

**CS330–Computer Networks Project**

**TCP-based client server application**

**Supervised by :**

**Dr.Basmah Alsouly**

**Students Names:**

|  |  |
| --- | --- |
| **Student Name** | **Id** |
| **Sara Ibrahim Al Mashharawi** | **440028560** |
| **Hanin Alanazi** | **440021299** |
| **Rawan Saad Alshalawi** | **440018784** |
| **Reenad slaiman Alharbi** | 437000339 |

**Section:** **372**

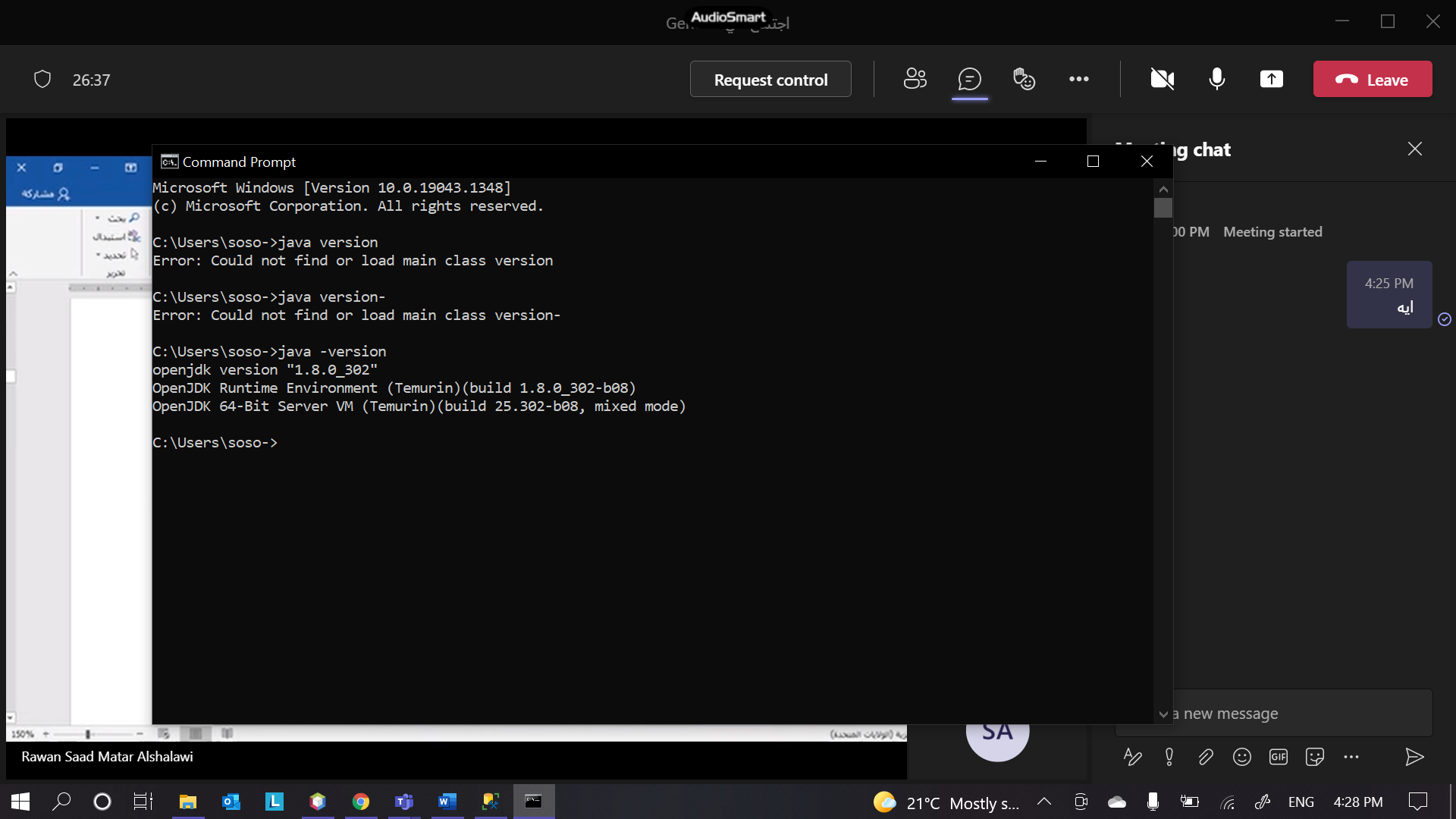
**Submission date** : **11/12/2021**

**1-Setting up the Programming Environment :**

We use JAVA programming language . since we have good experience in this language because we have studied it before and it is object-oriented language that has previous defined libraries such as IO and net .

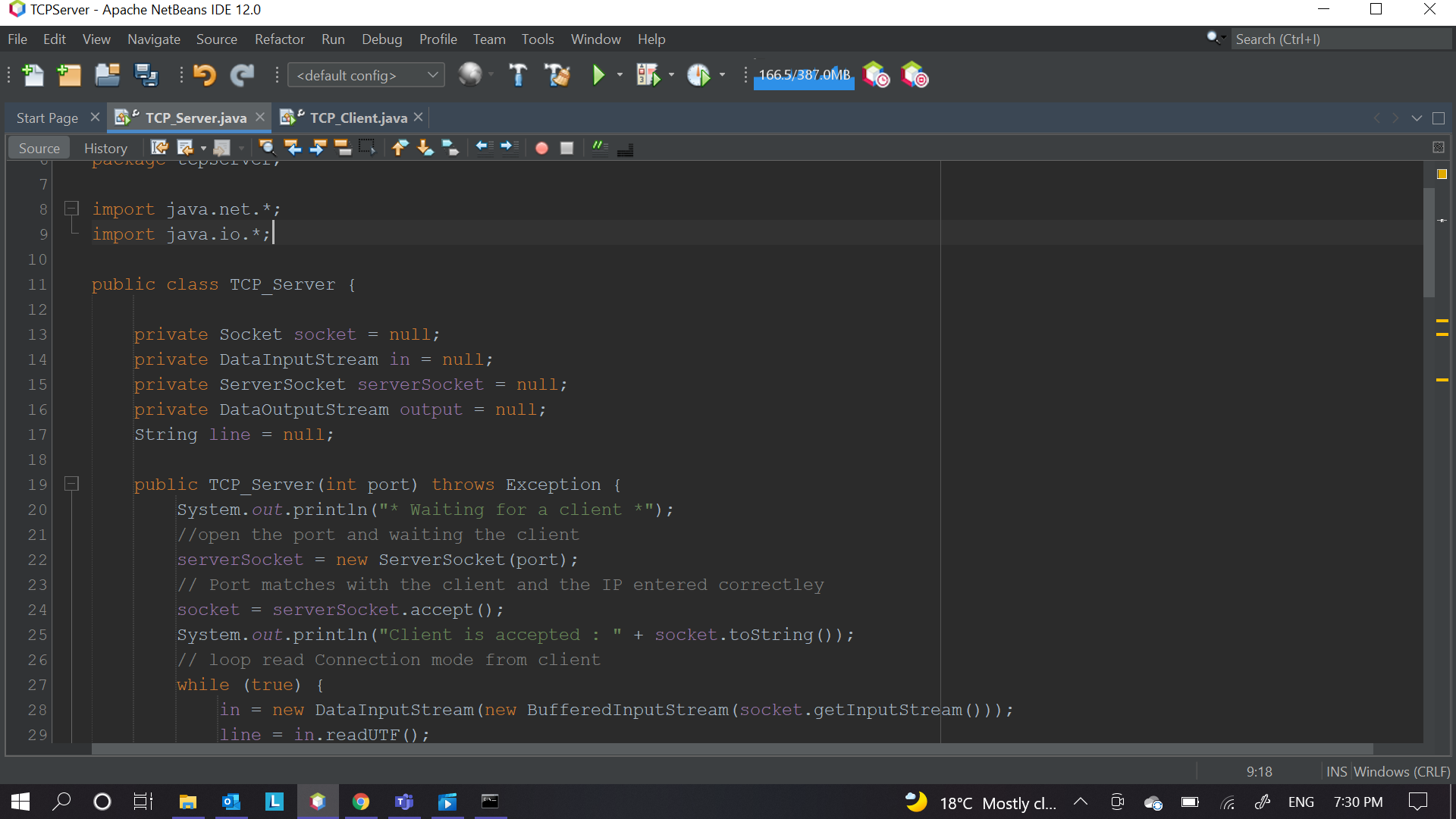
* JDK installation :

First, install JDK 1.8.0\_302 for windows 10 from ORACLE website . we use JDK Libraries to import Socket Programming library .

