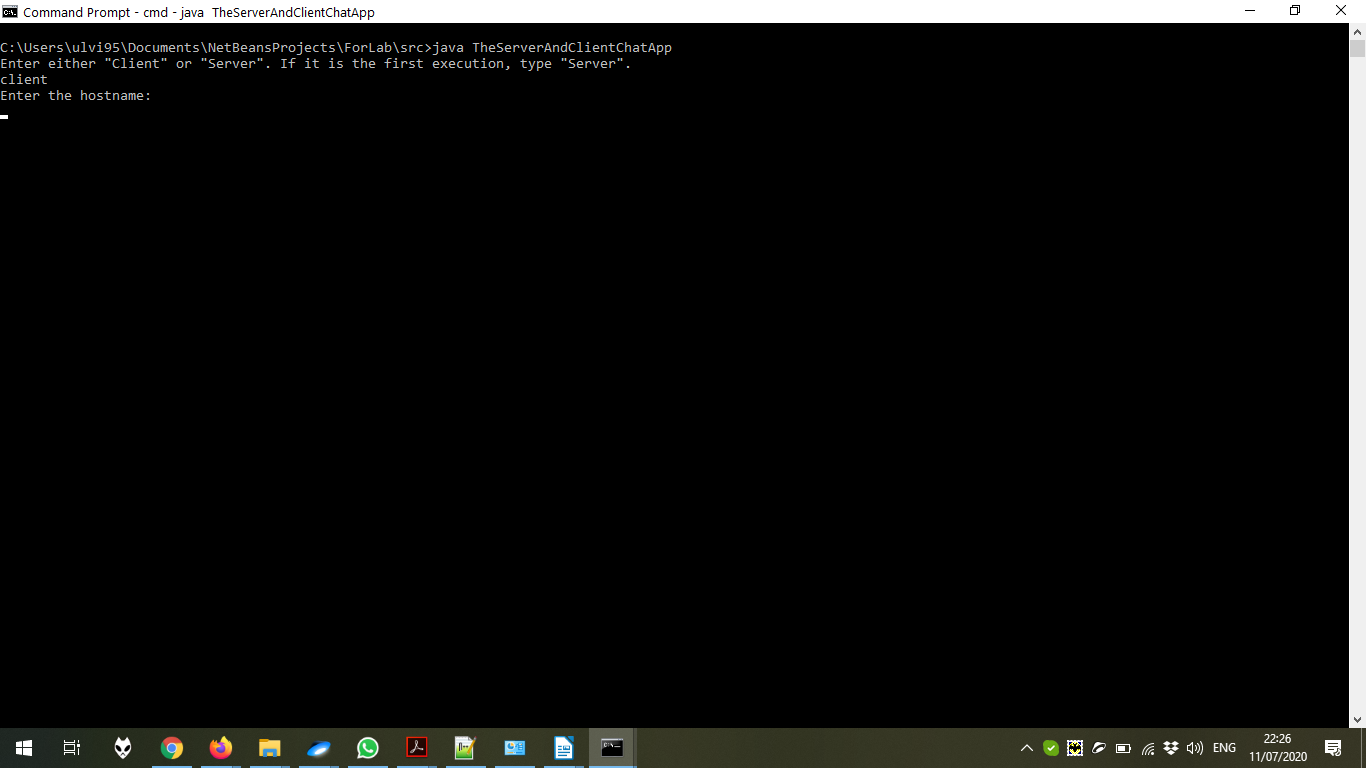


Creating 1 server.

