ASSIGNMENT#1 REPORT

CS-3001 COMPUTER NETWORKING

Group Members:

Ayaz Hasan [20k-1044]

Ferdeen Bakht [20k-0219]

Section :

SE-5A

Submitted to:

Prof. Dr. Sufian Hameed

FAST-NUCES Karachi

SOCKET PROGRAMMING

Socket programming with TCP (Transmission Control Protocol) is a method of establishing communication between two devices over a network. In TCP, data is sent as a stream of bytes, with the protocol providing a reliable, ordered and error-checked delivery of data. Here is a brief overview of how to use socket programming with TCP in Java:

1- Create a ServerSocket object on the server side, and bind it to a specific port.

```
// Create a new server socket and start listening on the specified port
serverSocket = new ServerSocket(port);
statusLabel.setText("Listening on port " + port);
```

2- On the client side, create a Socket object, and connect it to the server's IP address and port number.

```
socket = new Socket(ip, port);
```

3-On the server side, use the accept() method of the ServerSocket object to wait for an incoming connection from the client. This method blocks the program until a connection is made.

```
public void run() {
    try {
        // Accept a new client connection
        clientSocket = serverSocket.accept();
```

4-On the client side, use the getInputStream() method of the Socket object to get an InputStream, which can be used to receive data from the server. Use the getOutputStream() method to get an OutputStream, which can be used to send data to the server.

```
in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
out = new PrintWriter(socket.getOutputStream(), true);
```

5-On the server side, use the getInputStream() method of the Socket object returned by the accept() method to receive data from the client. Use the getOutputStream() method of the same Socket object to send data to the client.The ConnectionThread reads input from the client through the BufferedReader and writes output back to the client through the PrintWriter. The input reading is done in an infinite loop to continuously listen for input from the client.

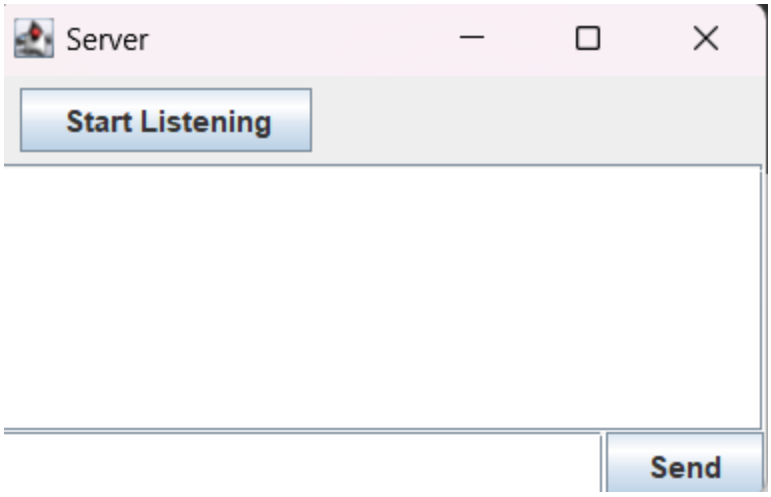
```
private class ConnectionThread implements Runnable {
    public void run() {
        try {
            // Accept a new client connection
            clientSocket = serverSocket.accept();
            // Create input and output streams for the client connection

            in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
            out = new PrintWriter(clientSocket.getOutputStream(), true);
            statusLabel.setText("Connected to client");

            // Listen for messages from the client and display them in the chat box
            while (true) {
                String message = in.readLine();
                if (message == null) {
                    break;
```

6-Once communication is complete, close the Socket objects on both the client and server side

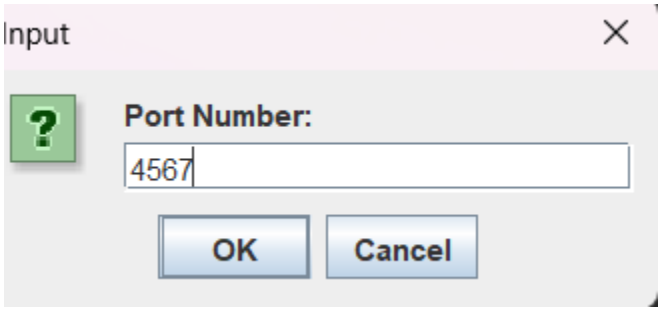
```
}
finally {
  // Close the input and output streams and the client socket
  try {
      in.close();
      out.close();
      clientSocket.close();
      isListening = false;
      startButton.setText("Start listening");
      statusLabel.setText("Not listening");
  } catch (IOException ex) {
      ex.printStackTrace();
  }
```

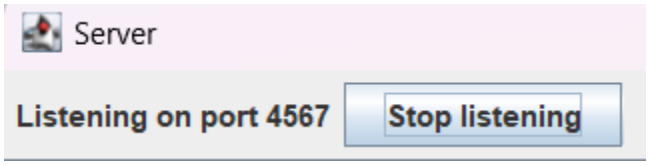
DEMO

We use java swing to create a simple GUI for the Client Server Application.The purpose of gui is to make it easier for the user to understand the flow of the work .

