جامعة الأهرام الكندية
AHRAM CANADIAN UNIVERSITY

Lecture (03)
Network Programming

Dr. Ahmed ElShafee

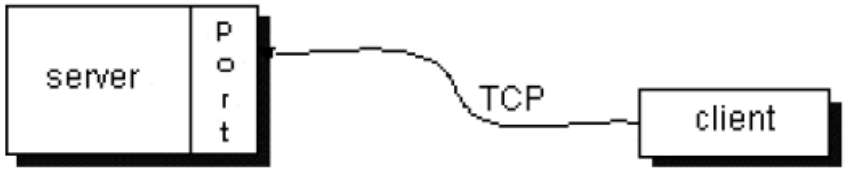
Concepts of socket programming

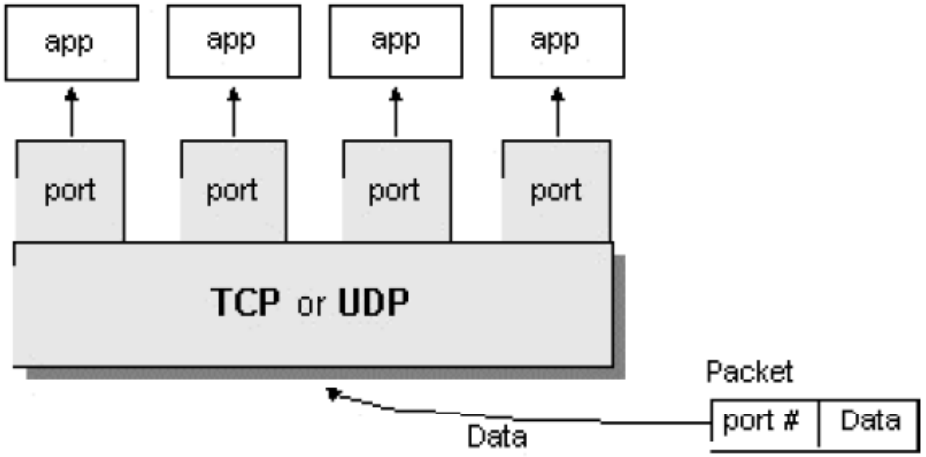
Important concepts needed for writing network program in Java

1) Communication protocols: TCP and UDP
2) Ports and Internet Addresses
3) Sockets
4) Uniform Resource Locator (URL)
5) Uniform Resource Identifier (URI)
6) Streams and Threads
7) Classes in java.net and java.io packages

Ports

- Ports are a virtual channels that enables applications to share the single network channel to exchange data.
- Port numbers are represented by 16-bit numbers. (0 to 65,535) The port numbers ranging from 0 - 1023 reserved for use by well known application





Sockets

You can reach required service via its network and port IDs. what then?

