

# Lecture 48

## Java Networking

NPTEL

NPTEL





IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

# OBJECT ORIENTED PROGRAMMING WITH JAVA

## Java Networking

**Debasis Samanta**

Department of Computer Science & Engineering  
Indian Institute of Technology Kharagpur

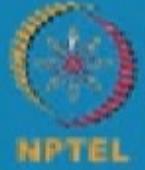




# Basics of Networking



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR

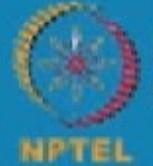


# What is Networking?

Networking is the process of connecting two or more computing devices together so that data and resources can be shared.



IIT KHARAGPUR



NPTEL  
ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



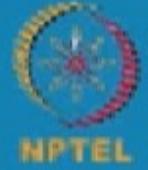
IIT KHARAGPUR



# Networking Terminologies



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR





# Networking Terminologies

The widely used networking terminologies are given below:

1. *LAN-MAN-WAN*
2. *WWW*
3. *IP Address*
4. *Port Number*
5. *URL*
6. *MAC Address*
7. *Socket*

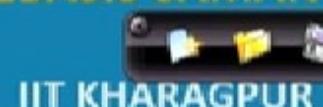


IIT KHARAGPUR



NPTEL  
NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR





# LAN-MAN-WAN

A **LAN** (Local Area Network) is a group of computers and network devices connected together, usually within the same building/ campus.

By definition, the connections must be high speed and relatively inexpensive (e.g., token ring or Ethernet). Most University departments are on LANs.



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR





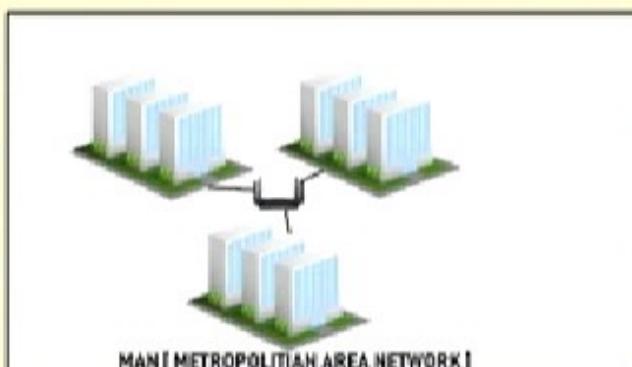
# LAN-MAN-WAN

A **LAN** (Local Area Network) is a group of computers and network devices connected together, usually within the same building/ campus.

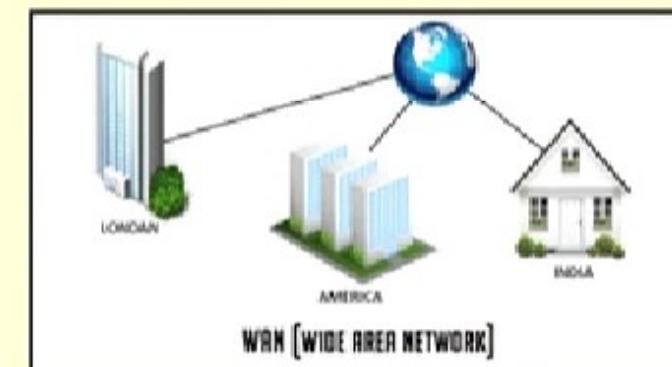
By definition, the connections must be high speed and relatively inexpensive (e.g., token ring or Ethernet). Most University departments are on LANs.



A **MAN** (Metropolitan Area Network) is a larger network that usually spans several buildings in the same city or town. Cable TV and Cable Internet are good examples for MAN.



A **WAN** (Wide Area Network), in comparison to a MAN, is not restricted to a geographical location, although it might be confined within the bounds of a state or country. The technology is high speed and relatively expensive. The Internet is an example of a worldwide public WAN.



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR



# WWW

The World Wide Web (WWW), also called the Web and W3, is an information space where documents and other web resources are identified by Uniform Resource Locators (URLs), interlinked by hypertext links, and accessible via the Internet.

English scientist Tim Berners-Lee invented the World Wide Web in 1989.

early milestones	Key Layers of the Internet	milestones
email@-1971 Ray Tomlinson	<b>CONTENT</b>	1987-HyperCard Bill Atkinson
Archie-1990 Emtage & Deutsch	<b>SEARCH ENGINE</b>	1998-Google Brin & Page
DOS Houdini-1986 Neil Larson	<b>BROWSERS</b>	1993-Mosaic Marc Andreessen
(Vannevar Bush, Ted Nelson, Douglas Engelbart)	<b>WORLD WIDE WEB</b>	1990-http:// Tim Berners-Lee
ARPANET-1969 J.C.R. Licklider	<b>INTERNET</b>	1975-TCP/IP Cerf & Kahn
SAGE-1956 George Valley	<b>NETWORKS</b>	1973-Ethernet Robert Metcalfe
Z3-1941 Konrad Zuse	<b>COMPUTERS</b>	1976-Apple Jobs & Wozniak