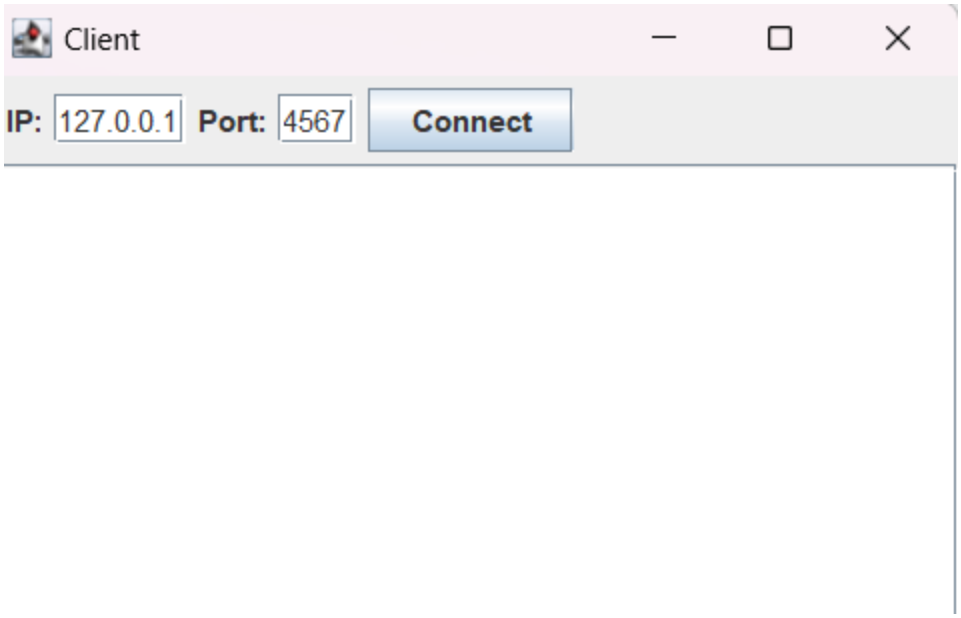


First click on the Start listening button the it will ask for the port number and until it is not connected to Client the below message is shows



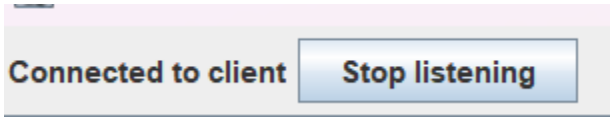


Now run the client side And it will ask for ip address and port number

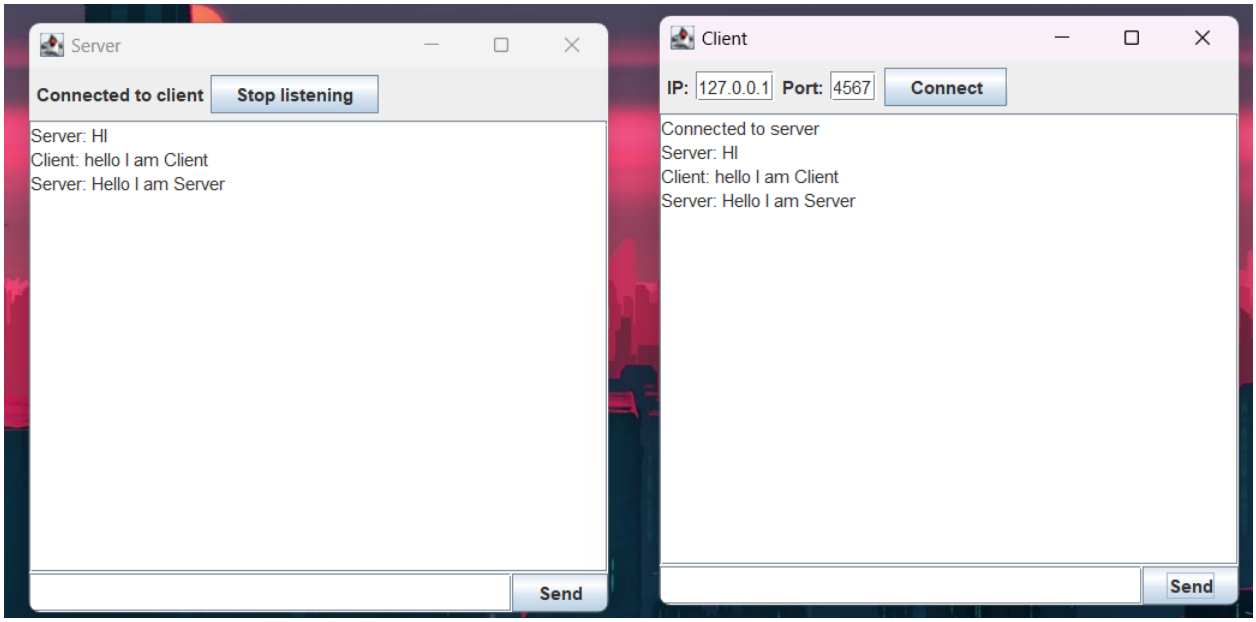


If user enter wrong port number it will not connected and show error message

As soon as user input port number ,on the server side connection message shown as below



Now they can communicate with each other.


Server
Connected to client
Stop listening
Server: HI
Client: hello I am Client
Server: Hello I am Server
Send
Client
IP: 127.0.0.1
Port: 4567
Connect
Connected to server
Server: HI
Client: hello I am Client
Server: Hello I am Server
Send