Shubham Chemate
Roll. No. 31118

PICT, PUNE

Assignment 07.

Title: Calculator using TCP.

Problem Statement:

write a TCP socket program for wired network for performing operations on calculator. Demonstrate the packet captured using wireshark Packet Analyzer Tool for P2P mode

Prerequisite:

Transport layer: Roles & Protocols (TCP/UDP)
Socket programming
Wireshark Tool

Learning objective:

students will be able to
- Understand socket programming
- Design networking applications using TCP protocol

S/w & H/w requirements:

- Windows10 64-bit
- wireshark Packet analyzer tool
- 8GB RAM, 512GB SSD, Intel i5 8 Gen. 4-core CPU.

Theory

Client-Server Model

Network applications can be divided in client & server with a communication link joining these 2 bodies.

$$\begin{tikzcd}\boxed{\text{client}} \arrow[r, shift left, "\text{request message}"] & \boxed{\text{server}} \arrow[l, shift left, "\text{response message}"]\end{tikzcd}$$

Normally, from client-side it is one-one connection from server side it is many one connection

The standard model for network applications is client-server model.

A server is a process that is waiting to be connected by client process so that server can do something for the client. Typical BSD sockets applications consists of two separate application level processes, client request a connection & server accept it.

Client server Using TCP:

TCP clients send requests to server & server responds with acknowledgement. Every time client communicates with server & receive response from it.

Algorithm:

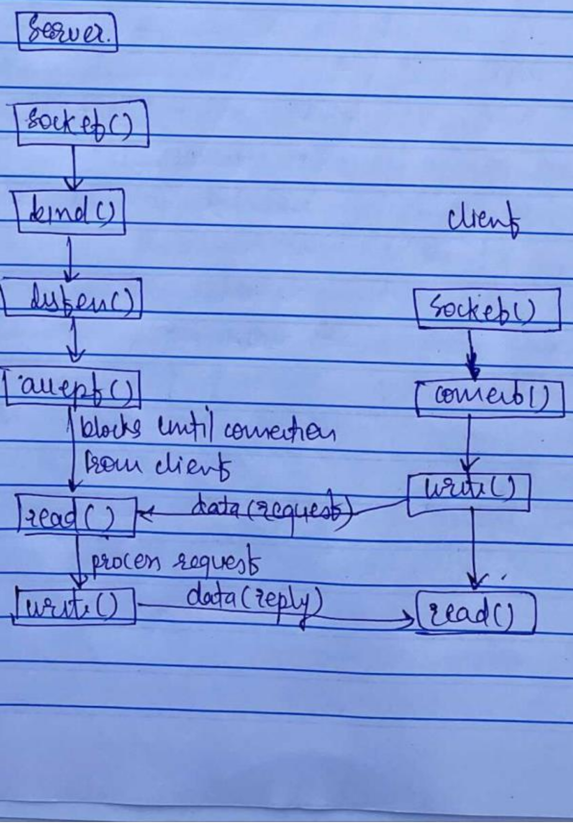
server:

```
i) Create server socket & bind it to the port.
ii) Listen for new connection & when connection arrives, accept it
iii) Read clients message & display it.
iv) Get a message from user & send it to client
```

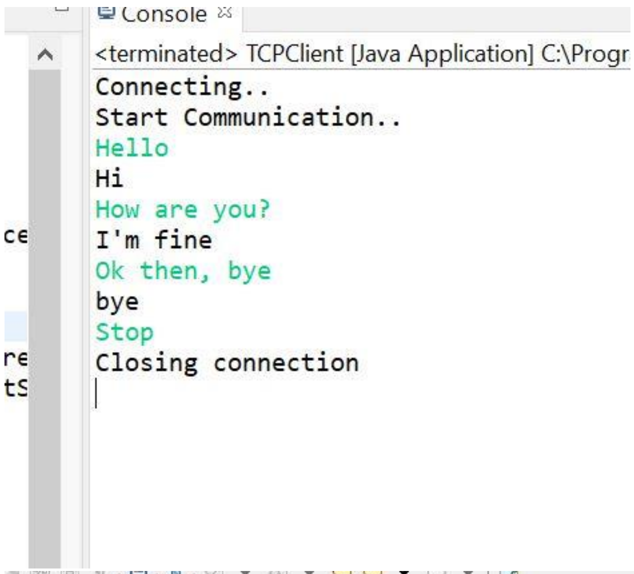
v> close the server socket
vi> stop.