IIT KHARAGPUR



NPTEL  
NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR

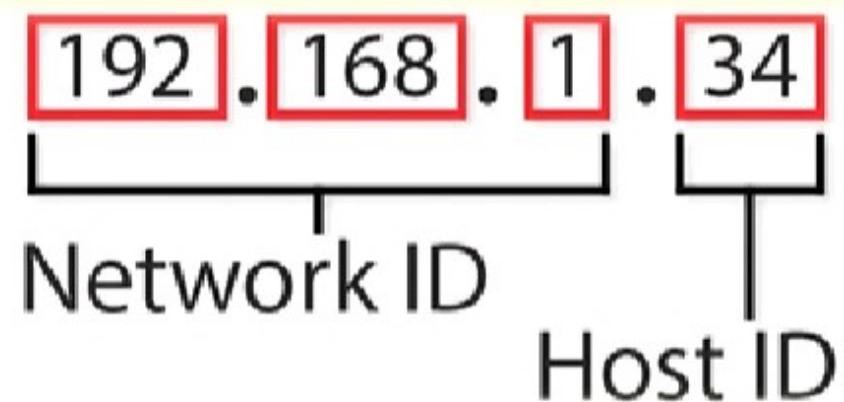


# IP Address

**IP address** is a unique number assigned to a node of a network, for example, 192.168.0.1 .

*It is composed of octets that range from 0 to 255.*

*It is a logical address that can be changed.*



IIT KHARAGPUR



NPTEL  
NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR

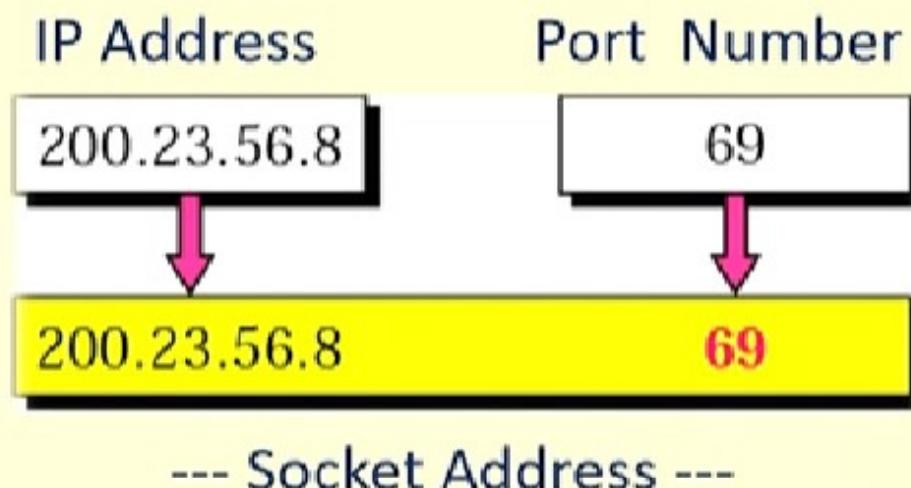


# Port number

The port number is used to uniquely identify different applications.

It acts as a communication endpoint between applications.

The port number is associated with the IP address for communication between two applications.

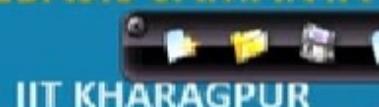


IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR





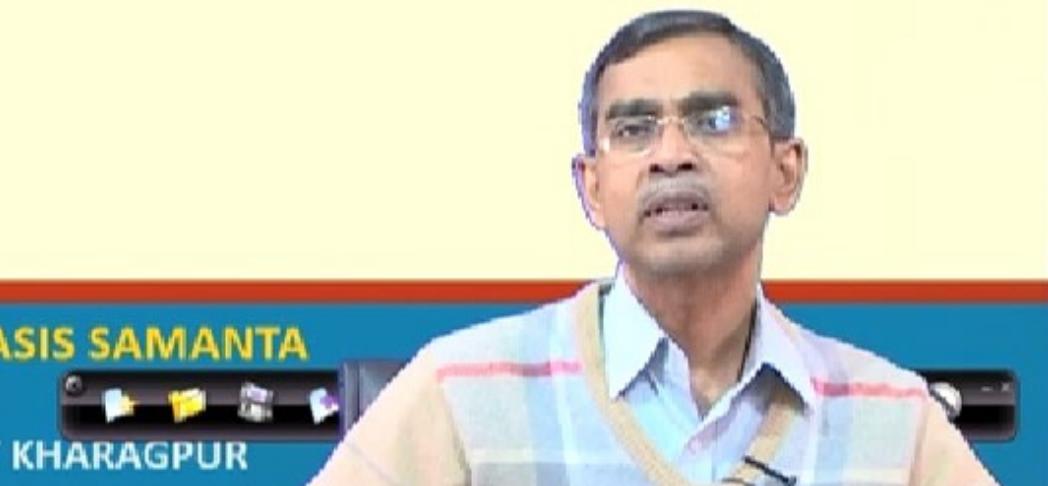
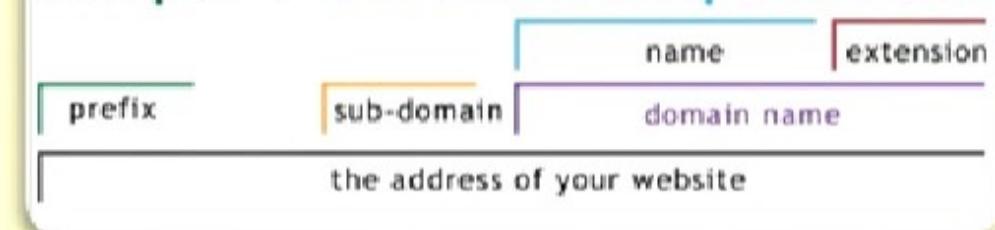
# URL

URL is the abbreviation of **Uniform Resource Locator** and is defined as the global address of documents and other resources on the World Wide Web.

To visit some website, for example Google, you'll go to the URL [www.google.com](http://www.google.com).

- The first part of the URL is called *protocol identifier* and it indicates what protocol to use,
- Second part is called *resource name* and it specifies the IP address or the domain name where the resource is located.
- The protocol identifier and the resource name are separated by a *colon* and *two forward slashes*.

`http://www.example.com`



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR



# MAC address

MAC (Media Access Control) address is a unique identifier of NIC (Network Interface Controller).

A network node can have multiple NICs but each with unique MAC.