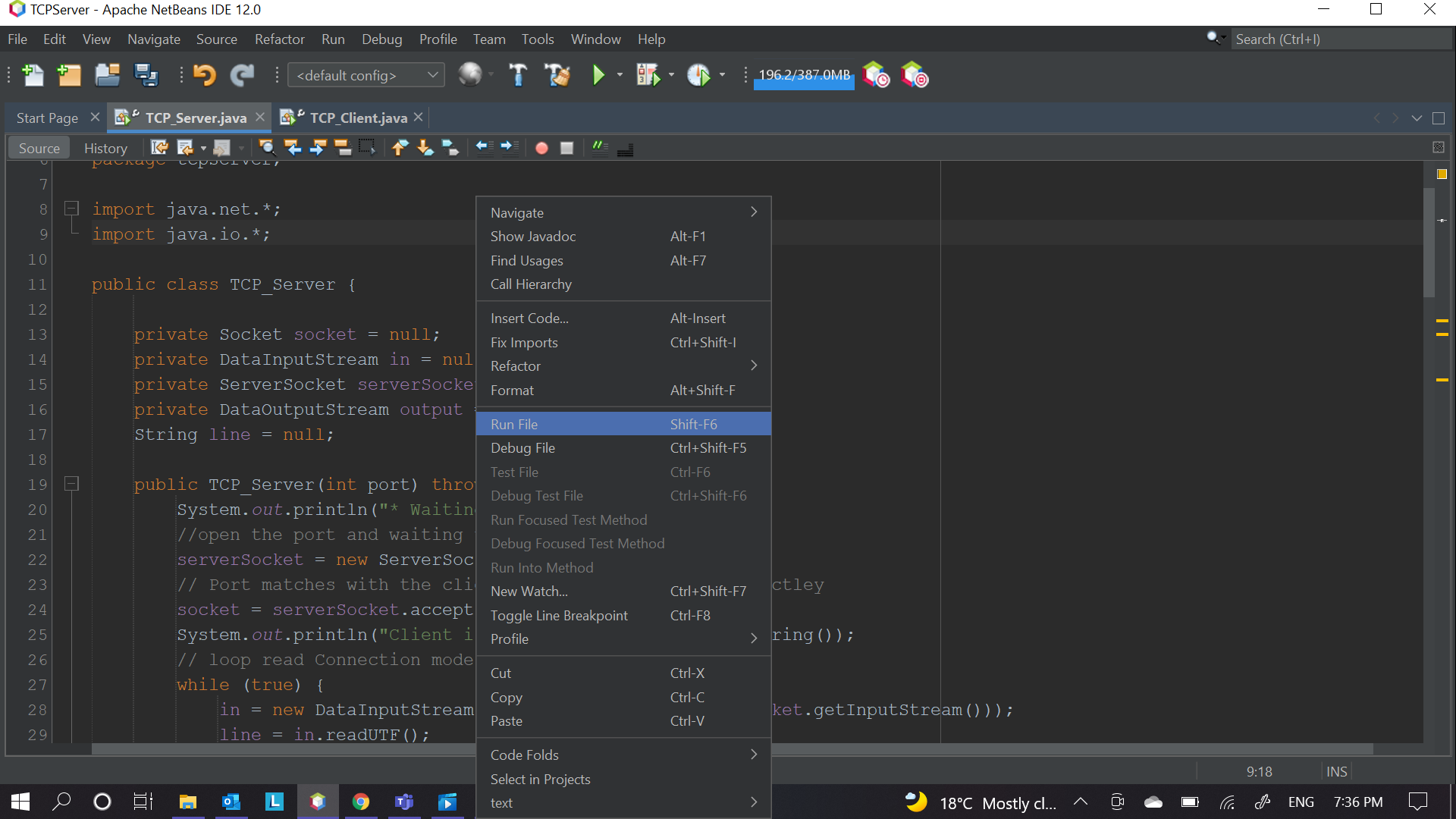


* NetBeans IDE :

We install netbeans from <https://netbeans.apache.org/> version 12.0



To run and compile a program in NetBeans press shift+f6 OR click right then choose Run File .

****

**2-Steps for TCP socket programming for client-server connection :**

We use **Java** socket programming to implement the client- server communication over TCP protocol .

We implement the basic algorithm of TCP programming from these resources :

First website :<https://www.geeksforgeeks.org/socket-programming-in-java/> .

Second website : <https://github.com/karanheart96/TCP_Pinger> .

* **The server will create the socket using:**

Socket package that makes one side connection of a two-way communication link between two programs running on the network and ServerSocket package to listen on a specific port.

1- we create socket to open the port and waiting the client to connect :

serverSocket = new ServerSocket(port);

2- Port matches with the client and the IP entered correctly so the client accepted and communicate

socket = serverSocket.accept();

3- Start reading from a socket using DataInputStream and its method readUTF() . Note : this step is for reading the connection state .

private DataInputStream in = null;

line = in.readUTF();

4-After reading the connection state go to method readingFromSocket() that will go switch(line) and process the connection that will read the massage that client will send according to the mode by using :

line = in.readUTF();

[Note : Secure mode is implemented in a for-loop that shift the letters by the key that the client will send it to the server and server will read it using key = in.readByte(); ] .

5-Now the massage is with the server , the server will resend the message as response for the client by writingIntoSocket(line) method :

output = new DataOutputStream(socket.getOutputStream());

output.writeBytes(msg + '\n');

6- will repeat these steps until read “ close “ , then it will close the connection.

* **The client will create :**

Socket package that is create one side connection of a two-way communication link between two programs running on the network .

1- initialize IP address and port number for server and connect with server if it was running

socket = new Socket("192.168.8.157", 50000);

2- Start writing through socket using DataInputStream and DataOutputStream .

DataInputStream input;

DataOutputStream out;

first, it will read from terminal the massage that client write by :

input = new DataInputStream(System.in);

String option = input.readLine();

then will send the massage through the socket using method that have client massage as parameter writingIntoSocket(option) :

out = new DataOutputStream(socket.getOutputStream());

out.writeUTF(line);

[Note : here will read the option that client choose ].

3-Now it will send the message in socket based in mode that client chose using switch(line)

line = input.readLine();

out.writeUTF(line);

[Note : Secure mode is implemented in a for-loop that shift the letters by the key that the client will send it to the server by out.writeByte(key);] .

4- Then it will wait the server to response his message by using

readingFromSocket(mode,key)

[Note : key is 0 in case open or close ].

5- inside the method readingFromSocket uses

input = new DataInputStream(socket.getInputStream());

String fromServer = input.readLine();

to read server response .

6- will repeat these steps until write“ close “ then send it, and receive closing massage from the server then it will close the connection.