a) If you are a client

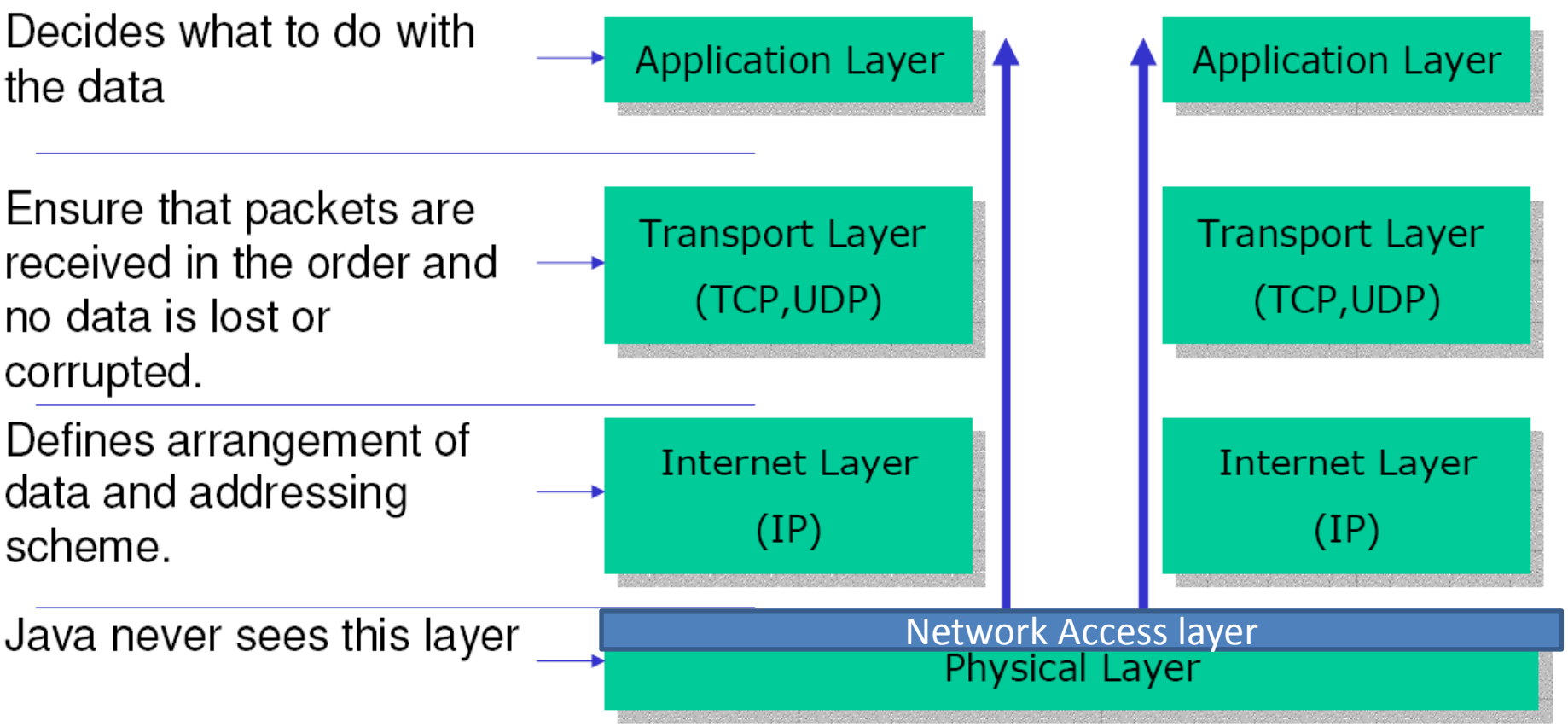
- you need an API that will allow you to send messages to that service and read replies from it

b) If you are a server

- you need to be able to create a port and listen at it.
- you need to be able to read the message comes in and reply to it.

The **Socket** and **ServerSocket** are the Java client and server classes to do this.

Network Layers



TCP and UDP

- Java only supports TCP (Transmission Control Protocol), UDP (User Datagram Protocol) and application layer protocols built on top of these.

Simple client example

```
import java.io.*;
import java.net.*;
public class tcpclient
    {
    public static void main(String[] args) throws IOException
            {
            Socket clientSocket = null;
            clientSocket = new Socket("192.168.1.2",5050);
            Writer w = new OutputStreamWriter(clientocket.getOutputStream(), true);
            InputStreamReader isr = new InputStreamReader
    (clientSocket.getInputStream());
             BufferedReader br = new BufferedReader (isr);
             System.out.print ("Connection established.\n>");
             w.write("Connection Established.\r\n>");
                        }}
```

Email protocols

IMAP (Internet Message Access Protocol) –

Is a standard protocol for accessing e-mail from your local server.

IMAP is a client/server protocol in which e-mail is received and held for you by your Internet server.

As this requires only a small data transfer this works well even over a slow connection such as a modem.

Only if you request to read a specific email message will it be downloaded from the server.

You can also create and manipulate folders or mailboxes on the server, delete messages etc.