## MAC

### Media Access Control Address



Organizationally Unique Identifier Universally Administered Address



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR





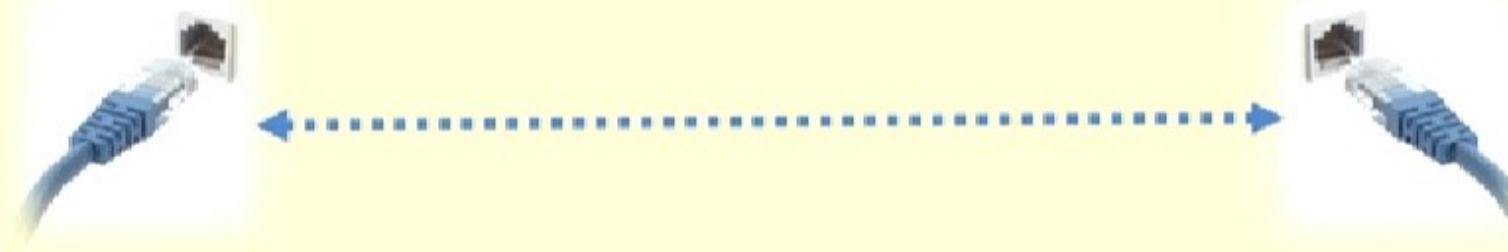
# Socket

A socket is an application program responsible for communication between two end points.

A socket is uniquely identified by an IP address and a Port.

Socket address

10.14.90.85:27017



Socket address

10.14.90.80:8080



**Note:** A socket is a software element, and it does not imply any hardware.

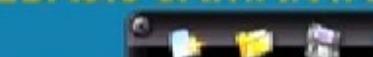


IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR

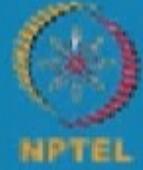




# Communication Protocols



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR





# Communication Protocols

1. Connection-less protocol (**UDP** – *User Datagram Protocol*)
2. Connection-oriented protocol (**TCP** – *Transmission Control Protocol*)
3. **TCP-IP** (*Internet Protocol*)
4. **FTP** (*File Transfer Protocol*)
5. **HTTP** (*Hypertext Transfer Protocol*)
6. **HTTPS** (*Secure Hypertext Transfer Protocol*)
7. **SMTP** (*Simple Mail Transfer Protocol*)

**https://www.**



IIT KHARAGPUR



NPTEL  
NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR



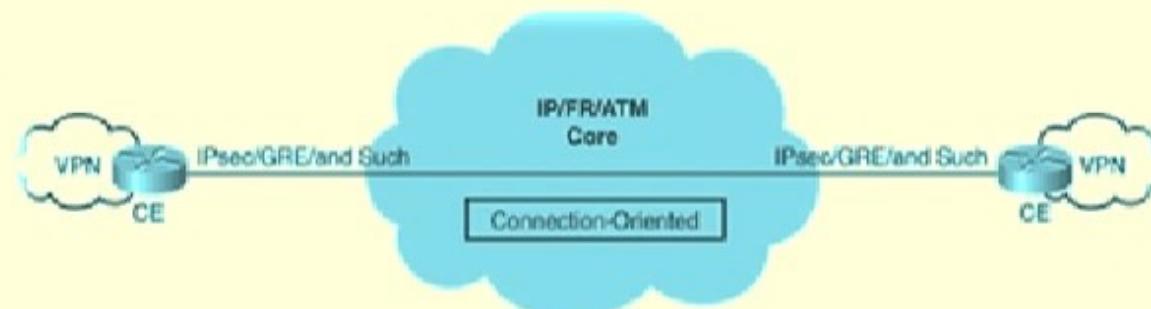


# Connection-oriented protocol

In connection-oriented protocol, acknowledgement is sent by the receiver.

So it is **reliable** but **slow**.

**TCP** follows connection-oriented protocol.



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA

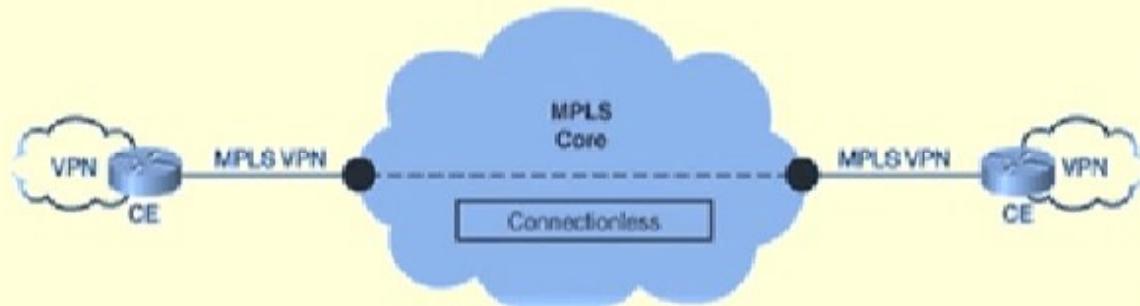


# Connection-less protocol

In connection-less protocol, acknowledgement is not sent by the receiver.

So it **is not reliable but fast**.

The example of connection-less protocol is **UDP**.



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA





# TCP-IP

**Transmission Control Protocol (TCP)** – It is known to provide reliable and error-free communication between end systems.

- It performs sequencing and segmentation of data.
- It also has acknowledgment feature and controls the flow of the data through flow control mechanism.
- It is a very effective protocol but has a lot of overhead due to such features. Increased overhead leads to increased cost.

**Internet Protocol (IP)** – It is responsible for delivering packets from the source host to the destination host by looking at the IP addresses in the packet headers.

IP has 2 versions: IPv4 and IPv6.

IPv4 is the one that most of the websites are using currently.

But IPv6 is growing as the number of IPv4 addresses are limited in number when compared to the number of users.



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR



**File Transfer Protocol (FTP)** is an application layer protocol which moves files between local and remote file systems. It runs on the top of TCP, like HTTP. To transfer a file, 2 TCP connections are used by FTP in parallel: *control connection and data connection*.

## Control connection

For sending control information like user identification, password, commands to change the remote directory, commands to retrieve and store files etc., FTP makes use of control connection. Control connection is initiated on port number 21.

## Data connection

For sending the actual file, FTP makes use of data connection. Data connection is initiated on port number 20.