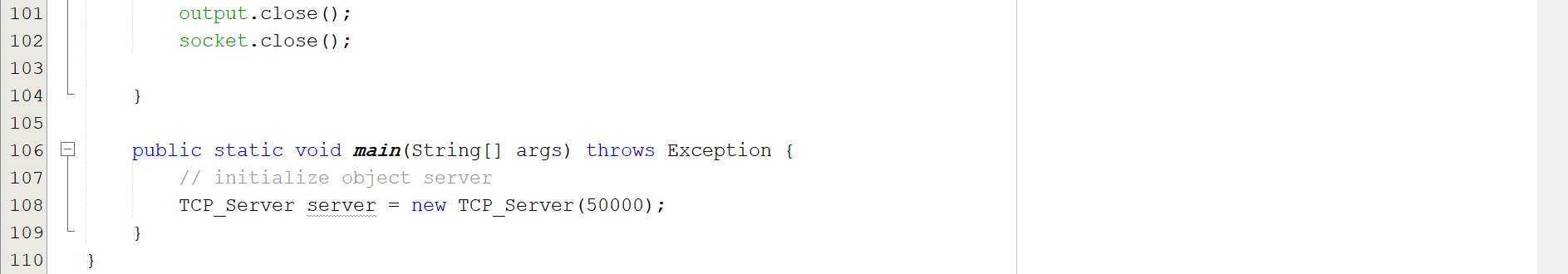
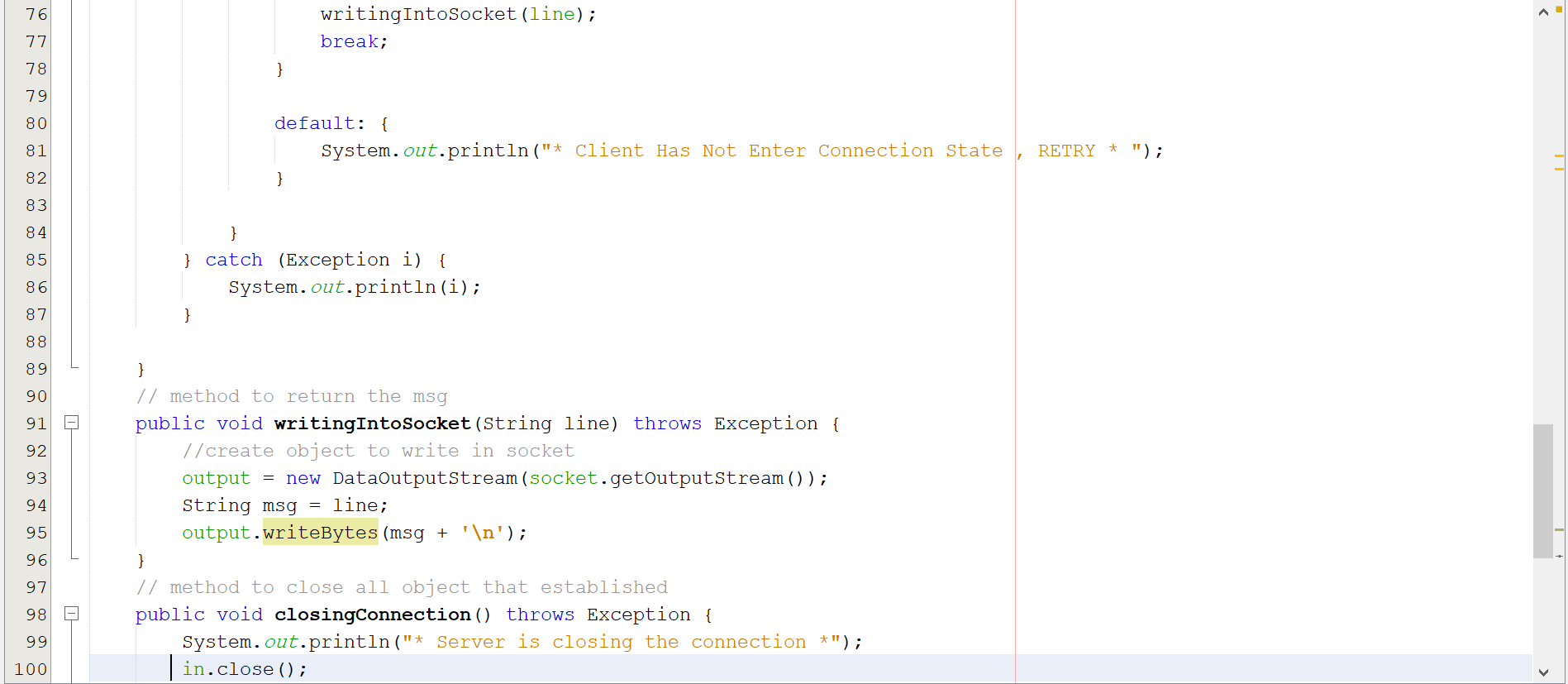
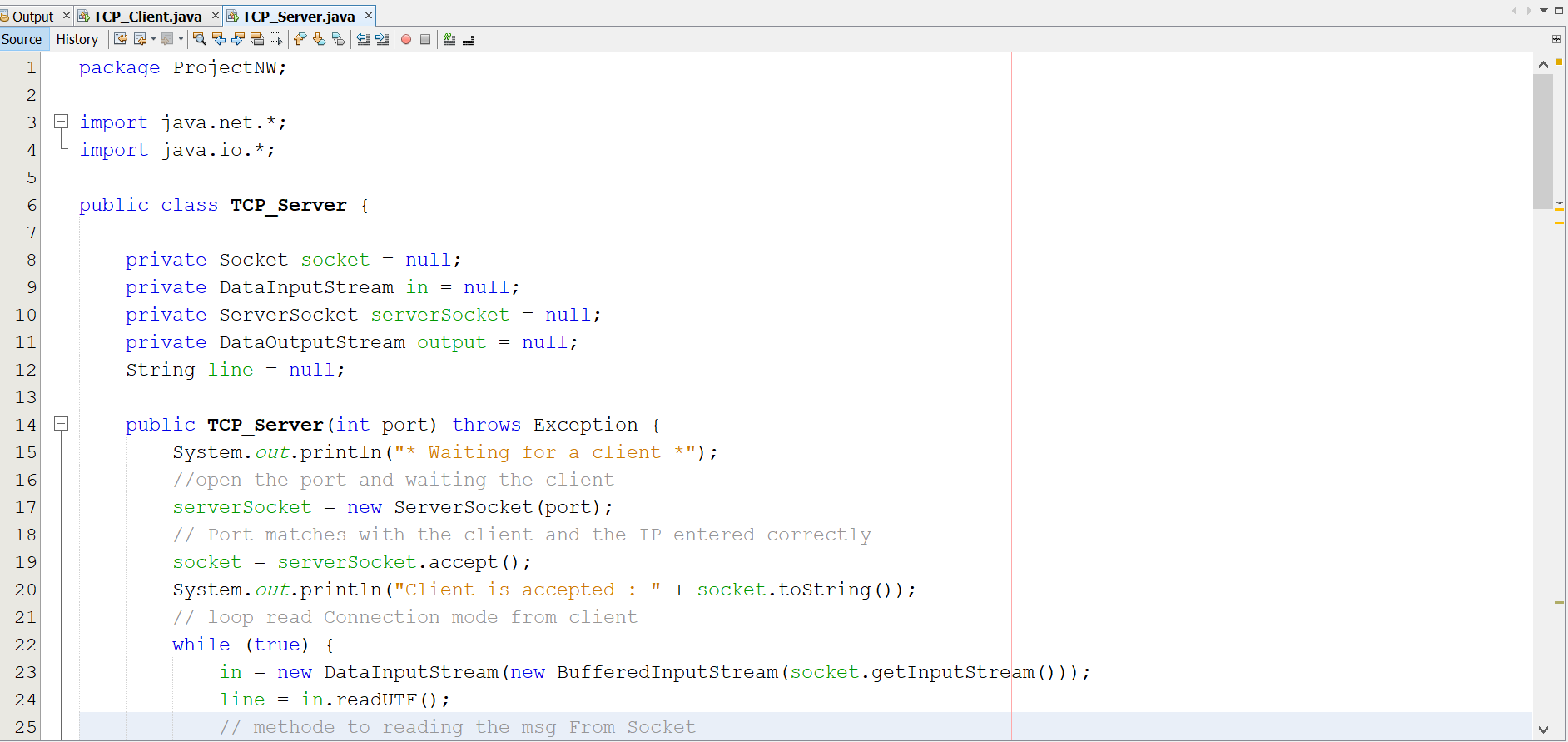
**3-Steps for setting up the network :**

We use two laptops, one is a client and the other one is a server and we connect them local wirelessly through protocol 802.11 using the server IP address 192.168.8.157.