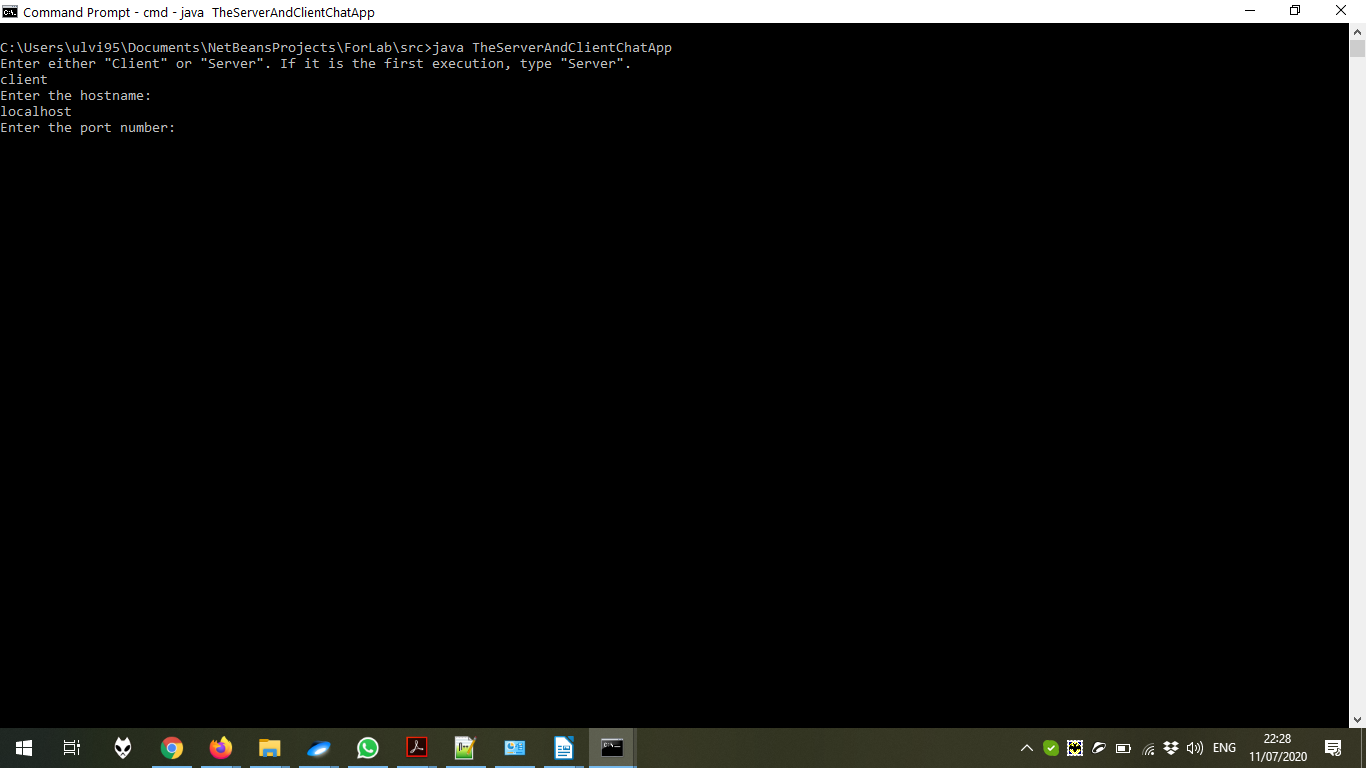
Entering the port number. I have chosen **3600**.



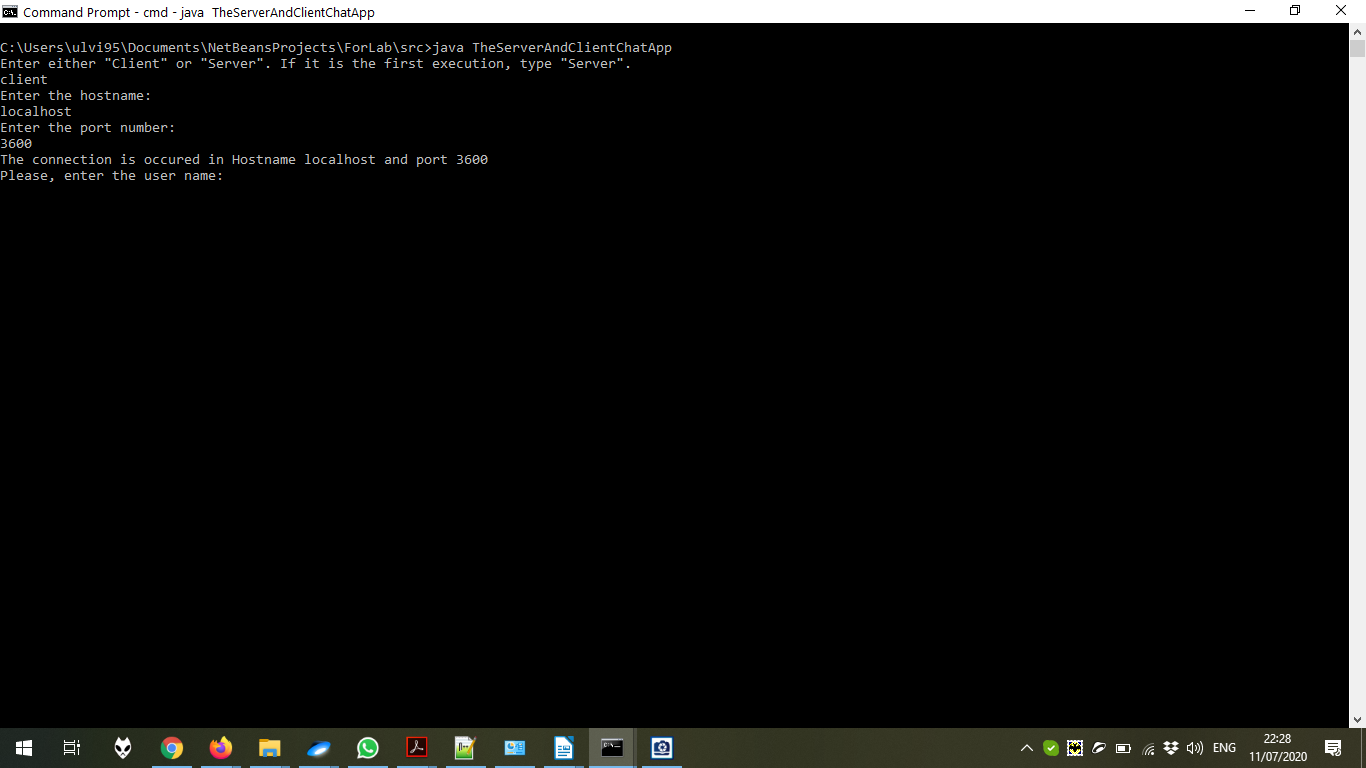
In the client, the Host name should be chosen first.



