



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

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 .

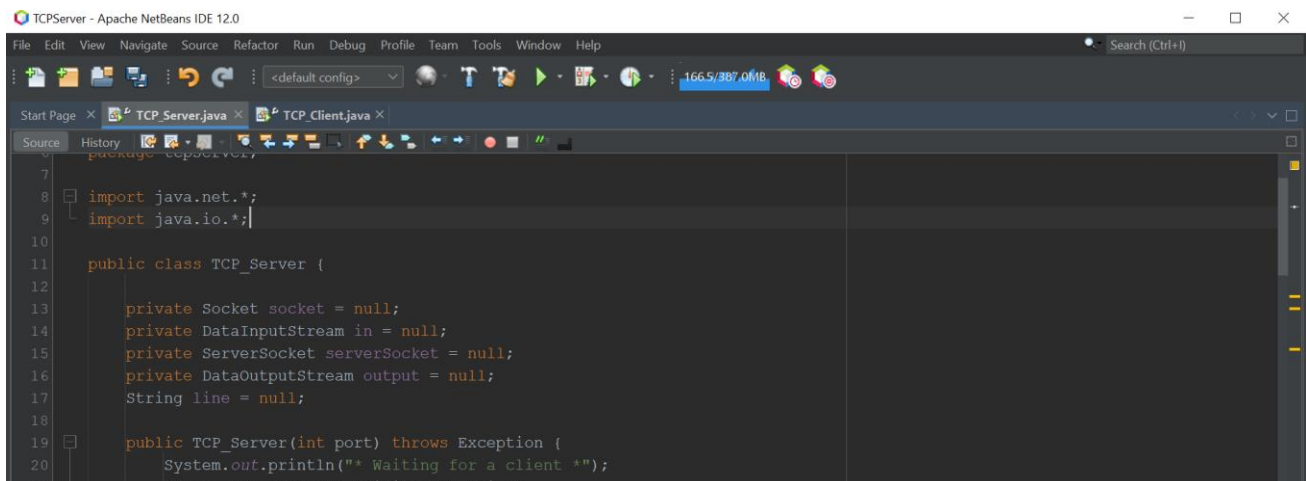
- JDK installation :

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

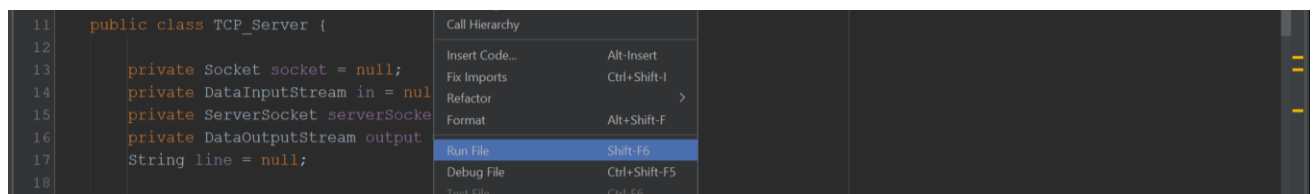
```
C:\Users\soso->java -version
openjdk version "1.8.0_302"
OpenJDK Runtime Environment (Temurin)(build 1.8.0_302-b08)
OpenJDK 64-Bit Server VM (Temurin)(build 25.302-b08, mixed mode)
```

- 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 message 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 message 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 message that client write by :

```
input = new DataInputStream(System.in);
```

```
String option = input.readLine();
```

then will send the message through the socket using method that have client message 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 message from the server then it will close the connection.