FTP sends the control information out-of-band as it uses a separate control connection. Some protocols send their request and response header lines and the data in the same TCP connection. For this reason, they are said to send their control information in-band. HTTP and SMTP are such examples.





# HTTP & HTTPS

HTTP means Hyper Text Transfer Protocol.

HTTP is the underlying protocol used by the World Wide Web

This protocol defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands.





# HTTP & HTTPS

HTTP means Hyper Text Transfer Protocol.

HTTP is the underlying protocol used by the World Wide Web

This protocol defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands.



HTTPS means Hyper Text Transfer Protocol Secure

It is the secure version of HTTP

It means all communications between your browser and the website are encrypted.





# SMTP

**SMTP** means **Simple Mail Transfer Protocol**

SMTP is an application layer protocol.

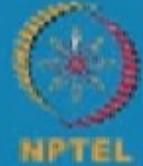
The client who wants to send the mail opens a TCP connection to the SMTP server and then sends the mail across the connection. The SMTP server is always on listening mode. As soon as it listens for a TCP connection from any client, the SMTP process initiates a connection on that port (numbered as 25). After successfully establishing the TCP connection the client process sends the mail instantly.



# SMTP



IIT KHARAGPUR



NPTEL  
NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR

# Overview HTTP-FTP-SMTP

PARAMETER	HTTP	FTP	SMTP
Port number	80	20 and 21	25
Type of band transfer	In-band	Out-of-band	In-band
State	Stateless	Maintains state	—
Number of TCP connections	1	2 (Data Connection and Control Connection)	1
Type of TCP connection	Can use both Persistent and Non-persistent	Persistent for Control connection. Non-persistent for Data Connection	Persistent
Type of Protocol	Pull Protocol (Mainly)	—	Push Protocol (Primarily)
Type of Transfer	Transfer files between Web server and Web client	Transfer directly between computers	Transfers mails via Mail Servers



# Java Networking



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR



# Objectives

For the networking with Java, it provides two types of sockets: **stream sockets** and **datagram sockets**.

## A. Stream Sockets

1. With stream sockets a process **establishes a connection** to another process.
2. While the connection is in place, data flows between the processes are **continuous streams**.
3. Here, stream sockets are said to provide a **connection-oriented service**.
4. The protocol used is **TCP** (Transmission Control Protocol).

## B. Datagram Sockets

1. With datagram sockets, individual **packets of information** are transmitted.
2. The transmission of packets follows a **connection less service**.
3. The protocol used is **UDP** (User Datagram Protocol).

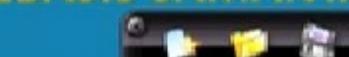


IIT KHARAGPUR

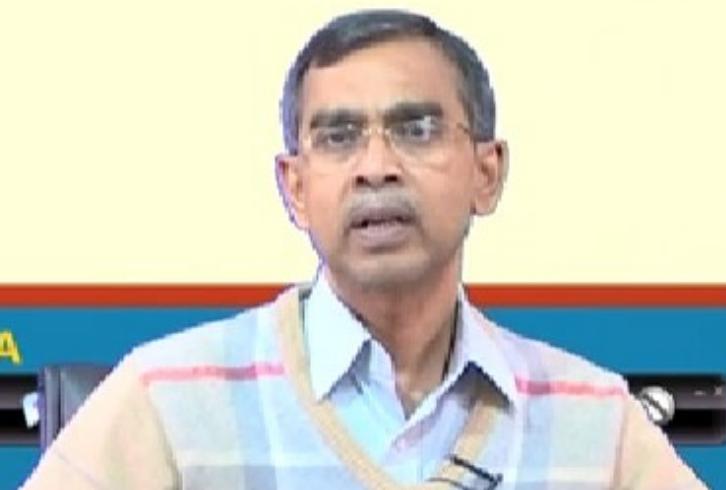


NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR





# TCP versus UDP

## TRANSMISSION CONTROL PROTOCOL (TCP)

TCP is a connection-oriented protocol. Connection-orientation means that the communicating devices should establish a connection before transmitting data and should close the connection after transmitting the data.

TCP is reliable as it guarantees delivery of data to the destination router.

TCP provides extensive error checking mechanisms. It is because it provides flow control and acknowledgment of data.

Sequencing of data is a feature of Transmission Control Protocol (TCP). this means that packets arrive in-order at the receiver.

TCP is comparatively slower than UDP.

Retransmission of lost packets is possible in TCP, but not in UDP.

TCP header size is 20 bytes.

TCP is heavy-weight.

TCP is used by HTTP, HTTPS, FTP, SMTP and Telnet

## USER DATAGRAM PROTOCOL (UDP)

UDP is the Datagram oriented protocol. This is because there is no overhead for opening a connection, maintaining a connection, and terminating a connection. UDP is efficient for broadcast and multicast type of network transmission.

The delivery of data to the destination cannot be guaranteed in UDP.

UDP has only the basic error checking mechanism using checksums.

There is no sequencing of data in UDP. If ordering is required, it has to be managed by the application layer.

UDP is faster, simpler and more efficient than TCP.

There is no retransmission of lost packets in User Datagram Protocol (UDP).

UDP Header size is 8 bytes.

UDP is lightweight.

UDP is used by DNS, DHCP, TFTP, SNMP, RIP, and VoIP.