Client.
i> Create a client socket & connect it to the server's port number
ii> Get a message from user & send it to server.
iii> Read server's responce & display it.
iv> close the client socket.
v> stop

Socket functions for TCP client/server in connection oriented scenario-

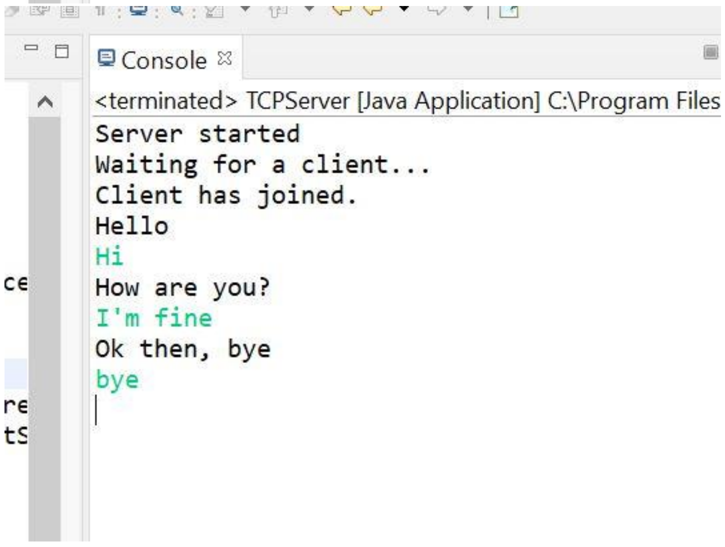


Conclusion: Sucessfully implemented calculator operations using TCP socket programming.



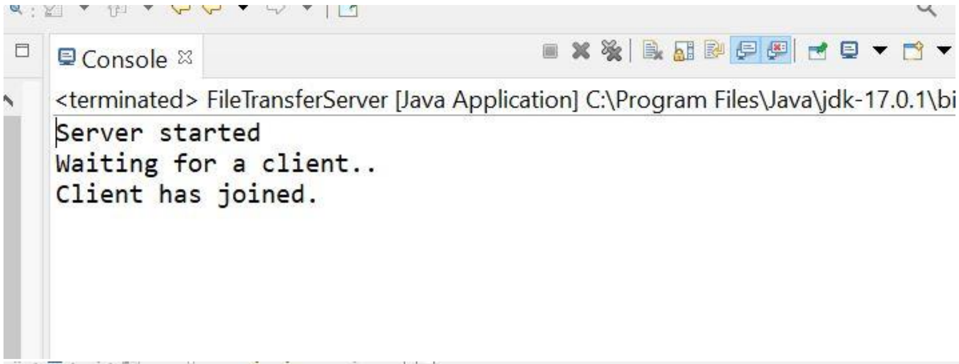
<terminated> TCPClient [Java Application] C:\Progr

```
Connecting..
Start Communication..
Hello
Hi
How are you?
I'm fine
Ok then, bye
bye
Stop
Closing connection
```