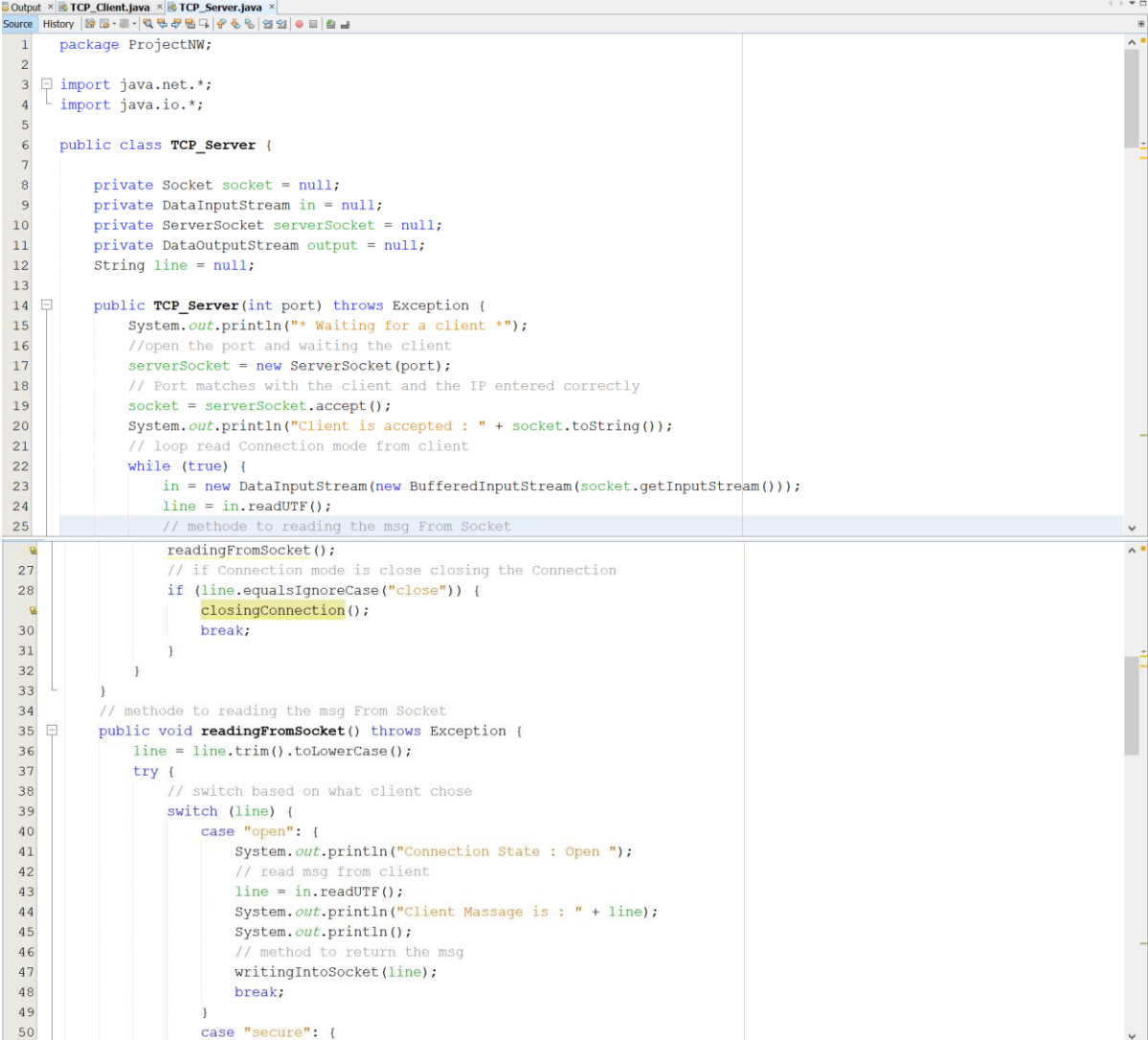
3-Steps for setting up the network :

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-lf1BjxB-fUrC3ziAc

4- Codes and comments:

Code of server side:



```
1 package ProjectNW;
2
3 import java.net.*;
4 import java.io.*;
5
6 public class TCP_Server {
7
8     private Socket socket = null;
9     private DataInputStream in = null;
10    private ServerSocket serverSocket = null;
11    private DataOutputStream output = null;
12    String line = null;
13
14    public TCP_Server(int port) throws Exception {
15        System.out.println("** Waiting for a client **");
16        //open the port and waiting the client
17        serverSocket = new ServerSocket(port);
18        // Port matches with the client and the IP entered correctly
19        socket = serverSocket.accept();
20        System.out.println("Client is accepted : " + socket.toString());
21        // loop read Connection mode from client
22        while (true) {
23            in = new DataInputStream(new BufferedInputStream(socket.getInputStream()));
24            line = in.readUTF();
25            // methode to reading the msg From Socket
26
27            readingFromSocket();
28            // if Connection mode is close closing the Connection
29            if (line.equalsIgnoreCase("close")) {
30                closingConnection();
31                break;
32            }
33        }
34        // methode to reading the msg From Socket
35        public void readingFromSocket() throws Exception {
36            line = line.trim().toLowerCase();
37            try {
38                // switch based on what client chose
39                switch (line) {
40                    case "open": {
41                        System.out.println("Connection State : Open ");
42                        // read msg from client
43                        line = in.readUTF();
44                        System.out.println("Client Message is : " + line);
45                        System.out.println();
46                        // method to return the msg
47                        writingIntoSocket(line);
48                        break;
49                    }
50                    case "secure": {
```

```

51      System.out.println("Connection State : Secure");
52      // read msg and key from client
53      line = in.readUTF();
54      int key = in.readByte();
55      // loop to decrypt Msg shift three
56      String decryptMsg = "";
57      for (int i = 0; i < line.length(); i++) {
58          decryptMsg += (char) (line.charAt(i) - key);
59      }
60      System.out.println("Client Message before encoding : " + line);
61      System.out.println("Client Message after encoding : " + decryptMsg);
62      System.out.println();
63      //loop to shift of msg
64      String encryptMsg = "";
65      for (int i = 0; i < line.length(); i++) {
66          encryptMsg += (char) (decryptMsg.charAt(i) + key);
67      }
68      // method to return the msg
69      writingIntoSocket(encryptMsg);
70      break;
71  }
72
73      case "close": {
74          System.out.println("Connection State : close ");
75          // method to return the msg
76          writingIntoSocket(line);
77          break;
78      }
79
80      default: {
81          System.out.println("** Client Has Not Enter Connection State , RETRY * ");
82      }
83
84  }
85  } catch (Exception i) {
86      System.out.println(i);
87  }
88
89  }
90  // method to return the msg
91  public void writingIntoSocket(String line) throws Exception {
92      //create object to write in socket
93      output = new DataOutputStream(socket.getOutputStream());
94      String msg = line;
95      output.writeBytes(msg + '\n');
96  }
97  // method to close all object that established
98  public void closingConnection() throws Exception {
99      System.out.println("** Server is closing the connection **");
100     in.close();
101     output.close();
102     socket.close();
103
104 }
105
106 public static void main(String[] args) throws Exception {
107     // initialize object server
108     TCP_Server server = new TCP_Server(50000);
109 }
110 }

```

Code for client side:

```
Output x TCP_Client.java x TCP_Server.java x
Source History

1 package ProjectNW;
2
3 import java.net.*;
4 import java.io.*;
5
6 public class TCP_Client {
7
8     Socket socket;
9     DataInputStream input;
10    DataOutputStream out;
11
12    public TCP_Client() throws Exception {
13        try {
14            //initialize ip address and port number for server
15            socket = new Socket("192.168.8.156", 50000);
16            // loop to redisplay connection state message to change the mode
17            while (true) {
18                System.out.println("Connection is established , Type the option you chose of the mode : open , secure , close");
19                //create object to read from client and put in the stream
20                input = new DataInputStream(System.in);
21                String option = input.readLine();
22                //create object to write in socket
23                out = new DataOutputStream(socket.getOutputStream());
24                // method to enable client to send msg through socket
25                writingIntoSocket(option);
26                // if client chose close mode will break loop
27                if (option.equalsIgnoreCase("close")) {
28                    closingConnection();
29                    break;
30                }
31            }
32            // it if client run before server print exception and server is down
33        } catch (Exception i) {
34            System.err.println("Server is Down " + i);
35        }
36    }
37    // method to enable client to send msg through socket
38    public void writingIntoSocket(String line) throws Exception {
39        // send the mode to the server
40        out.writeUTF(line);
41        line = line.trim().toLowerCase();
42        // switch the mode that client chose
43        switch (line) {
44            case "open": {
45                System.out.print("Write your message : ");
46                //read msg from screen
47                line = input.readLine();
48                //send the msg to server
49                out.writeUTF(line);
50                System.out.println("Your message sent successfully to the server. ");
51                // method to read server response
52                readingFromSocket("open",0);
53                break;
54            }
55            case "secure": {
56                System.out.print("Write your message : ");
57                //read msg from screen
58                line = input.readLine();
59                // initialize key for encrypt msg
60                int key = 3;
61                String encryptMsg = "";
62                //loop to shift three letters of msg
63                for (int i = 0; i < line.length(); i++) {
64                    encryptMsg += (char) (line.charAt(i) + key);
65                }
66                //send the msg encrypted to server
67                out.writeUTF(encryptMsg);
68                //send key to server
69                out.writeByte(key);
70                System.out.println("Your message sent successfully to the server. ");
71                // method to read server response
72                readingFromSocket("secure",key);