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



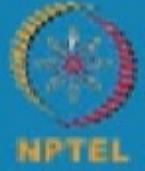
IIT KHARAGPUR



## Classes for Networking



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR



# Classis in Java for Networking

`URL`

`URLConnection`

`HttpURLConnection`

`InetAddress`



IIT KHARAGPUR



NPTEL  
NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR





# Classis in Java for Networking

`URL`

`URLConnection`

`HttpURLConnection`

`InetAddress`

`DatagramSocket`

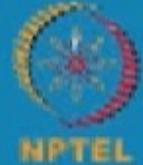
`ServerSocket`

`Socket`

`java.net` package



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



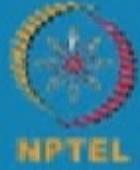
IIT KHARAGPUR



# Class URL



IIT KHARAGPUR

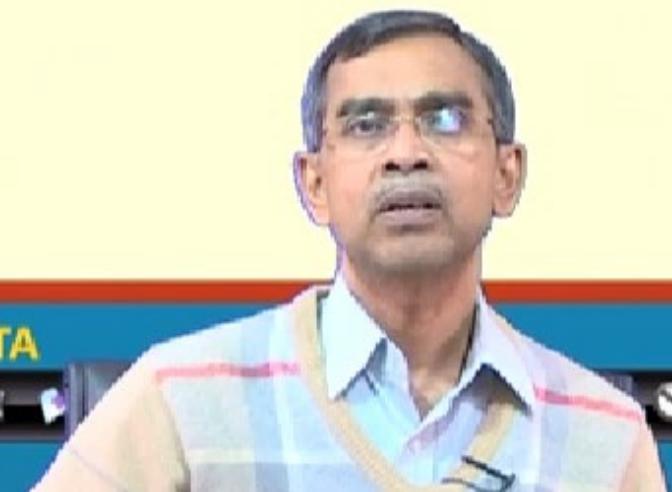


NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR





# Class URL

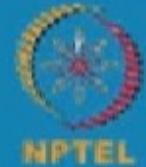
<https://nptel.ac.in/course.php>

A URL contains many information. For example

- **Protocol:** In this case, `https` is the protocol.
- **Server name or IP Address:** In this case, `nptel.ac.in` is the server name.
- **Port Number:** It is an optional attribute. If we write `https://nptel.ac.in:80/`, `80` is the port number. If port number is not mentioned in the URL, it will return `-1`.
- **File Name or directory name:** In this case, `course.php` is the file name.

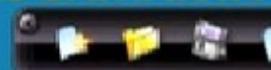


IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR





# Class URL

155.18.20.31

<https://nptel.ac.in/course.php>

A URL contains many information. For example

- **Protocol:** In this case, https is the protocol.
- **Server name or IP Address:** In this case, nptel.ac.in is the server name.
- **Port Number:** It is an optional attribute. If we write https://nptel.ac.in:80/ , 80 is the port number. If port number is not mentioned in the URL, it will return -1.
- **File Name or directory name:** In this case, course.php is the file name.

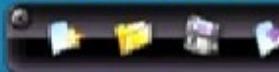


IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA





# Class URL

The Java URL class represents an URL. URL is an acronym for Uniform Resource Locator. It points to a resource on the World Wide Web. For example:

`https://nptel.ac.in/course.php`



**A URL contains many information. For example**

- **Protocol:** In this case, `https` is the protocol.
- **Server name or IP Address:** In this case, `nptel.ac.in` is the server name.
- **Port Number:** It is an optional attribute. If we write `https://nptel.ac.in:80/`, `80` is the port number. If port number is not mentioned in the URL, it will return `-1`.
- **File Name or directory name:** In this case, `course.php` is the file name.





# Class URL : Methods

The `java.net.URL` class provides many methods. The important methods of URL class are given below.

Method	Description
<code>public String getProtocol()</code>	Returns the protocol of the URL.
<code>public String getHost()</code>	Returns the host name of the URL.
<code>public String getPort()</code>	Returns the Port Number of the URL.
<code>public String getFile()</code>	Returns the file name of the URL.
<code>public URLConnection openConnection()</code>	It returns an instance of URLConnection associated with the URL.

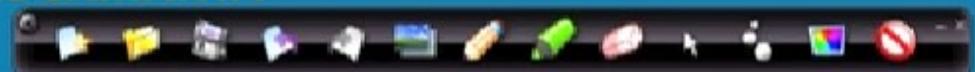


IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA





# Class URL : An example

```
import java.net.*;

public class URLClass{
    public static void main(String[] args) {
        try{
            URL url = new URL("https://nptel.ac.in/course.php");

            System.out.println("Protocol: "+url.getProtocol());
            System.out.println("Host Name: "+url.getHost());
            System.out.println("Port Number: "+url.getPort());
            System.out.println("File Name: "+url.getFile());

        } catch(Exception e){
            System.out.println(e);
        }
    }
}
```



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR



# Class URL : An example

```
import java.net.*;

public class URLClass{
    public static void main(String[] args) {
        try{
            URL url = new URL("https://nptel.ac.in/course.php");

            System.out.println("Protocol: "+url.getProtocol());
            System.out.println("Host Name: "+url.getHost());
            System.out.println("Port Number: "+url.getPort());
            System.out.println("File Name: "+url.getFile());

        } catch(Exception e){
            System.out.println(e);
        }
    }
}
```

```
H:\WORK\IMP\All Final\Week 10\Lecture Slides\Networking Demos\1. URL Class>java URLClass
Protocol: https
Host Name: nptel.ac.in
Port Number: -1
File Name: /course.php
```



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA

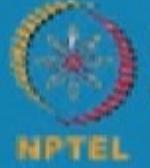




# Class URLConnection



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR



# Class URLConnection

The Java [URLConnection](#) class represents a communication link between the URL and the application. This class can be used to read and write data to the specified resource referred by the URL.