Email protocols (2)

The **POP (Post Office Protocol 3)** protocol provides a simple, standardized way for users to access mailboxes and download messages to their computers.

When using the POP protocol all your eMail messages will be downloaded from the mail server to your local computer.

You can choose to leave copies of your eMails on the server as well.

Email protocols (3)

The **SMTP (Simple Mail Transfer Protocol)** protocol is used by the Mail Transfer Agent (MTA) to deliver your eMail to the recipient's mail server.

The SMTP protocol can only be used to send emails, not to receive them.

Depending on your network / ISP settings, you may only be able to use the SMTP protocol under certain conditions

Email protocols (4)

The **HTTP protocol** is not a protocol dedicated for email communications, but it can be used for accessing your mailbox.

Also called web based email, this protocol can be used to compose or retrieve emails from an your account.

Hotmail is a good example of using HTTP as an email protocol.

Email

- SMTP E-mail is sent by socket communication with port 25 on a computer system.
- Open a socket connected to port 25 on some system, and speak “mail protocol” to the daemon at the other end.

Email (2)

Using telnet

telnet 192.168.1.2 25

<<220 192.168.1.2 ESMTP (Code-Crafters Ability Mail Server 2.52)

HELO 192.168.1.2

<<250 192.168.1.2

MAIL FROM: user001@192.168.1.2

<<250 Email address accepted. <user001@192.168.1.2>

RCPT TO: user002@192.168.1.2

<< 250 Email address accepted. <user002@192.168.1.2>

Email (3)

DATA

<<354 Please send the data and end with a <CRLF>.<CRLF>.

SUBJECT: email subject<ret><ret>

Body of the test email

.

<<250 Mail accepted and queued for delivery

quit

Sendemail.java

```
import java.io.*;
import java.net.*;
public class sendemail
    {
    public static void main(String args[]) throws    IOException
            {
            Socket sock;
            DataInputStream dis;
            BufferedReader br;
            PrintStream ps;
            System.out.println(">>> Connect 192.168.1.2");
            sock = new Socket("192.168.1.2", 25);
            br = new BufferedReader (new InputStreamReader(sock.getInputStream()));
```

Sendemail.java (2)

```
ps = new PrintStream( sock.getOutputStream());
System.out.println( br.readLine() );
System.out.println(">>> HELO 192.168.1.2");
ps.println("HELO 192.168.1.2");
System.out.println( br.readLine() );
System.out.println(">>> MAIL FROM: user001@192.168.1.2");
ps.println("MAIL FROM: <user001@192.168.1.2>");
System.out.println( br.readLine() );
String Addressee= "user002@192.168.1.2";
System.out.println(">>> Rcpt to: " + Addressee);
ps.println("RCPT TO: <" + Addressee +">");
System.out.println( br.readLine() );
System.out.println(">>> Send \"data\"");
ps.println("DATA");
```

Sendemail.java (3)