```

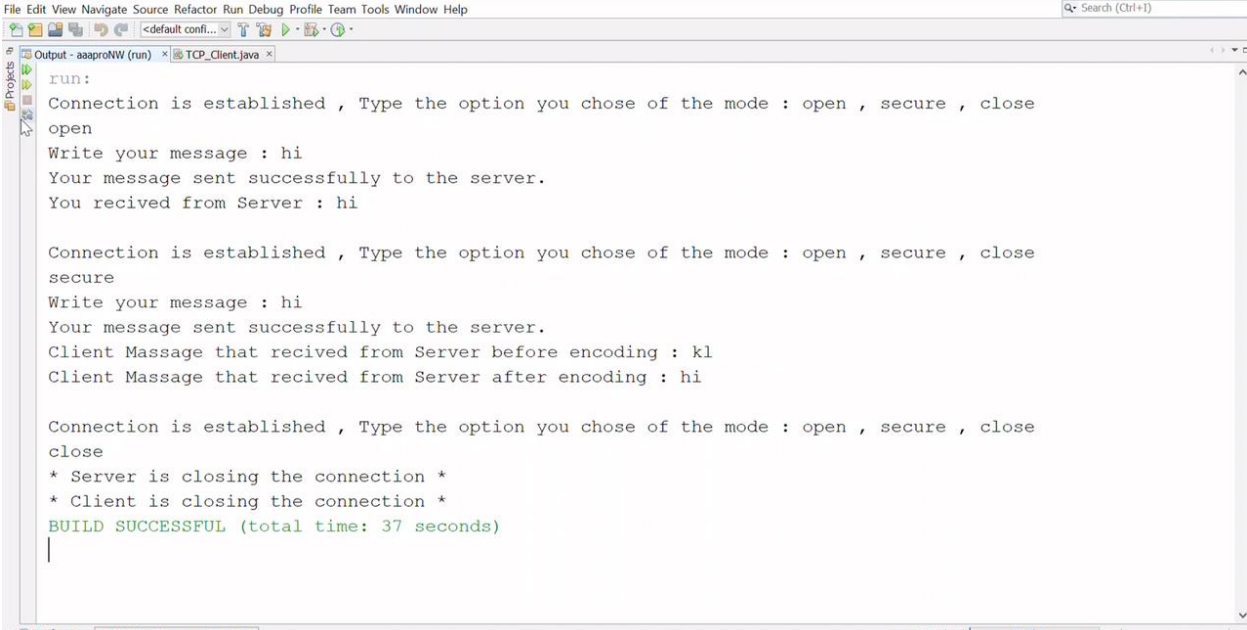
```

73         break;
74     }
75     case "close":{
76         // method to read server response
77         readingFromSocket("close" , 0);
78         break;}
79     default: {
80         System.out.println("** You Have Not Enter Connection State or you enter wrong option , RETRY **");
81     }
82 }
83 }
84 // method to read server response
85 public void readingFromSocket(String line, int key) throws Exception {
86     // read the msg from server
87     input = new DataInputStream(socket.getInputStream());
88     String fromServer = input.readLine();
89     // switch the mode that client passed from parameter
90     switch (line) {
91         case "open": {
92             // read from server what recived from client
93             System.out.println("You recived from Server : " + fromServer);
94             System.out.println();
95         }
96         break;
97         case "secure": {
98             // read from server encrepted msg what recived from client
99             System.out.println("Client Message that recived from Server before encoding : " + fromServer);
100             String decreptMsg = "";
101             // loop to decrept Msg
102             for (int i = 0; i < fromServer.length(); i++) {
103                 decreptMsg += (char) (fromServer.charAt(i) - key);
104             }
105             // after encoding the msg from server
106             System.out.println("Client Message that recived from Server after encoding : " + decreptMsg);
107             System.out.println();
108         }
109         break;
110         case "close": {
111             System.out.println("** Server is closing the connection **");
112             break;
113         }
114     }
115 }
116 // method to close all object that established
117 public void closingConnection() throws Exception {
118     System.out.println("** Client is closing the connection **");
119     input.close();
120     out.close();
121     socket.close();
122 }
123 public static void main(String[] args) throws Exception {
124     // initialize object client
125     TCP_Client Client = new TCP_Client();
126 }
127 }