Some of the important **methods** provided by this class is shown below:

Method	Description
<code>openConnection()</code>	Returns the object of URLConnection class.
<code>getInputStream()</code>	Returns all the data of the specified URL in the stream that can be read and displayed

The syntax to get the object of URLConnection class

```
public URLConnection openConnection() throws IOException{}
```



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR



# Class URLConnection : An example

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

public class URLConnectionClass{
    public static void main(String[] args){
        try{
            URL url = new URL("https://nptel.ac.in/course.php");
            URLConnection urlcon = url.openConnection();
            InputStream stream = urlcon.getInputStream();
            int b;
            while((b = stream.read()) != -1){
                System.out.print((char)b);
            }
            }catch(Exception e){
                System.out.println(e);
            }
        }
}
```



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR

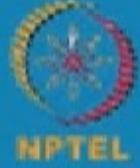




# Class HttpURLConnection



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA





# Class HttpURLConnection

The **Java HttpURLConnection** class is **http** specific **URLConnection**. It works for **HTTP protocol** only.

**This class can be used for extracting the following information of any HTTP URL**

- Header information.
- Status code.
- Response code, etc.

The syntax to get the object of **HttpURLConnection** class using typecasting

1. **public URLConnection openConnection() throws IOException{}**
2. **URL url = new URL("http://www.nptel.ac.in/java-tutorial");**
3. **HttpURLConnection huc = (HttpURLConnection) url.openConnection();**

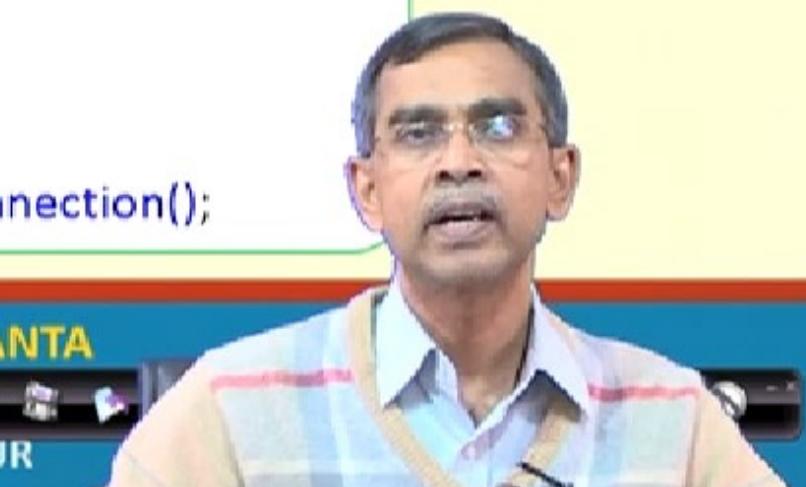
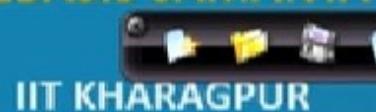


IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA





# Class HttpURLConnection

The Java [URLConnection](#) class represents a communication link between the URL and the application. This class can be used to read and write data to the specified resource referred by the URL.

Some of the important **methods** provided by this class is shown below:

Method	Description
<a href="#"><code>getHeaderFieldKey(int n)</code></a>	Returns the information in the header field n.
<a href="#"><code>getHeaderField(java.lang.String name)</code></a>	Returns all the header field.

*Note: There are many more methods in this class*

The syntax to get the object of URLConnection class

```
public URLConnection openConnection() throws IOException{}
```



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR





# Class HttpURLConnection : An example

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

public class HttpURLConnectionDemo{
    public static void main(String[] args){
        try{
            URL url = new URL("https://nptel.ac.in/course.php");
            HttpURLConnection huc = (HttpURLConnection)url.openConnection();
            for(int i=1;i<=8;i++){
                System.out.println(huc.getHeaderFieldKey(i)+" = "+huc.getHeaderField(i));
            }
            huc.disconnect();
        }catch(Exception e){
            System.out.println(e);
        }
    }
}
```



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR





# Class HttpURLConnection : An example

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

public class HttpURLConnectionDemo{
    public static void main(String[] args){
        try{
            URL url = new URL("https://nptel.ac.in/course.php");
            HttpURLConnection huc = (HttpURLConnection)url.openConnection();
            for(int i=1;i<5;i++){
                System.out.println(huc.getHeaderFieldKey(i)+" = "+huc.getHeaderField(i));
            }
            huc.disconnect();
        }catch(Exception e){
            System.out.println(e);
        }
    }
}
```



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA

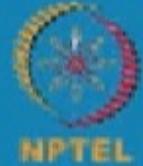




# Class InetAddress



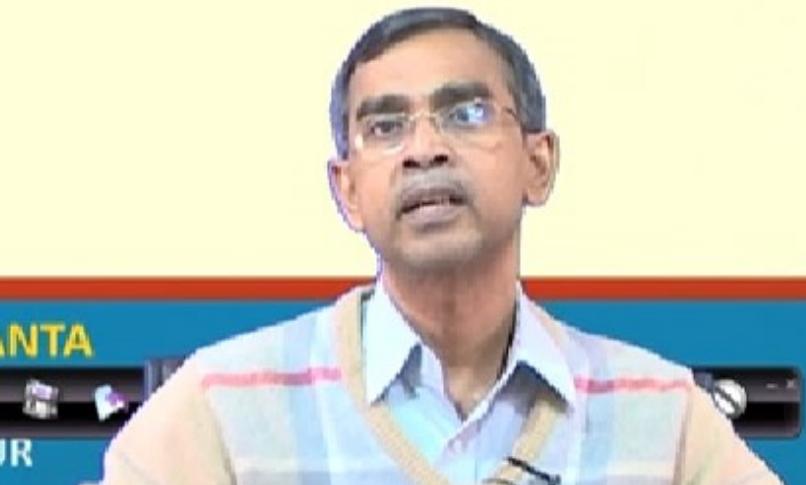
IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA

IIT KHARAGPUR



# Class InetAddress

Java **InetAddress** class represents an IP address. The **java.net.InetAddress** class provides methods to get the IP of any host name, For examples:

[www.nptel.ac.in](http://www.nptel.ac.in) [www.google.com](http://www.google.com) [www.wikipedia.org](http://www.wikipedia.org)

The **java.net.InetAddress** class provides many methods. The important methods of this class are given below.

Method	Description
<code>public static InetAddress getByName(String host) throws UnknownHostException</code>	Returns the instance of InetAddress containing LocalHost IP and name.
<code>public static InetAddress getLocalHost() throws UnknownHostException</code>	Returns the instance of InetAddress containing local host name and address.
<code>public String getHostName()</code>	Returns the host name of the IP address.
<code>public String getHostAddress()</code>	Returns the IP address in string format.
<code>public URLConnection openConnection()</code>	It returns instance of URLConnection i.e. associated with this URL.



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA





# Class InetAddress : An example

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

public class InetAddressClass{
    public static void main(String[] args){
        try{
            InetAddress ip = InetAddress.getByName("www.nptel.ac.in");

            System.out.println("Host Name: "+ip.getHostName());
            System.out.println("IP Address: "+ip.getHostAddress());

        }catch(Exception e){
            System.out.println(e);
        }
    }
}
```

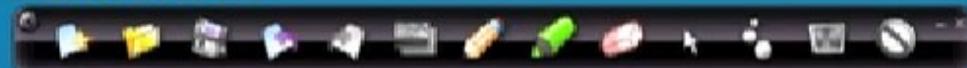


IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR



# Class InetAddress : An example

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

public class InetAddressClass{
    public static void main(String[] args){
        try{
            InetAddress ip = InetAddress.getByName("www.nptel.ac.in");

            System.out.println("Host Name: "+ip.getHostName());
            System.out.println("IP Address: "+ip.getHostAddress());

        }catch(Exception e){
            System.out.println(e);
        }
    }
}
```

```
H:\WORK\IMP\All Final\Week 10\Lecture Slides\Networking Demos\4. InetAddress Class>java InetAddressClass
Host Name: www.nptel.ac.in
IP Address: 14.139.160.71
```



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA





# Class DatagramSocket



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR



# Class DatagramSocket

Java **DatagramSocket** class represents a connection-less socket for sending and receiving **datagram** packets.

A datagram is basically an information but there is no guarantee of its content, arrival or arrival time.

*Commonly used Constructors of DatagramSocket class:*

Constructors	Description
<code>DatagramSocket()</code> throws SocketException	It creates a datagram socket and binds it with the available Port Number on the localhost machine.
<code>DatagramSocket(int port)</code> throws SocketException	It creates a datagram socket and binds it with the given Port Number.
<code>DatagramSocket(int port, InetAddress address)</code> throws SocketException	It creates a datagram socket and binds it with the specified port number and host address.



# Class DatagramSocket : Methods

The **DatagramSocket** class provides many methods. The important methods of this class are given below.

Method	Description
public InetAddress <b>getAddress</b> ( )	Returns the destination InetAddress. It is typically used for sending.
public int <b>getport</b> ( )	Returns the integer destination port number. It is typically used for sending.
public byte [ ] <b>getData</b> ( )	Returns the byte array of data contained in the datagram. It is used to retrieve data from the datagram after it has been received.
public int <b>getLength</b> ( )	Returns the length of the valid data contained in the byte array that would be returned from the getData() method.



# Class DatagramSocket : An example

```
import java.net.*;
public class Receiver{
    public static void main(String[] args) throws Exception {
        DatagramSocket ds = new DatagramSocket(3000);
        byte[] buf = new byte[1024];
        DatagramPacket dp = new DatagramPacket(buf, 1024);
        ds.receive(dp);
        String str = new String(dp.getData(), 0, dp.getLength());
        System.out.println(str);
        ds.close();
    }
}
```

Receiver

```
import java.net.*;
public class Sender{
    public static void main(String[] args) throws Exception {
        DatagramSocket ds = new DatagramSocket();
        String str = "Welcome to NPTEL";
        InetAddress ip = InetAddress.getByName("127.0.0.1");
        DatagramPacket dp = new DatagramPacket(str.getBytes(), str.length(), ip, 3000);
        ds.send(dp);
        ds.close();
    }
}
```

Sender

This example illustrate how a sender program sends some messages to the receiver program using **DatagramSocket** UDP protocol.

*Steps:*

1. First run the receiver program.
2. Run the sender program.



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR



# Class DatagramSocket : An example

```
import java.net.*;
public class Receiver{
    public static void main(String[] args) throws Exception {
        DatagramSocket ds = new DatagramSocket(3000);
        byte[] buf = new byte[1024];
        DatagramPacket dp = new DatagramPacket(buf, 1024);
        ds.receive(dp);
        String str = new String(dp.getData(), 0, dp.getLength());
        System.out.println(str);
        ds.close();
    }
}
```

Receiver

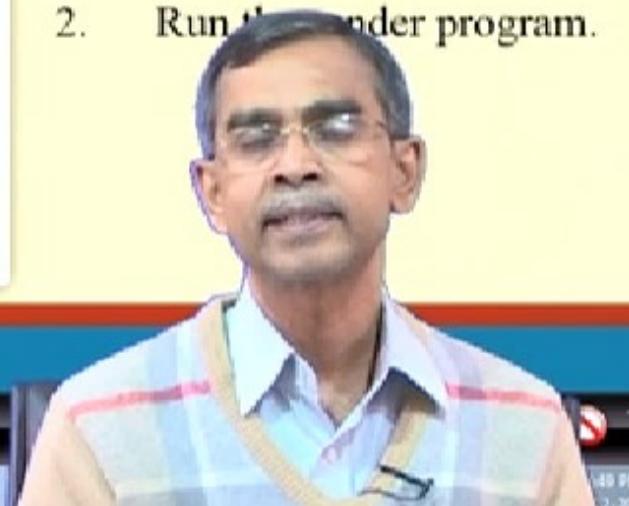
```
import java.net.*;
public class Sender{
    public static void main(String[] args) throws Exception {
        DatagramSocket ds = new DatagramSocket();
        String str = "Welcome to NPTEL";
        InetAddress ip = InetAddress.getByName("127.0.0.1");
        DatagramPacket dp = new DatagramPacket(str.getBytes(), str.length(), ip, 3000);
        ds.send(dp);
        ds.close();
    }
}
```

Sender

This example illustrate how a sender program sends some messages to the receiver program using **DatagramSocket** UDP protocol.

### Steps:

1. First run the receiver program.
2. Run the sender program.

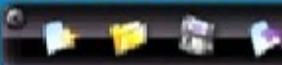


IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



# Class DatagramSocket : An example