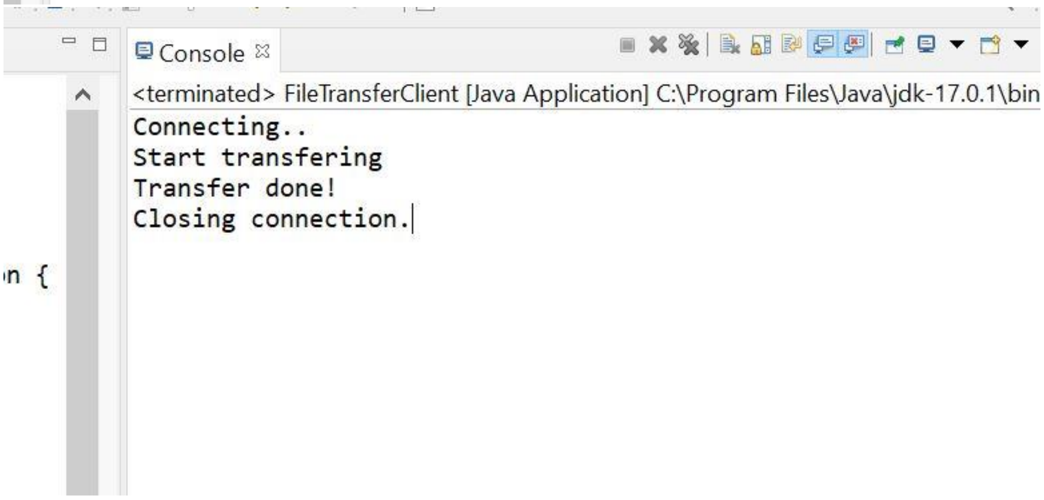


<terminated> TCPServer [Java Application] C:\Program Files

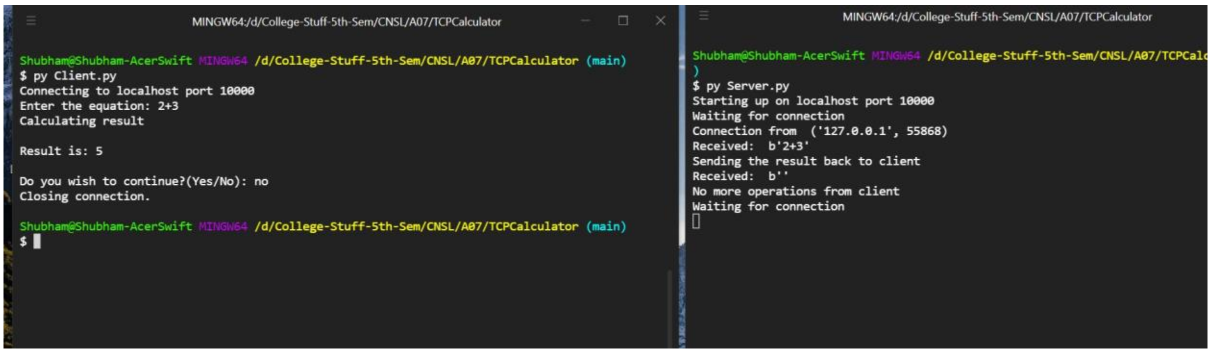
```
Server started
Waiting for a client...
Client has joined.
Hello
Hi
How are you?
I'm fine
Ok then, bye
bye
```



```
<terminated> FileTransferServer [Java Application] C:\Program Files\Java\jdk-17.0.1\bi
Server started
Waiting for a client..
Client has joined.
```



```
<terminated> FileTransferClient [Java Application] C:\Program Files\Java\jdk-17.0.1\bin
Connecting..
Start transfering
Transfer done!
Closing connection.
```


MINGW64:/d/College-Stuff-5th-Sem/CNSL/A07/TCPCalculator
MINGW64:/d/College-Stuff-5th-Sem/CNSL/A07/TCPCalculator

```
Shubham@Shubham-AcerSwift MINGW64 /d/College-Stuff-5th-Sem/CNSL/A07/TCPCalculator (main)
$ py Client.py
Connecting to localhost port 10000
Enter the equation: 2+3
Calculating result

Result is: 5

Do you wish to continue?(Yes/No): no
Closing connection.

Shubham@Shubham-AcerSwift MINGW64 /d/College-Stuff-5th-Sem/CNSL/A07/TCPCalculator (main)
$
```

```
Shubham@Shubham-AcerSwift MINGW64 /d/College-Stuff-5th-Sem/CNSL/A07/TCPCalc
)
$ py Server.py
Starting up on localhost port 10000
Waiting for connection
Connection from  ('127.0.0.1', 55868)
Received:  b'2+3'
Sending the result back to client
Received:  b''
No more operations from client
Waiting for connection
```