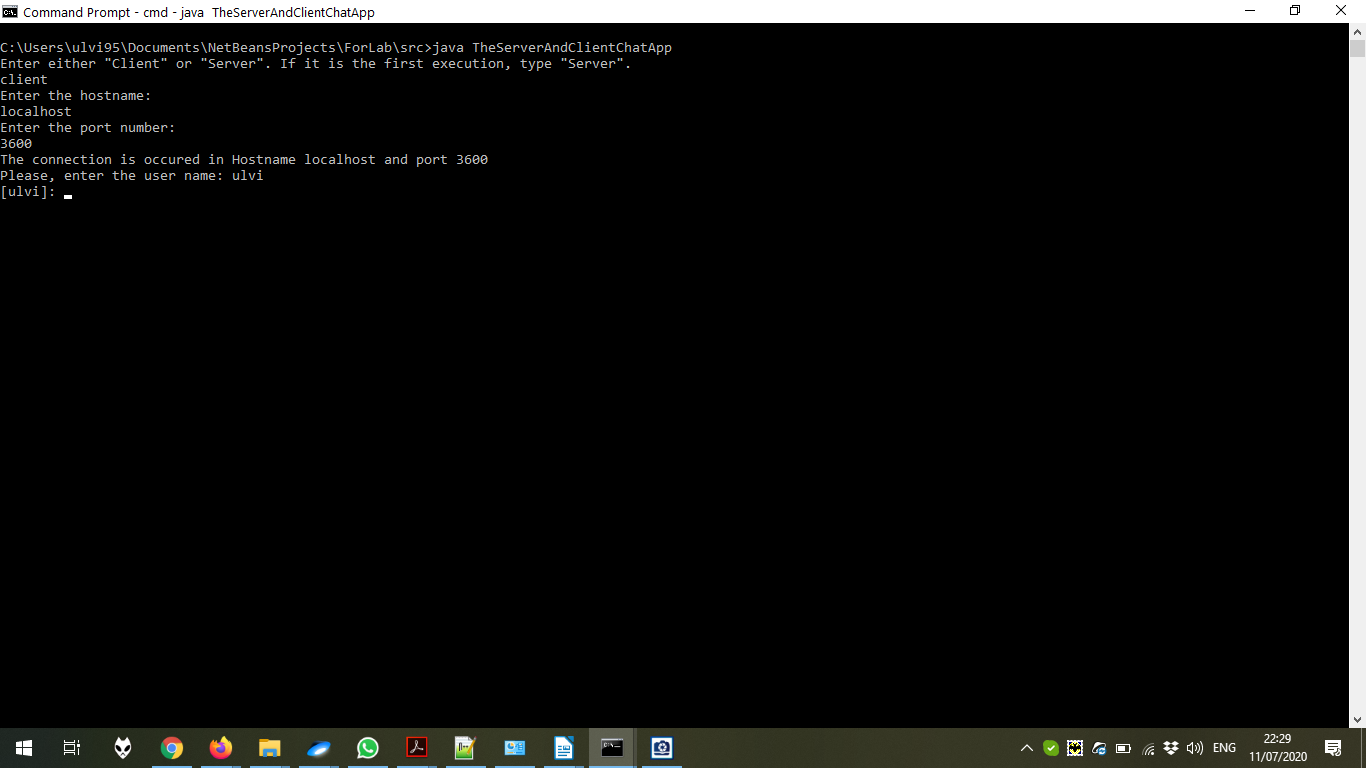
After choosing **localhost**, the port number should be defined.



