

## TCP

### Server

// Demonstrating Server-side Programming

import java.net.\*;

import java.io.\*;

public class Server {

// Initialize socket and input stream

private Socket s = null;

private ServerSocket ss = null;

private DataInputStream in = null;

// Constructor with port

public Server(int port) {

// Starts server and waits for a connection

try

{

ss = new ServerSocket(port);

System.out.println("Server started");

System.out.println("Waiting for a client ...");

s = ss.accept();

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

// Takes input from the client socket

in = new DataInputStream(

new BufferedInputStream(s.getInputStream()));

String m = "";

// Reads message from client until "Over" is sent

while (!m.equals("Over"))

{

try

{

m = in.readUTF();

System.out.println(m);

}

catch(IOException i)

{

System.out.println(i);

}

}

System.out.println("Closing connection");

// Close connection

s.close();

in.close();

```

    }
    catch(IOException i)
    {
        System.out.println(i);
    }
}

public static void main(String args[])
{
    Server s = new Server(5000);
}
}

```

### **Client**

// Demonstrating Client-side Programming

import java.io.\*;

import java.net.\*;

```

public class Client {

    // Initialize socket and input/output streams
    private Socket s = null;
    private DataInputStream in = null;
    private DataOutputStream out = null;

    // Constructor to put IP address and port
    public Client(String addr, int port)
    {
        // Establish a connection
        try {
            s = new Socket(addr, port);
            System.out.println("Connected");

            // Takes input from terminal
            in = new DataInputStream(System.in);

            // Sends output to the socket
            out = new DataOutputStream(s.getOutputStream());
        }
        catch (UnknownHostException u) {
            System.out.println(u);
            return;
        }
        catch (IOException i) {
            System.out.println(i);
            return;
        }
    }

    // String to read message from input
    String m = "";

    // Keep reading until "Over" is input

```

```
while (!m.equals("Over")) {
    try {
        m = in.readLine();
        out.writeUTF(m);
    }
    catch (IOException i) {
        System.out.println(i);
    }
}

// Close the connection
try {
    in.close();
    out.close();
    s.close();
}
catch (IOException i) {
    System.out.println(i);
}
}

public static void main(String[] args) {
    Client c = new Client("127.0.0.1", 5000);
}
}
```

## UDP

### Server

```
// Java program to illustrate Server side
// Implementation using DatagramSocket
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;

public class Server
{
    public static void main(String[] args) throws IOException
    {
        // Step 1 : Create a socket to listen at port 1234
        DatagramSocket ds = new DatagramSocket(1234);
        byte[] receive = new byte[65535];

        DatagramPacket DpReceive = null;
        while (true)
        {

            // Step 2 : create a DatagramPacket to receive the data.
            DpReceive = new DatagramPacket(receive, receive.length);

            // Step 3 : receive the data in byte buffer.
            ds.receive(DpReceive);

            System.out.println("Client:-" + data(receive));

            // Exit the server if the client sends "bye"
            if (data(receive).toString().equals("bye"))
            {
                System.out.println("Client sent bye.....EXITING");
                break;
            }

            // Clear the buffer after every message.
            receive = new byte[65535];
        }
    }

    // A utility method to convert the byte array
    // data into a string representation.
    public static StringBuilder data(byte[] a)
    {
        if (a == null)
            return null;
        StringBuilder ret = new StringBuilder();
        int i = 0;
        while (a[i] != 0)
        {
            ret.append((char) a[i]);
            i++;
        }
    }
}
```

```

    }
    return ret;
}
}

```

### Client

```

// Java program to illustrate Client side
// Implementation using DatagramSocket
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;

public class Client
{
    public static void main(String args[]) throws IOException
    {
        Scanner sc = new Scanner(System.in);

        // Step 1: Create the socket object for
        // carrying the data.
        DatagramSocket ds = new DatagramSocket();

        InetAddress ip = InetAddress.getLocalHost();
        byte buf[] = null;

        // loop while user not enters "bye"
        while (true)
        {
            String inp = sc.nextLine();

            // convert the String input into the byte array.
            buf = inp.getBytes();

            // Step 2 : Create the datagramPacket for sending
            // the data.
            DatagramPacket DpSend =
                new DatagramPacket(buf, buf.length, ip, 1234);

            // Step 3 : invoke the send call to actually send
            // the data.
            ds.send(DpSend);

            // break the loop if user enters "bye"
            if (inp.equals("bye"))
                break;
        }
    }
}

```