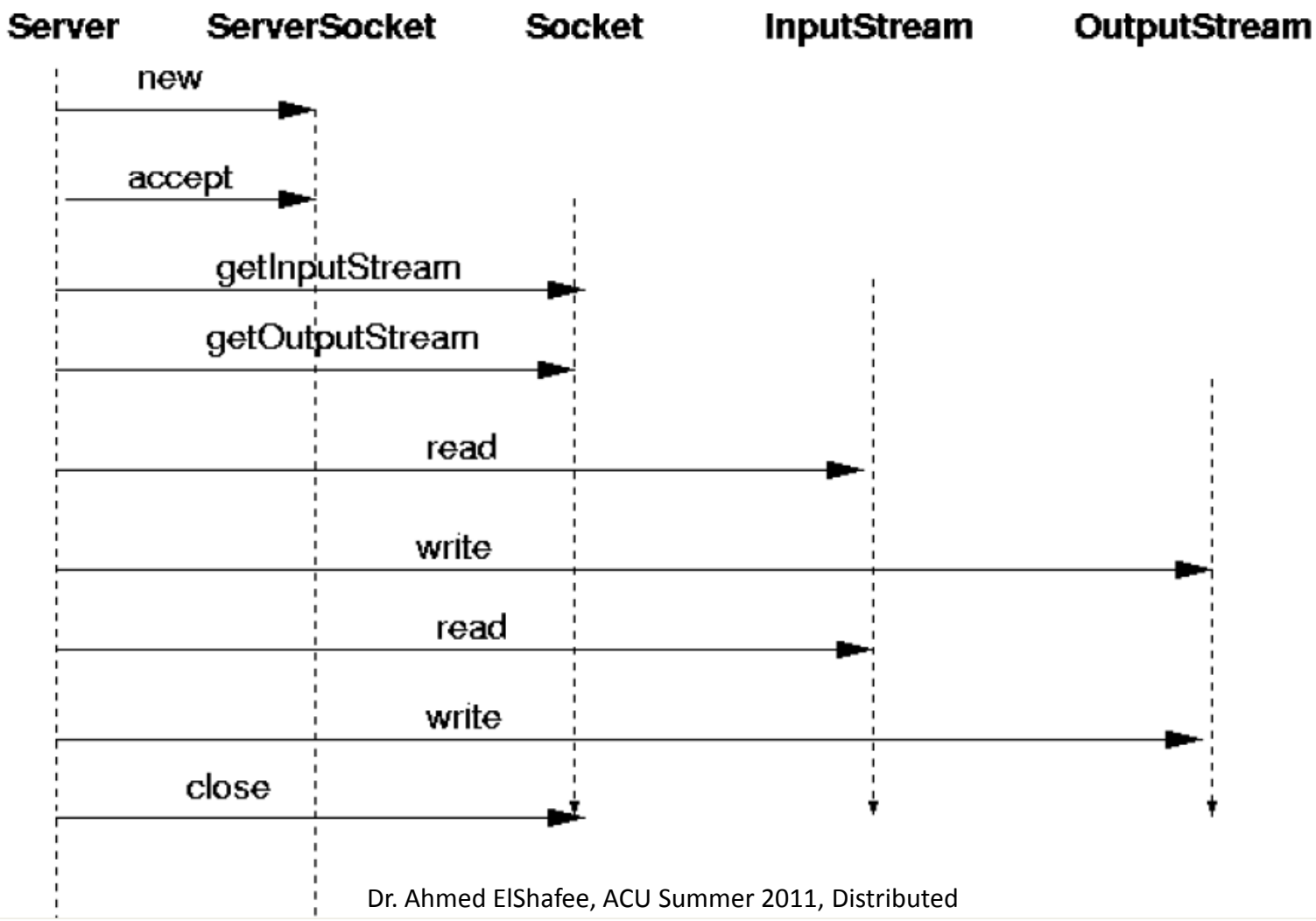
```
System.out.println( br.readLine() );
System.out.println(">>>>>>>>>>");
System.out.println(">>> Subject: test email\n\n");
System.out.println(">>> This is the message\n thatJava sent");
System.out.println(">>> We are testing Socket Programming");
System.out.println(">>> in ACU-CSIT.");
System.out.println(">>>>>>>>>>");
ps.println("SUBJECT:Test email\n");
ps.println("This is the message\n that Java sent");
ps.println("We are testing Socket Programming");
ps.println("in ACU-CSIT Training program.");
System.out.println(">>> .");
ps.println(".");
System.out.println( br.readLine() );
```

Sendemail.java (4)

```
            ps.println("QUIT");
            System.out.println( br.readLine() );
            ps.flush();
            sock.close();
    }
}
```

TCP/IP client/server

TCP Server


Server
ServerSocket
Socket
InputStream
OutputStream
new
accept
getInputStream
getOutputStream
read
write
read
write
close

Echoserver.java (1)

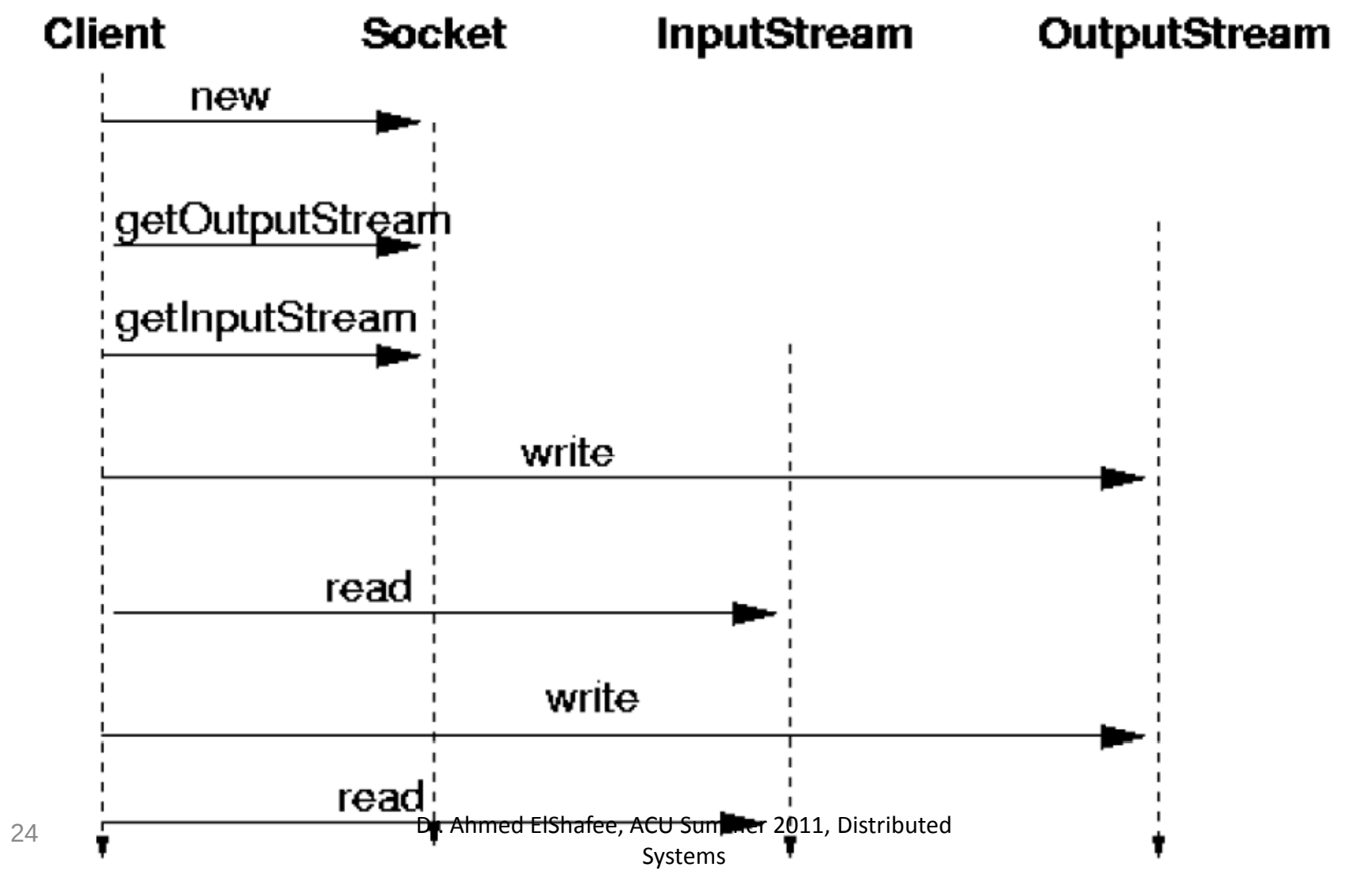
```
ServerSocket s = null;
try {s = new ServerSocket(MYECHOPORT);}

while (true)
        {
        Socket incoming = null;
        try {incoming = s.accept();}
        try {incoming.setSoTimeout(10000); //10 seconds}
        try {handleSocket(incoming);}
        try {incoming.close();}
        }
}
```

```
public static void handleSocket(Socket incoming) throws IOException
    {
    BufferedReader reader = new BufferedReader(new InputStreamReader(
    incoming.getInputStream()));
    PrintStream out = new PrintStream(incoming.getOutputStream());
    out.println(">>Server sent: Hello. Enter BYE to exit,..");
    System.out.println(">> Host is now connected,..");
    boolean done = false;
    while ( ! done)
            {
            String str = reader.readLine();
            out.println(str);
            System.out.println(">> Host sent: "+str);
            if (str.equals("BYE"))
                        done = true;
            }
```

```
        System.out.println(">> Host is now dis-connected,..");
        incoming.close();
        }
}
```

TCP client


Client
Socket
InputStream
OutputStream
new
getOutputStream
getInputStream
write
read
write
read

Echoclient.java (1)

```
Socket echoSocket = null;
PrintWriter out = null;
BufferedReader in = null;
BufferedReader stdIn = null;
try
        {
        echoSocket = new Socket(args[0], 8189);
        out = new PrintWriter(echoSocket.getOutputStream(), true);
        in = new BufferedReader(new
InputStreamReader(echoSocket.getInputStream()));
        System.out.println (in.readLine());
        }
```

```
try
        {
        stdIn = new BufferedReader(new InputStreamReader(System.in));
        String userInput;
        while (true)
                {
                System.out.print(">");
                userInput = stdIn.readLine();

                out.println(userInput);
                System.out.println(">>>Server sent: " + in.readLine());
                if (userInput.equals("BYE")) break;
                }
        }
```

UDP/IP client/server

Udpserver.java (1)

```
DatagramSocket serverSocket = new DatagramSocket(9876);
byte[] receiveData = new byte[1024];
byte[] sendData = new byte[1024];
while(true)
        {
        DatagramPacket receivePacket =
        new DatagramPacket(receiveData , receiveData.length);
        serverSocket.receive (receivePacket);
        String sentence = new String(receivePacket.getData());
        sentence=sentence.trim();
        InetAddress IPAddress = receivePacket.getAddress();
        int port = receivePacket.getPort();
```

Udpserver.java (2)

```
System.out.println(">>"+IPAddress.toString()+" sent: "+sentence );
                    sentence="Echo:"+sentence;
                    sendData = sentence.getBytes();
                    DatagramPacket sendPacket=new
  DatagramPacket(sendData,sendData.length,IPAddress,port);
                    serverSocket.send (sendPacket);
                    }
```

Udpclient.java (1)

```
BufferedReader inFromUser =new BufferedReader(new
InputStreamReader(System.in));
DatagramSocket clientSocket = new DatagramSocket();
InetAddress IPAddress = InetAddress.getByName (args[0]);
byte[] sendData = new byte[1024];
byte[] receiveData = new byte[1024];
while(true)
        {
        System.out.print(">>");
        String sentence = inFromUser.readLine();
        sendData = sentence.getBytes();
        DatagramPacket sendPacket= new DatagramPacket(sendData,
sendData.length , IPAddress, 9876);
        clientSocket.send(sendPacket);
```

Udpclient.java (2)

```
DatagramPacket receivePacket =
                    new DatagramPacket(receiveData, receiveData.length);
                    clientSocket.receive(receivePacket);
                    String modifiedSentence = new String (receivePacket. getData());
                    modifiedSentence=modifiedSentence.trim();
                    System.out.println(">>Sever sent: " + modifiedSentence );
                    if(sentence.equals("BYE"))break;

                    }
                    clientSocket.close();
```

Thanks,

See you next Week, isA