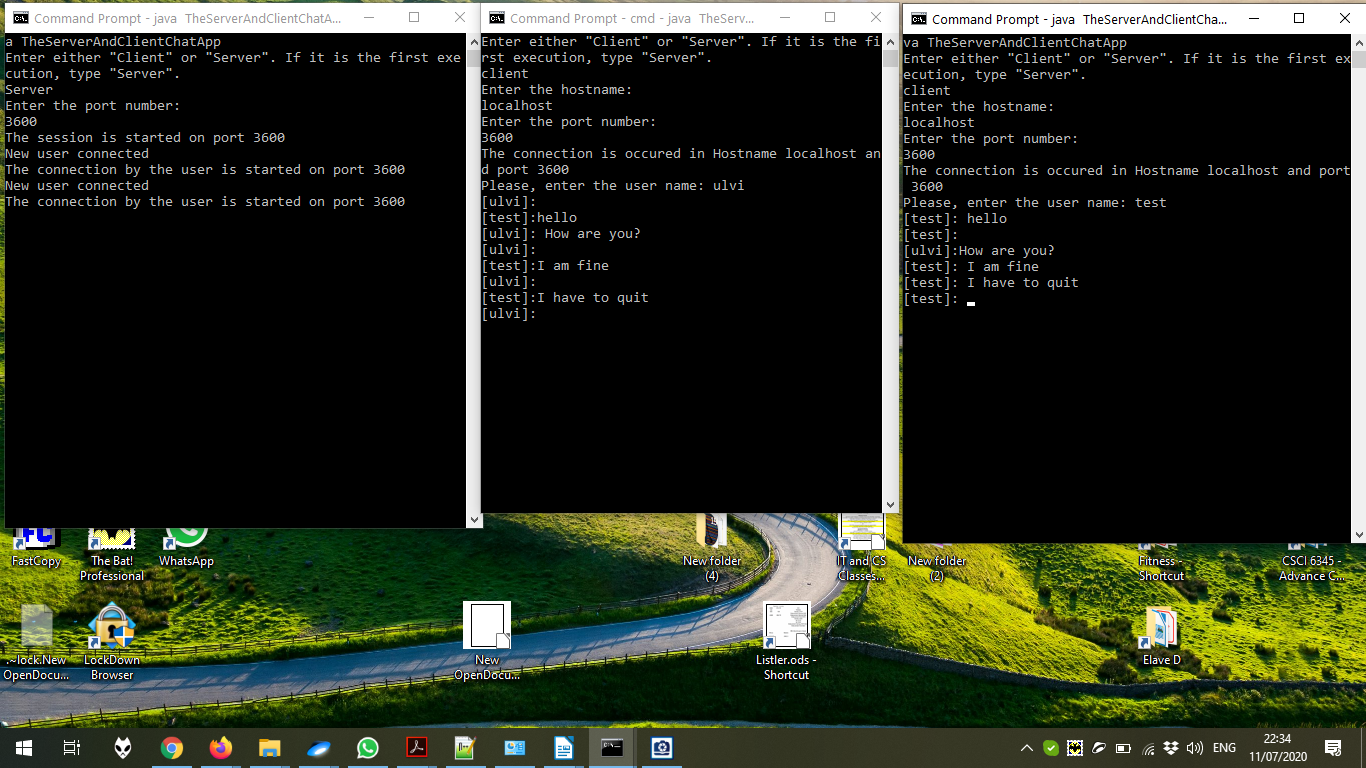
The same **3600** port number is chosen.



Now, the user name is defined.



After doing the same executions for the next client with **test** user name, we can test the chat:



The ending of the sessions:

