Name : Bramha Nimbalkar

Roll no : 7

Srn : 202100381

ASSIGMENT 7

Socket Programming for TCP Client and TCP Server.

Client

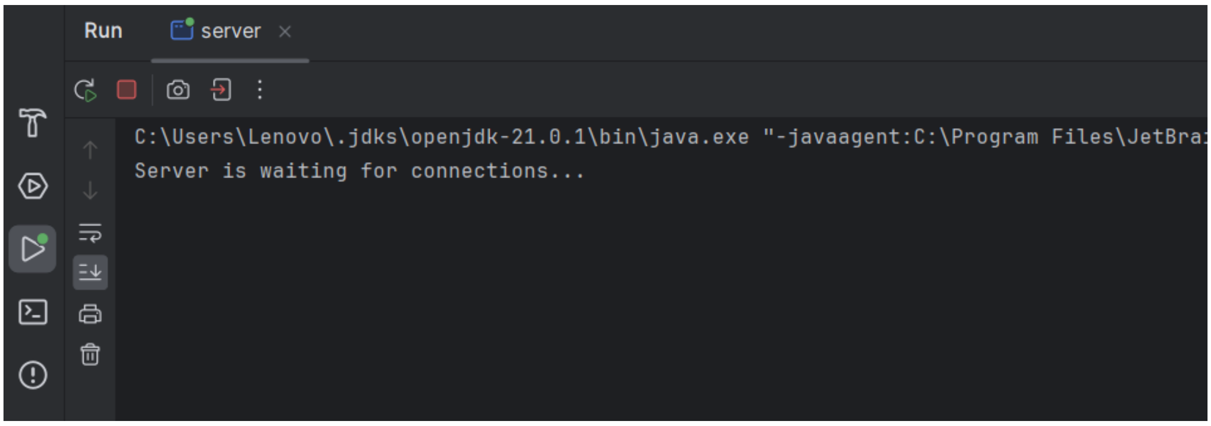
```
import java.io.*;
import java.net.*;

public class client {
    public static void main(String[] args) {
        try {

            Socket clientSocket = new Socket("localhost", 4000);


            InputStream input = clientSocket.getInputStream();
            OutputStream output = clientSocket.getOutputStream();


            BufferedReader reader = new BufferedReader(new
InputStreamReader(input));
            PrintWriter writer = new PrintWriter(output, true);


            writer.println("Hello, server! This is the client.");
            writer.println("Hello, server! This is the Bramha.");
            writer.println("Hello, server! This is the friends .");

            String serverMessage = reader.readLine();
            String serverMessage1 = reader.readLine();
            String serverMessage2 = reader.readLine();

            System.out.println("Server: " + serverMessage);
            System.out.println("Server: " + serverMessage1);
            System.out.println("Server: " + serverMessage2);


            clientSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Server

```
import java.io.*;
import java.net.*;


public class server {
    static String clientMessage = null;
    static String clientMessage1 = null;
    static String clientMessage2 = null;
    public static void main(String[] args) {
        try {

            ServerSocket serverSocket = new ServerSocket(4000);
            System.out.println("Server is waiting for connections...");


            Socket clientSocket = serverSocket.accept();
            System.out.println("Client connected!");


            InputStream input = clientSocket.getInputStream();
            OutputStream output = clientSocket.getOutputStream();


            BufferedReader reader = new BufferedReader(new
InputStreamReader(input));
            PrintWriter writer = new PrintWriter(output, true);


            clientMessage = reader.readLine();
            clientMessage1 = reader.readLine();
            clientMessage2 = reader.readLine();
            System.out.println("Client: " + clientMessage);
            System.out.println("Client: " + clientMessage1);
            System.out.println("Client: " + clientMessage2);

            if(clientMessage!=null){
                writer.println("ACK");
            }

            if (clientMessage1!=null){
                writer.println("ACK");
            }

             if (clientMessage2!=null){
                 writer.println("ACK");
            }


            clientSocket.close();
            serverSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}


//    UDP (User Datagram Protocol) communication, on the other hand, uses
DatagramSocket and DatagramPacket
//    for sending and receiving discrete, connectionless datagrams, and it
doesn't involve establishing
//    a continuous connection as TCP does.
```

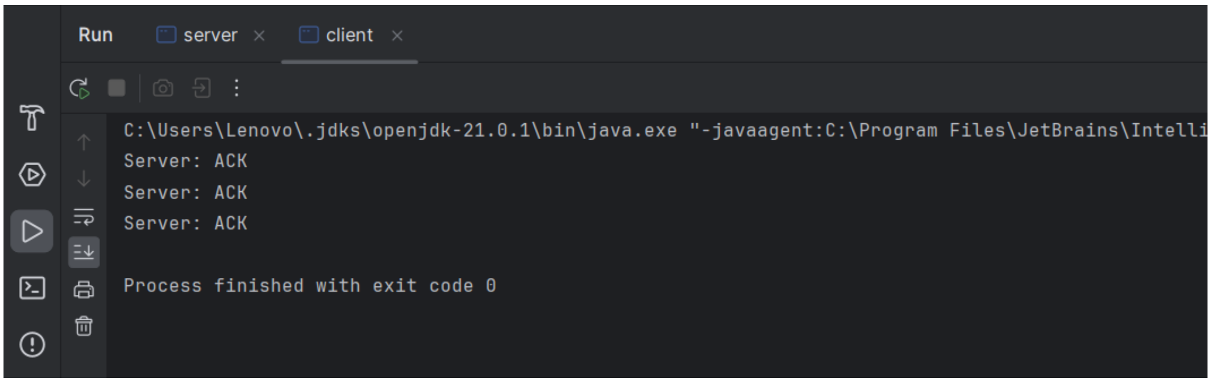
Output

Server



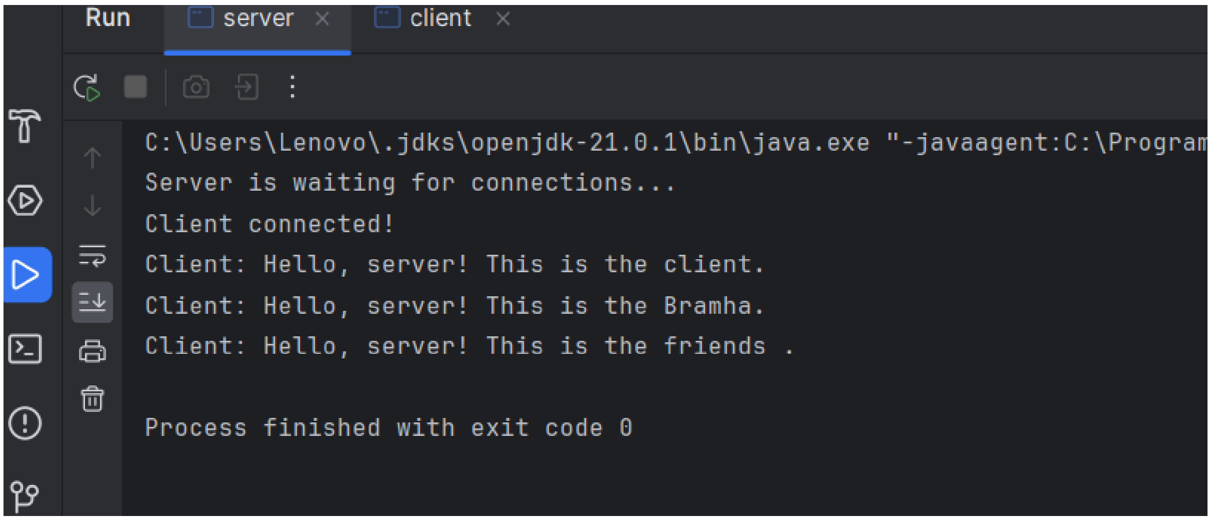
```
C:\Users\Lenovo\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBra
Server is waiting for connections...
```

Client



```
C:\Users\Lenovo\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\Intelli
Server: ACK
Server: ACK
Server: ACK

Process finished with exit code 0
```

Server



```
C:\Users\Lenovo\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:C:\Program
Server is waiting for connections...
Client connected!
Client: Hello, server! This is the client.
Client: Hello, server! This is the Bramha.
Client: Hello, server! This is the friends .

Process finished with exit code 0
```