TCP Sequential Client/Server

```
public class EchoServer {

    public static void main(String[] args) throws Exception {

        ServerSocket ss = new ServerSocket(6789);

        Socket client = ss.accept();

        BufferedReader fromClient =
            new BufferedReader(
                new InputStreamReader(
                    client.getInputStream()));

        DataOutputStream toClient =
            new DataOutputStream(
                client.getOutputStream());

        String msg = fromClient.readLine();

        msg = msg.toUpperCase();

        toClient.writeBytes(msg + "\n");

        toClient.flush();

        client.close();
        ss.close();
    }
}
```

```
public class EchoClient {

    public static void main(String[] args) throws Exception {

        Socket cs = new Socket("localhost", 6789);

        BufferedReader UserInput =
            new BufferedReader(
                new InputStreamReader(System.in));

        BufferedReader fromServer =
            new BufferedReader(
                new InputStreamReader(
                    cs.getInputStream()));

        DataOutputStream toServer =
            new DataOutputStream(
                cs.getOutputStream());

        System.out.print("Enter a word:");

        String text = UserInput.readLine();

        toServer.writeBytes(text + "\n");
        toServer.flush();

        String res = fromServer.readLine();

        System.out.println("From Server:" + res);

        cs.close();
    }
}
```

TCP Parallel Echo Server (Multi-Threaded)

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

public class EchoServer {

    public static void main(String[] args) throws Exception {

        ServerSocket ss = new ServerSocket(6789);

        while (true) {
            Socket client = ss.accept();

            ClientHandler t = new ClientHandler(client);
            t.start();
        }
    }
}
```

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

public class ClientHandler extends Thread {

    Socket client;

    public ClientHandler(Socket client) {
        this.client = client;
    }

    public void run() {
        try {

            BufferedReader fromClient =
                new BufferedReader(
                    new InputStreamReader(
                        client.getInputStream()));

            DataOutputStream toClient =
                new DataOutputStream(
                    client.getOutputStream());

            String msg = fromClient.readLine();

            msg = msg.toUpperCase();

            toClient.writeBytes(msg + "\n");
            toClient.flush();

            client.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

UDP Sequential Client/Server

```
public class EchoServer {

    public static void main(String[] args) throws Exception {

        DatagramSocket ss = new DatagramSocket(6789);

        byte[] receiveData = new byte[1024];

        while (true) {

            DatagramPacket fromClient = new DatagramPacket(receiveData, receiveData.length);

            ss.receive(fromClient);

            String msg = new String(fromClient.getData(),0,fromClient.getLength());

            System.out.println("Received: " + msg);

            String reversed = new StringBuilder(msg).reverse().toString();

            byte[] sendData = reversed.getBytes();

            InetAddress clientIP = fromClient.getAddress();
            int clientPort = fromClient.getPort();

            DatagramPacket sendPacket =
                    new DatagramPacket(sendData, sendData.length, clientIP, clientPort);

            ss.send(sendPacket);
        }
    }
}
```

```
public class EchoClient {

    public static void main(String[] args) throws Exception {

        BufferedReader UserInput = new BufferedReader(new InputStreamReader(System.in));

        DatagramSocket clientSocket = new DatagramSocket();

        InetAddress IPAddress = InetAddress.getByName("localhost");

        System.out.print("Enter a string: ");
        String sentence = UserInput.readLine();

        byte[] sendData = sentence.getBytes();

        DatagramPacket sendPacket =
                new DatagramPacket(sendData, sendData.length, IPAddress, 6789);

        clientSocket.send(sendPacket);

        byte[] receiveData = new byte[1024];

        DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);

        clientSocket.receive(receivePacket);

        String result = new String(receivePacket.getData(),0,receivePacket.getLength());

        System.out.println("Result from server: " + result);

        clientSocket.close();
    }
}
```

UDP Parallel Client/Server

```
public class EchoServer {

    public static void main(String[] args) throws Exception {

        DatagramSocket ss = new DatagramSocket(6789);

        while (true) {

            byte[] receiveData = new byte[1024];

            DatagramPacket receivePacket = new DatagramPacket(receiveData,receiveData.length);

            ss.receive(receivePacket);

            Helper handler = new Helper( ss, receivePacket);

            handler.start();
        }
    }
}
```

```
public class Helper extends Thread {

    private DatagramSocket socket;
    private DatagramPacket packet;

    public Helper(
            DatagramSocket socket,
            DatagramPacket packet) {

        this.socket = socket;
        this.packet = packet;
    }

    @Override
    public void run() {

        try {

            String message = new String(packet.getData(),0,packet.getLength());

            String upperCase = message.toUpperCase();

            byte[] responseBytes = upperCase.getBytes();

            DatagramPacket reply =
                    new DatagramPacket(
                            responseBytes,
                            responseBytes.length,
                            packet.getAddress(),
                            packet.getPort());

            socket.send(reply);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```