```


5- Snapshots of the application outputs :

Client Output :

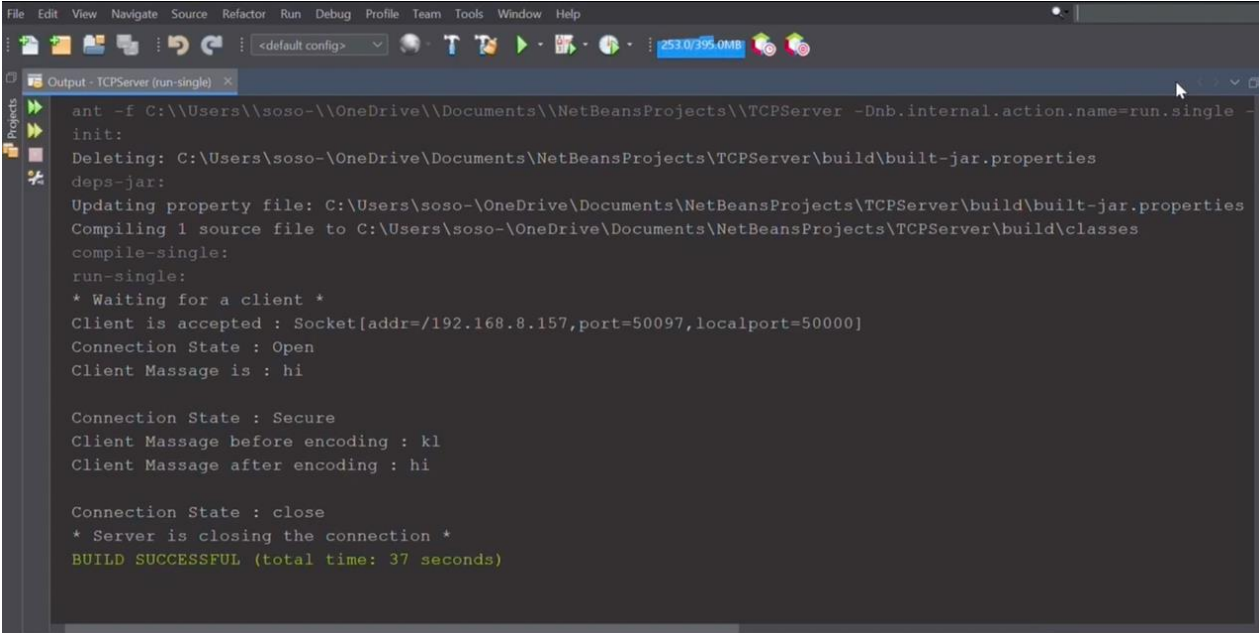


```
run:
Connection is established , Type the option you chose of the mode : open , secure , close
open
Write your message : hi
Your message sent successfully to the server.
You recived from Server : hi

Connection is established , Type the option you chose of the mode : open , secure , close
secure
Write your message : hi
Your message sent successfully to the server.
Client Message that recived from Server before encoding : kl
Client Message that recived from Server after encoding : hi

Connection is established , Type the option you chose of the mode : open , secure , close
close
* Server is closing the connection *
* Client is closing the connection *
BUILD SUCCESSFUL (total time: 37 seconds)
```

Server Output :



```
ant -f C:\Users\soso-\OneDrive\Documents\NetBeansProjects\TCPServer -Dnb.internal.action.name=run.single -
init:
Deleting: C:\Users\soso-\OneDrive\Documents\NetBeansProjects\TCPServer\build\build-jar.properties
deps-jar:
Updating property file: C:\Users\soso-\OneDrive\Documents\NetBeansProjects\TCPServer\build\build-jar.properties
Compiling 1 source file to C:\Users\soso-\OneDrive\Documents\NetBeansProjects\TCPServer\build\classes
compile-single:
run-single:
* Waiting for a client *
Client is accepted : Socket[addr=/192.168.8.157,port=50097,localport=50000]
Connection State : Open
Client Message is : hi

Connection State : Secure
Client Message before encoding : kl
Client Message after encoding : hi

Connection State : close
* Server is closing the connection *
BUILD SUCCESSFUL (total time: 37 seconds)
```

