**TCP echo server**

import java.io.\*;

import java.net.\*;

public class TcpEchoServer

{

private final static int PORT = 8080;

public static void main(String[] args) throws IOException

{

ServerSocket serverSocket = new ServerSocket(PORT);

System.out.println("Listening on port " + PORT);

while (true) {

try (Socket socket = serverSocket.accept();

InputStreamReader isr = new InputStreamReader(socket.getInputStream());

BufferedReader in = new BufferedReader(isr);

PrintWriter out = new PrintWriter(socket.getOutputStream(), true)

) {

System.out.println("Connection accepted");

String line;

while ((line = in.readLine()) != null) {

System.out.println("Server received: " + line + ". Sending to client");

out.println(line);

if (line.equals("Bye")) {

break;

}

}

}

}

}

}

**TCP echo Client**

import java.io.\*;

import java.net.\*;

public class TcpEchoClient

{

private final static String HOSTNAME = "localhost";

private final static int PORT = 8080;

public static void main(String[] args) throws IOException

{

try (Socket clientSocket = new Socket(HOSTNAME, PORT);

InputStreamReader isr = new InputStreamReader(clientSocket.getInputStream());

BufferedReader in = new BufferedReader(isr);

PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true)

) {

System.out.println("Connected to " + HOSTNAME + " on port " + PORT);

String data = "Hello\nBye";

System.out.println("Sending to server:\n" + data);

out.println(data);

String line;

while ((line = in.readLine()) != null) {

System.out.println("Client received: " + line);

}

}

}

}

**Server Output**

Listening on port 8080

Connection accepted

Server received: Hello. Sending to client

Server received: Bye. Sending to client

**Client output**

Connected to localhost on port 8080

Sending to server:

Hello

Bye

Client received: Hello

Client received: Bye

**Create Single-Threaded TCP Echo Server and Client using Java**

**TCP** is a communication protocol that allows exchange data between devices in a network. TCP is a connection-oriented protocol. This means that communication between devices is reliable and guarantees that all data will be properly send and received. An **echo server** is a server that receives data from a client and sends back an identical copy of the data to a client.

This tutorial shows how to create a single-threaded TCP echo server and client using Java. A single-threaded server means that it accepts only one client connection at a time.

TCP echo server is implemented in the TcpEchoServer class. In the main method, an instance of the [ServerSocket](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/net/ServerSocket.html) class is created to listen connections on specified port.

We use an infinite while loop to accept connection from a client infinity times. The [accept](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/net/ServerSocket.html#accept()) method is invoked, which blocks code execution until it receives a connection from a client. When connection is established, the [accept](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/net/ServerSocket.html#accept()) method returns an instance of a [Socket](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/net/Socket.html) class. We get the input and output streams of a client socket. The try-with resources statement is used to ensure that opened resources will be automatically closed at the end of the statement.

The [readLine](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/io/BufferedReader.html#readLine()) method is used to read received data from the client line by line. Data is printed to the standard output and sent back to the client. Reading process will be finished when "Bye" message is received.