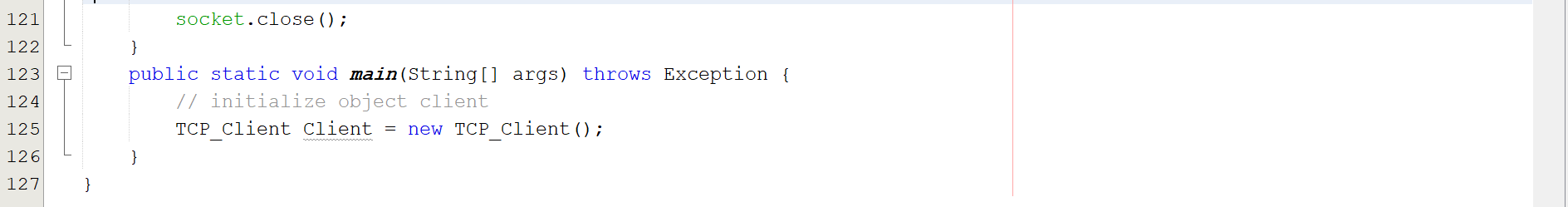
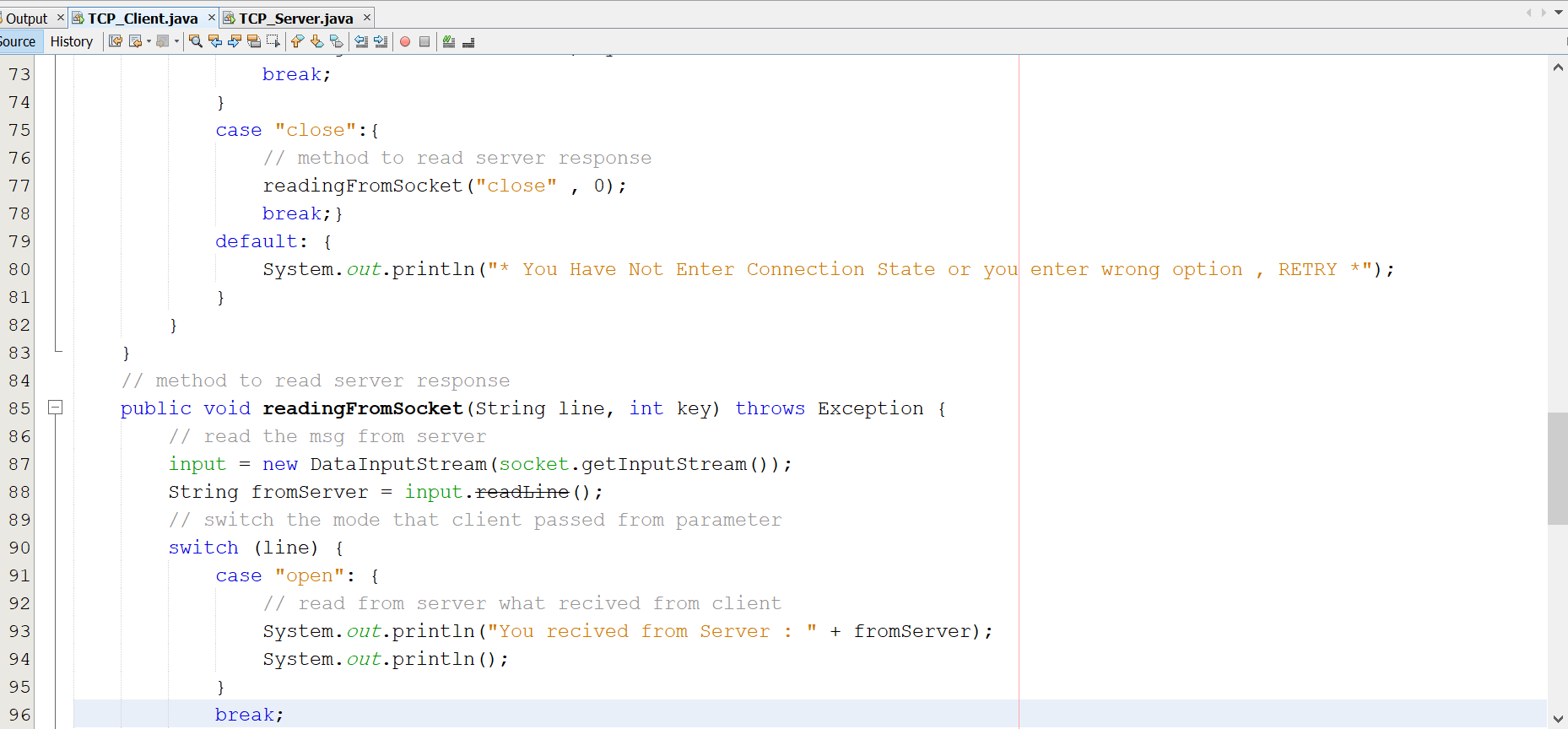
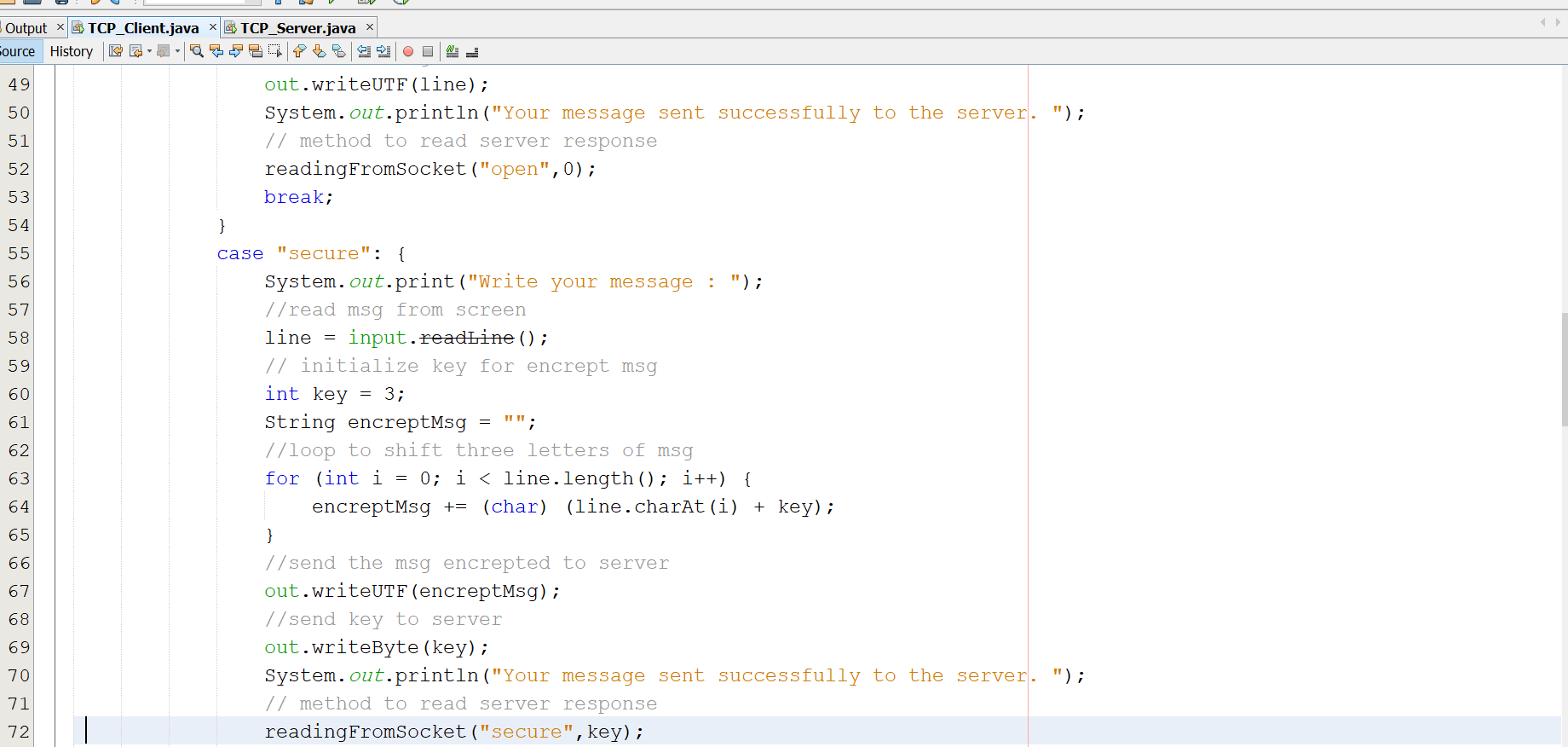
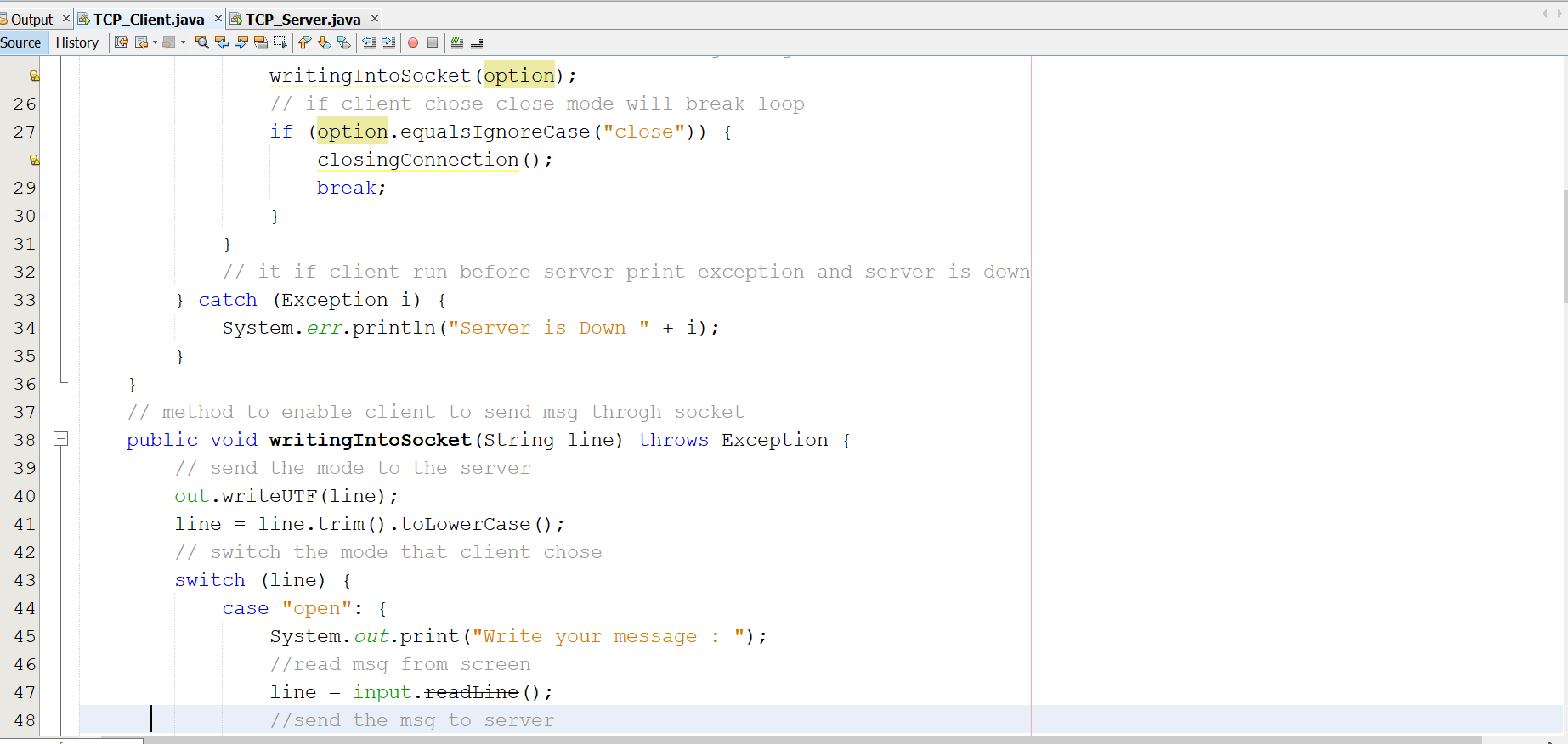