6- Problems and solutions :

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

output TCP_Server

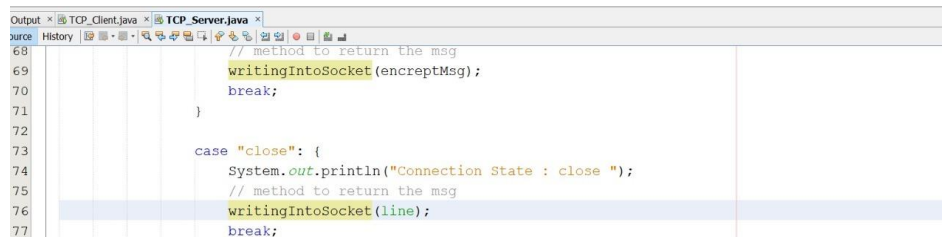
```
run:
* Waiting for a client *
Client is accepted : Socket[addr=/127.0.0.1,port=50041,localport=50000]
Connection State : close
* Server is closing the connection *
Exception in thread "main" java.lang.NullPointerException
    at aaapronw.TCP_Server.closingConnection(TCP_Server.java:101)
    at aaapronw.TCP_Server.<init>(TCP_Server.java:29)
    at aaapronw.TCP_Server.main(TCP_Server.java:108)
C:\Users\rmmr\AppData\Local\NetBeans\Cache\8.2\executor-snippets\run.xml:53: Java returned: 1
BUILD FAILED (total time: 5 seconds)
```

output TCP_Client

```
run:
Connection is established , Type the option you chose of the mode : open , secure , close
close
* Server is closing the connection *
* Client is closing the connection *
BUILD SUCCESSFUL (total time: 3 seconds)
```

Solution:

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



```
Output x TCP_Client.java x TCP_Server.java x
Source History
68 // method to return the msg
69 writingIntoSocket(encryptMsg);
70 break;
71 }
72
73 case "close": {
74     System.out.println("Connection State : close ");
75     // method to return the msg
76     writingIntoSocket(line);
77     break;
```

Problem 2: Not closing the connection successfully : output TCP_Client

```
Client Message before encoding : h
Client Message that received from Server after encoding : h

Connection is established , Type the option you chose of the mode : open , secure , close
close
* Client is closing the connection *
Server is Down
Connection is established , Type the option you chose of the mode : open , secure , close
BUILD SUCCESSFUL (total time: 15 seconds)
```

output TCP_Server

```
Client Message before encoding : h
Client Message after encoding : h

Client is accepted : Socket[addr=/127.0.0.1,port=56452,localport=50000]
* Server is closing the connection *
Client is accepted : Socket[addr=/127.0.0.1,port=56452,localport=50000]
Exception in thread "main" java.net.SocketException: Socket is closed
    at java.base/java.net.Socket.getInputStream(Socket.java:943)
    at tcpclient_TCP_Server.<init>(TCP_Server.java:32)
    at tcpclient_TCP_Server.main(TCP_Server.java:114)
C:\Users\soso\OneDrive\Documents\NetBeansProjects\TCPClient\nbproject\build-impl.xml:1341: The following error occurred while executing this line:
C:\Users\soso\OneDrive\Documents\NetBeansProjects\TCPClient\nbproject\build-impl.xml:936: Java returned: 1
BUILD FAILED (total time: 17 seconds)
```

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

Before :

```
22 while (true) {
23     System.out.println("Connection is established , Type the option you chose of the mode : open , secure , close");
24     input = new DataInputStream(System.in);
25     if (input.toString().equalsIgnoreCase("close")) {
26         closingConnection();
27         break;
28     }
29     out = new DataOutputStream(socket.getOutputStream());
30     writingIntoSocket(input.readLine());
31 }
```

After :

```
26 String option = input.readLine();
27 //create object to write in socket
28 out = new DataOutputStream(socket.getOutputStream());
29 // method to enable client to send msg through socket
30 writingIntoSocket(option);
31 // if client chose close mode will break loop
32 if (option.equalsIgnoreCase("close")) {
33     closingConnection();
34     break;
35 }
```

output TCP_Client

1 Notifications aaaproNW (run) (1 more...)

output TCP_Server

🔔 **Notifications**

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 .