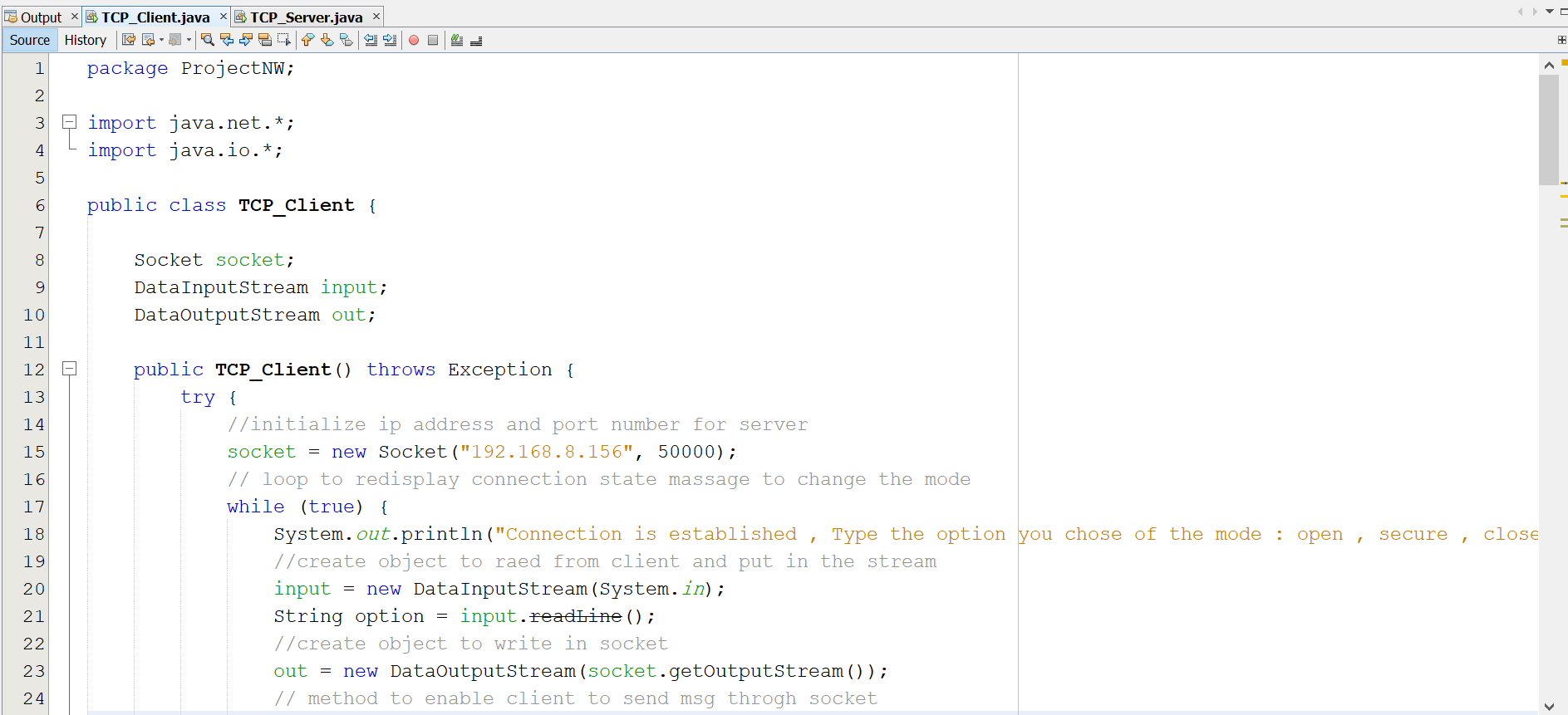
demo link : <https://mega.nz/file/UxdjAawA#CRdQV6_dwKL6W_eDYj7W6oOGN-lflBjxB-fUrC3ziAc>

**4- Codes and comments:**

**Code of server side:**

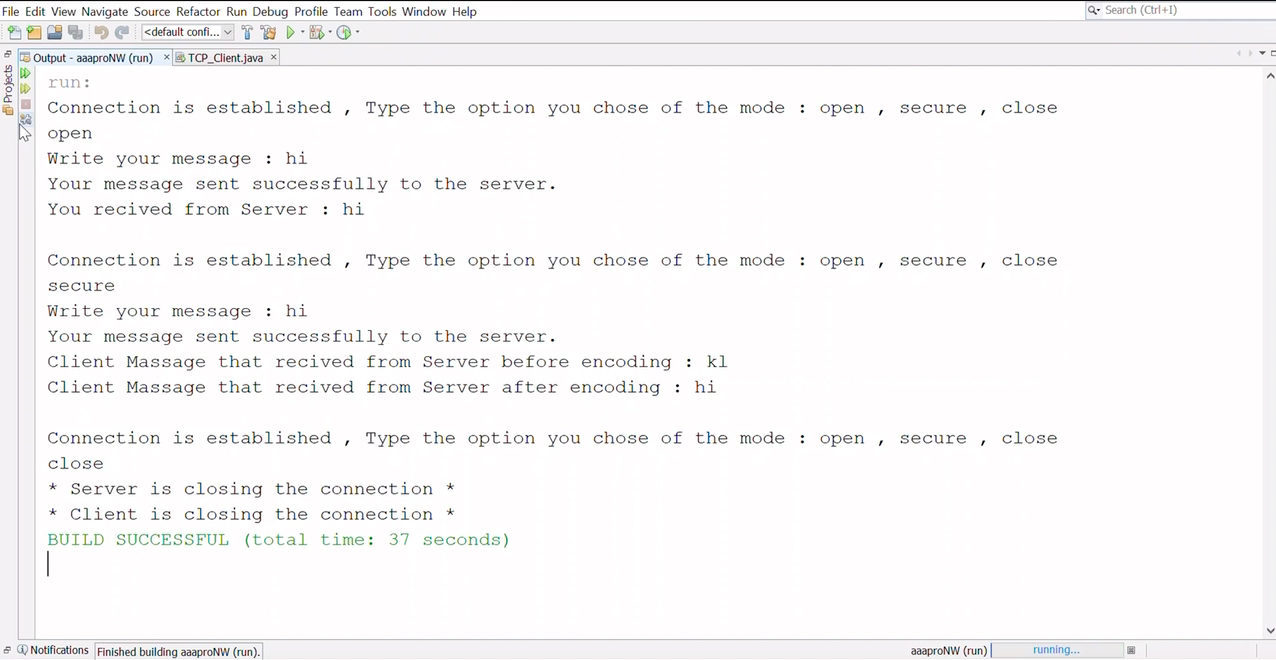


**Code for client side:**

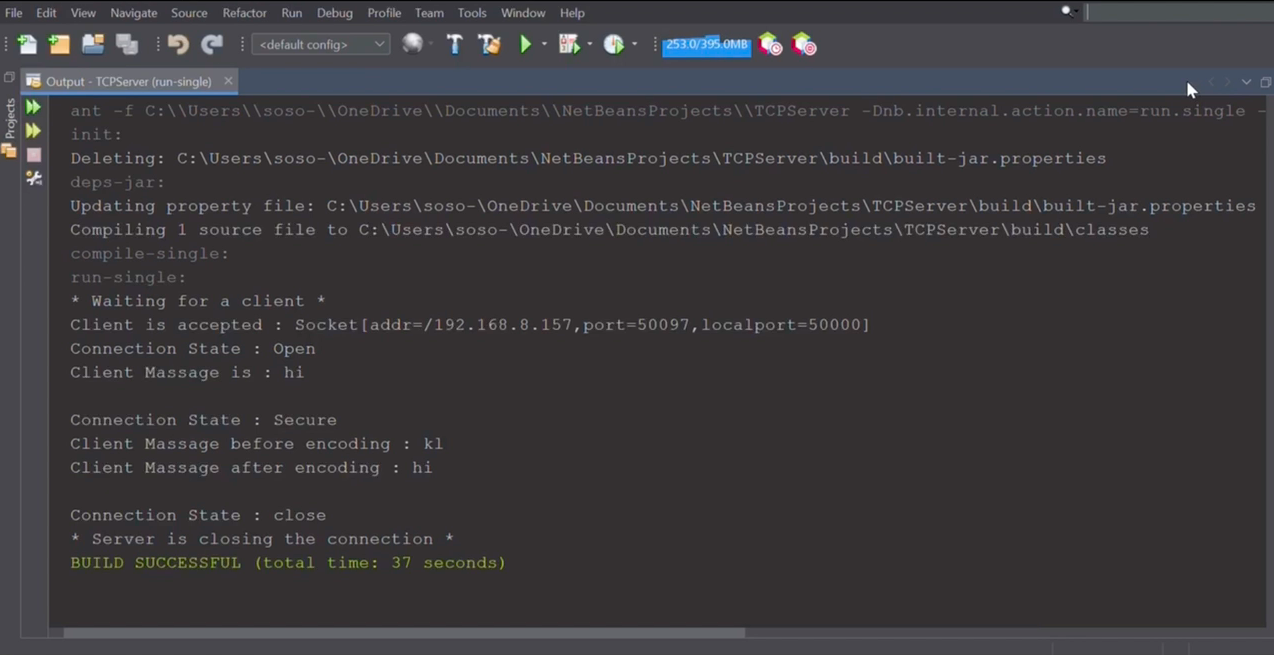
****

**5- Snapshots of the application outputs :**

**Client Output :**

****

**Server Output :**

****

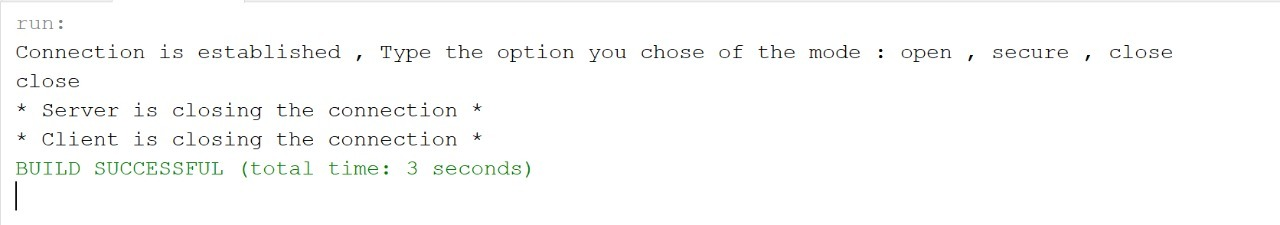
**6-Problems and solutions :**

**Problem 1:** NullPointerException : when we close the connection this exception appears in server output

**output TCP\_Server**

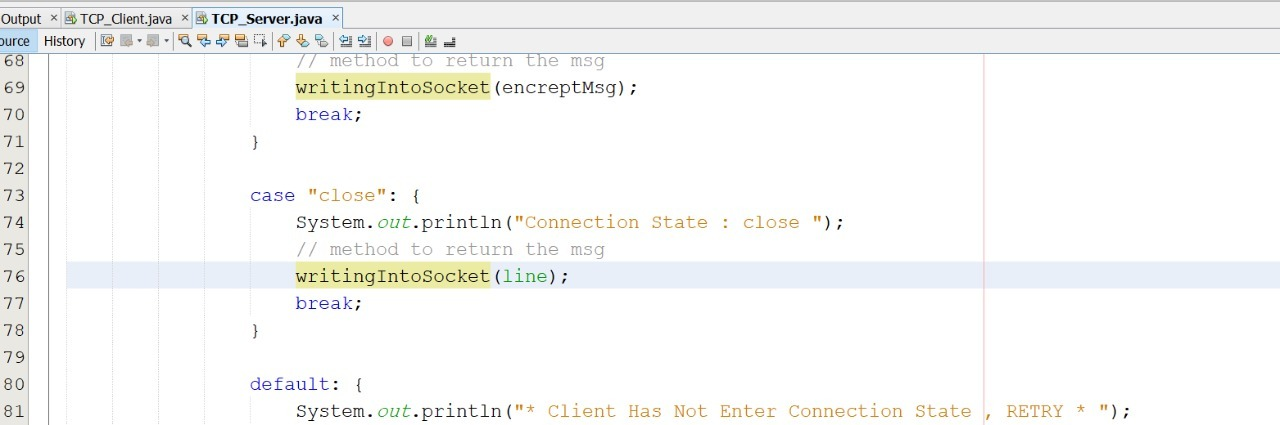


**output TCP\_Client**

****

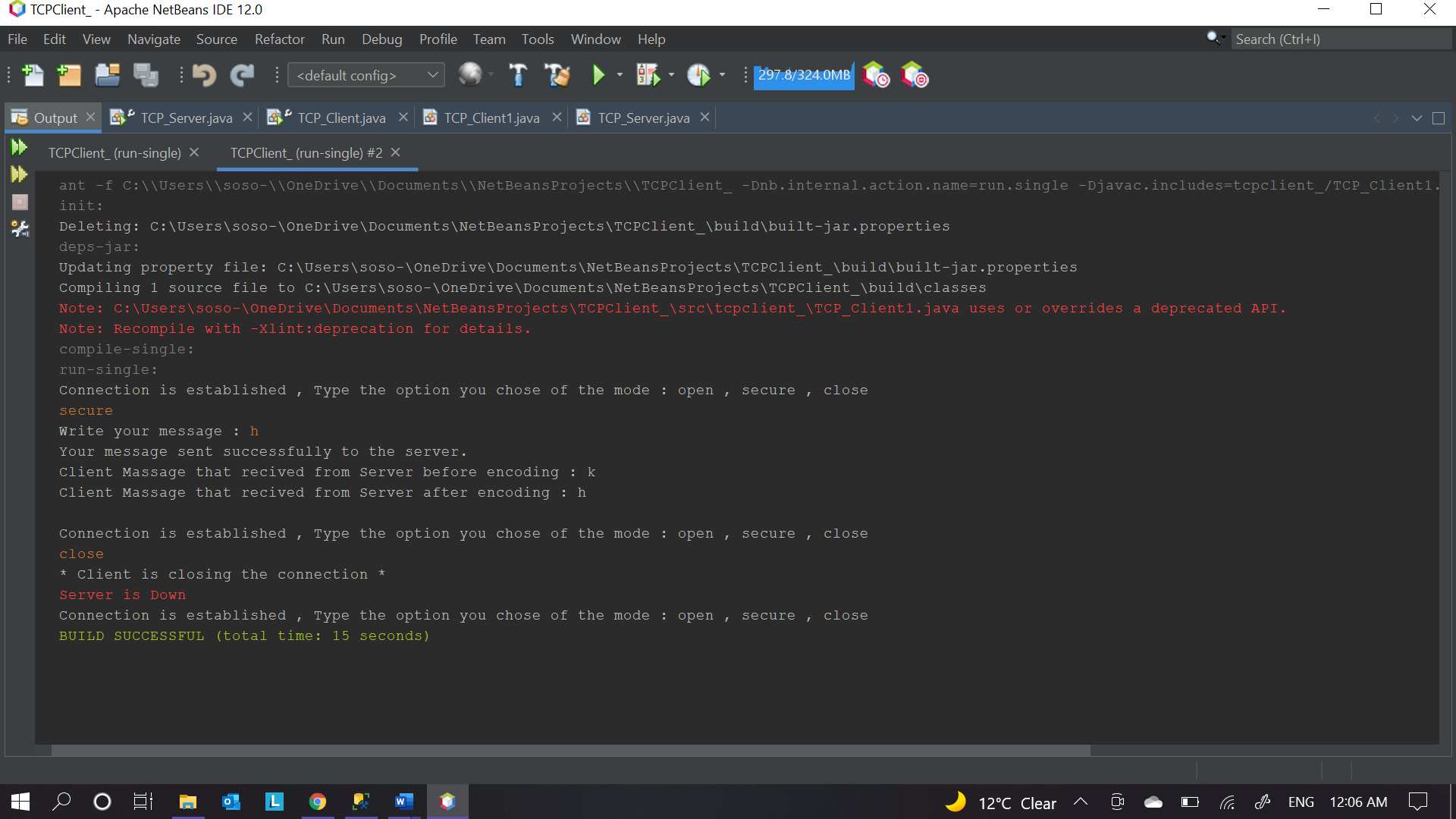
**Solution:**

add method in line 76 to let the server send in any way that it will close the connection

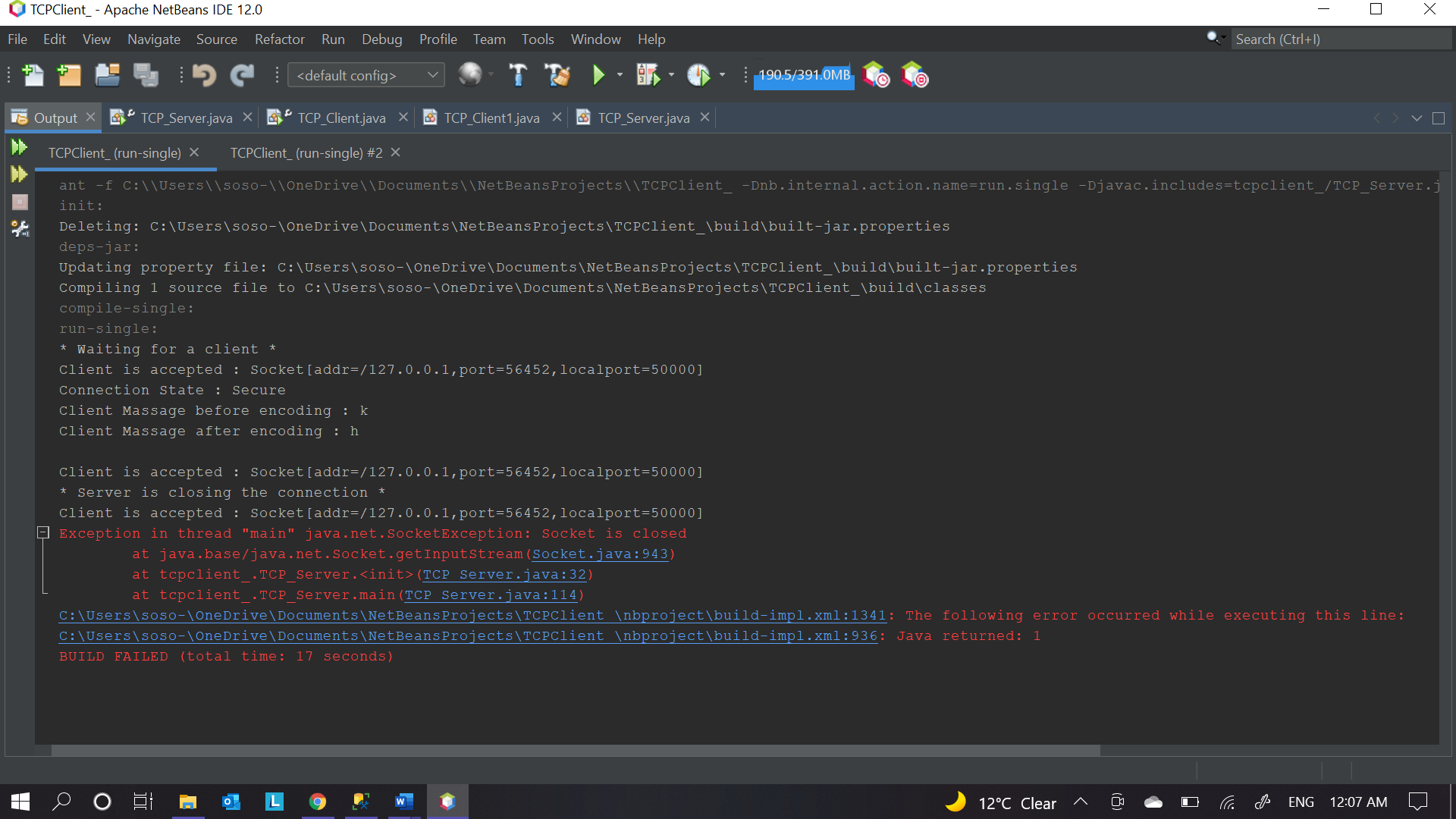


**Problem 2:** Not closing the connection successfully :

**output TCP\_Client**

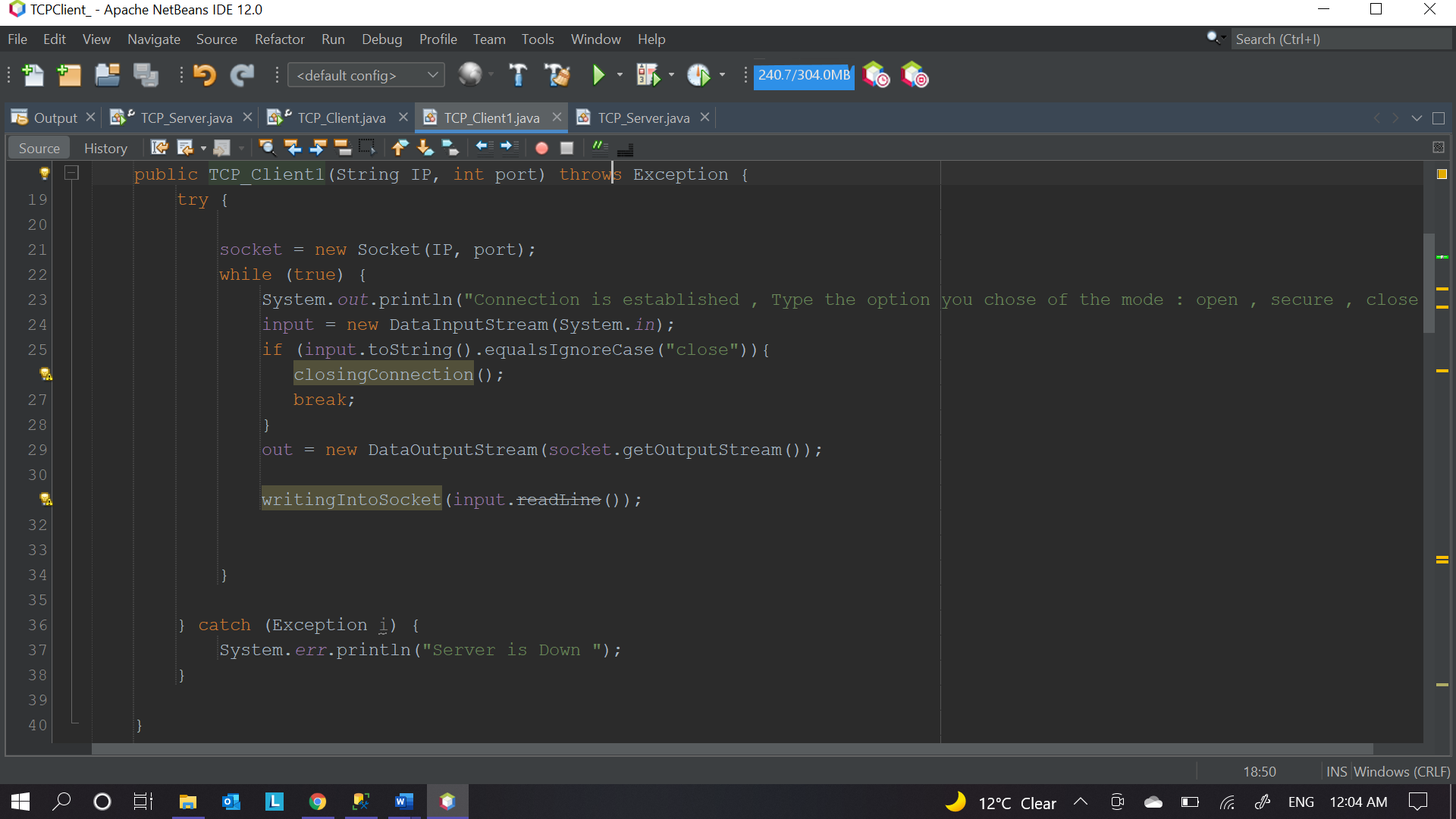


**output TCP\_Server**

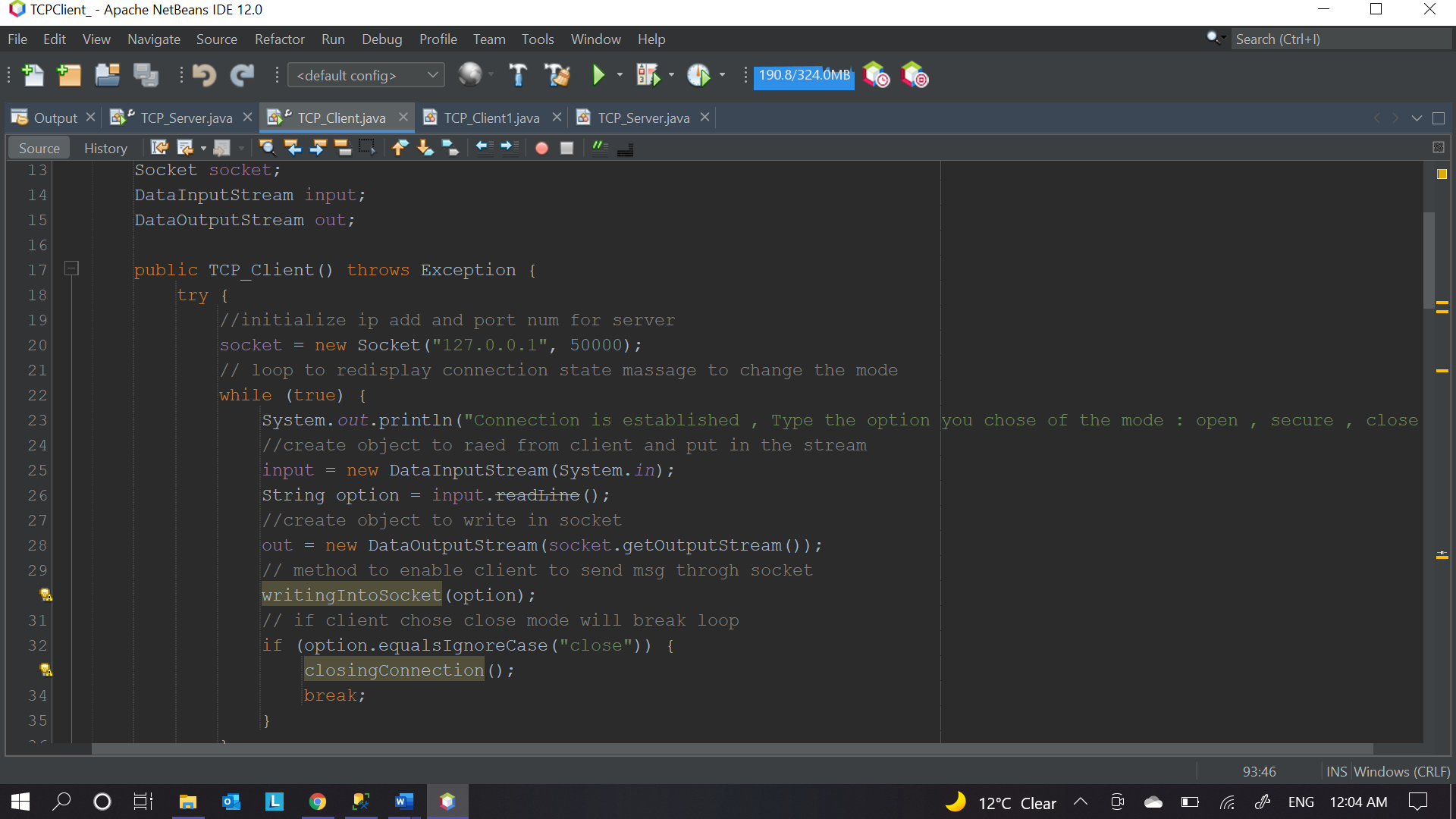
****

**Solution:** enforce the client to close the connection after server’s closing response send and reorder if statement .

**Before :**

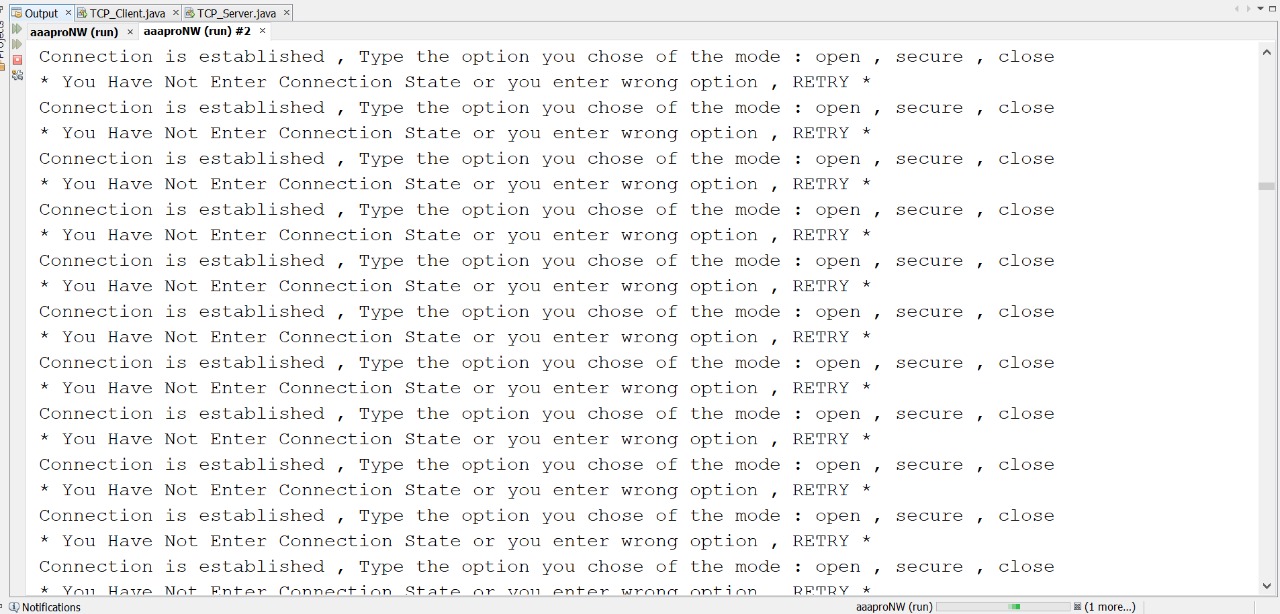
****

**After :**

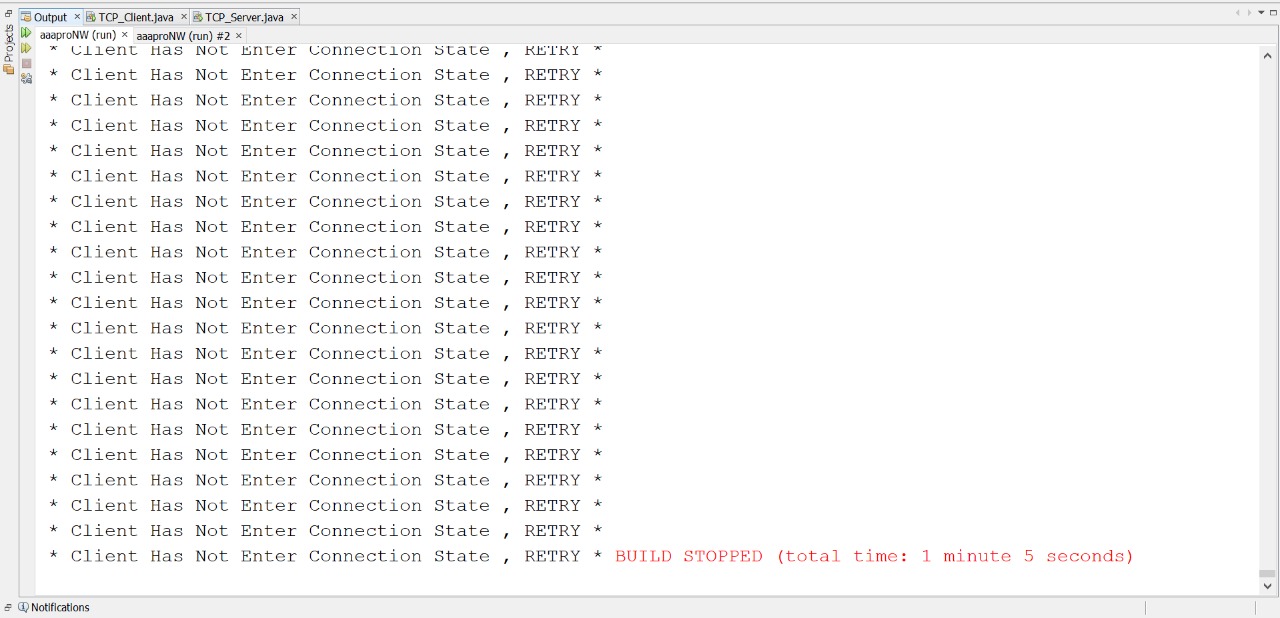


**Problem 3 :** enfinity output ,and cannot read the connection state

**output TCP\_Client**

****

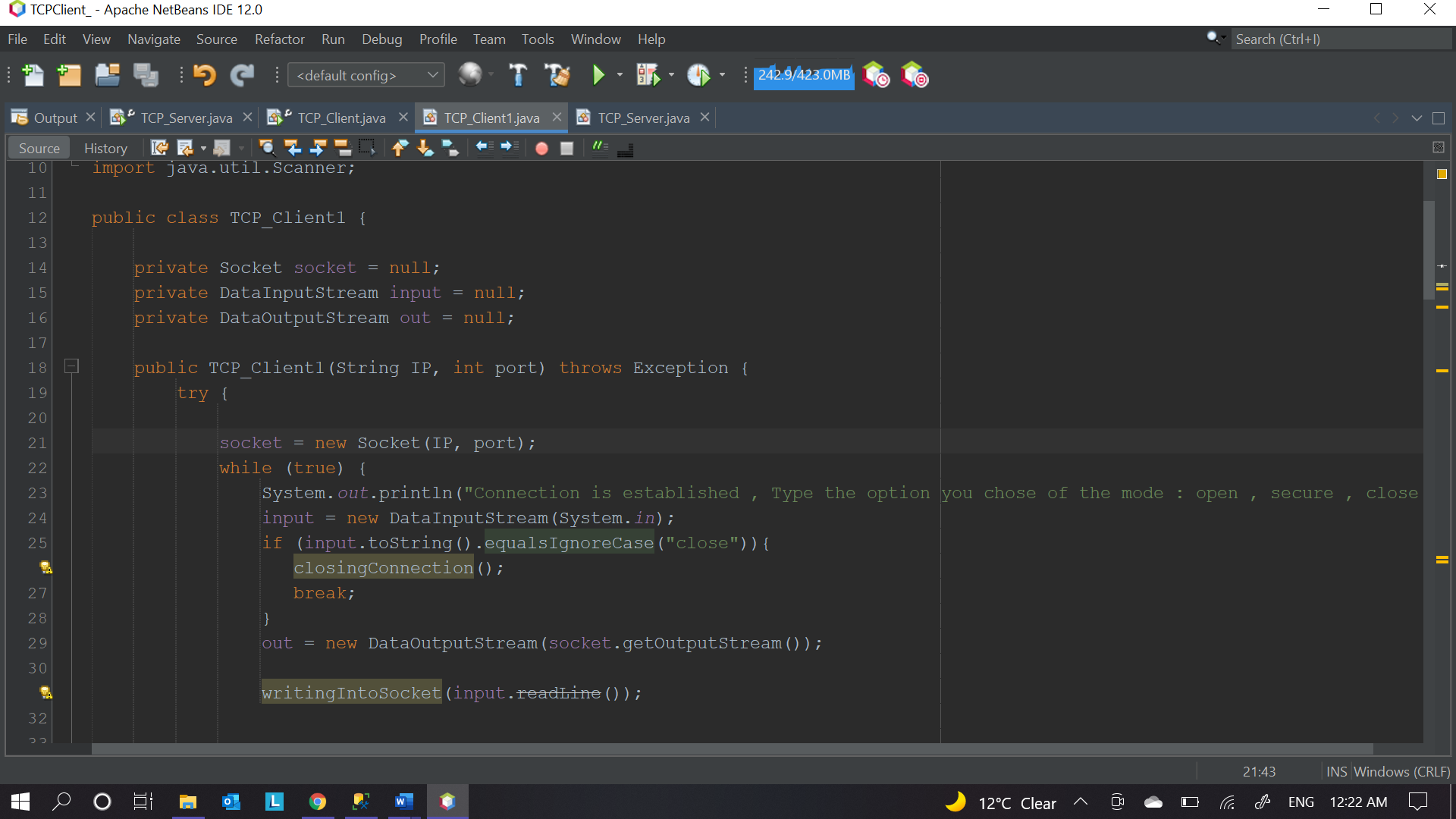
**output TCP\_Server**



**Solution :** we were using input.toString() to read the client massage in terminal

so , we replace it with input.readLine() .

**Before :**

****

**After :**

****

**7- References :**

[1] <https://docs.oracle.com/javase/tutorial/networking/sockets/clientServer.html> .

[2] <https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html>

[3] <https://www.geeksforgeeks.org/socket-programming-in-java/> .

[4] <https://www.youtube.com/watch?v=3IAv4GJkGxc> .

[5] <https://github.com/karanheart96/TCP_Pinger> .