```
import java.net.*;
public class Receiver{
    public static void main(String[] args) throws Exception {
        DatagramSocket ds = new DatagramSocket(3000);
        byte[] buf = new byte[1024];
        DatagramPacket dp = new DatagramPacket(buf, 1024);
        ds.receive(dp);
        String str = new String(dp.getData(), 0, dp.getLength());
        System.out.println(str);
        ds.close();
    }
}
```

Receiver

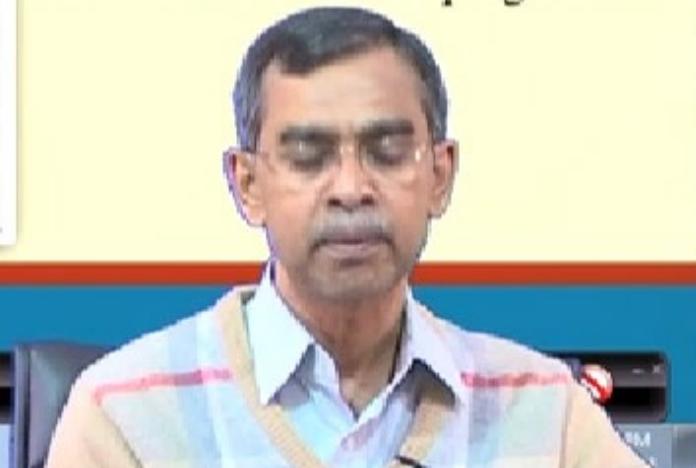
```
import java.net.*;
public class Sender{
    public static void main(String[] args) throws Exception {
        DatagramSocket ds = new DatagramSocket();
        String str = "Welcome to NPTEL";
        InetAddress ip = InetAddress.getByName("127.0.0.1");
        DatagramPacket dp = new DatagramPacket(str.getBytes(), str.length(), ip, 3000);
        ds.send(dp);
        ds.close();
    }
}
```

Sender

This example illustrate how a sender program sends some messages to the receiver program using **DatagramSocket** UDP protocol.

Steps:

1. First run the receiver program.
2. Run the sender program.



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA

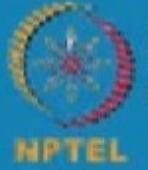




# Class ServerSocket



IIT KHARAGPUR



NPTEL  
NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA

IIT KHARAGPUR



# Class ServerSocket

The `java.net.ServerSocket` class is used by **server** applications to obtain a *port* and *listen* for client requests.

**ServerSocket** class has the following *constructors*:

Constructors	Description
<code>public ServerSocket(int port) throws IOException</code>	Attempts to create a server socket bound to the specified port. An exception occurs if the port is already bound by another application.
<code>public ServerSocket(int port, int backlog) throws IOException</code>	Similar to the previous constructor, the backlog parameter specifies how many incoming clients to store in a wait queue.
<code>public ServerSocket(int port, int backlog, InetAddress address) throws IOException</code>	Similar to the previous constructor, the InetAddress parameter specifies the local IP address to bind to. The InetAddress is used for servers that may have multiple IP addresses, allowing the server to specify which of its IP addresses to accept client requests on.
<code>public ServerSocket() throws IOException</code>	Creates an unbound server socket. When using this constructor, use the bind() method when you are ready to bind the server socket.



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA

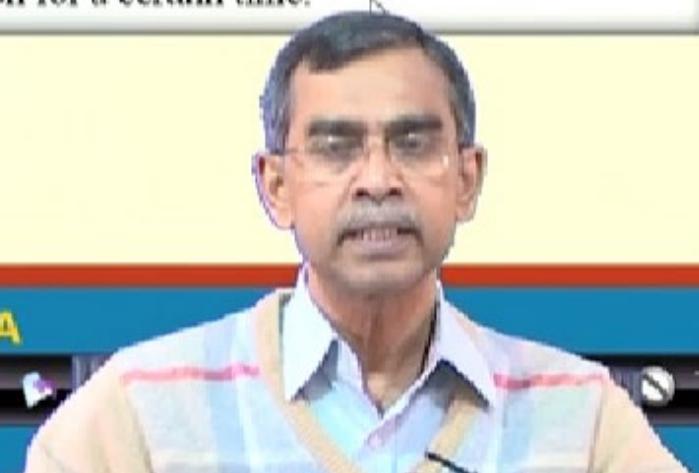




# Class ServerSocket

The `ServerSocket` class can be used to create a server socket. This object is used to establish communication with the clients and it has the following methods:

Method	Description
<code>public int getLocalPort()</code>	It connects to the specified server at the specified port. If this constructor does not throw an exception, the connection is successful and the client is connected to the server.
<code>public Socket accept() throws IOException</code>	This method accepts a connection to another socket (client).
<code>public void setSoTimeout(int timeout)</code>	Connects to the specified host and port, creating a socket on the local host at the specified address and port.
<code>public void bind(SocketAddress host, int backlog)</code>	This method establishes a connection for a certain time.

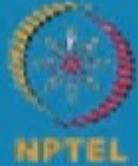




# Class Socket



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA

IIT KHARAGPUR





# Class Socket

A socket is simply an endpoint for communications between the machines. The `Socket` class can be used to create a socket. It has the following constructors :

Constructors	Description
<code>public Socket(String host, int port) throws UnknownHostException, IOException</code>	It connects to the specified server at the specified port. If this constructor does not throw an exception, the connection is successful and the client is connected to the server.
<code>public Socket(InetAddress host, int port) throws IOException</code>	This method is identical to the previous constructor, except that the host is denoted by an <code>InetAddress</code> object.
<code>public Socket(String host, int port, InetAddress localAddress, int localPort) throws IOException</code>	Connects to the specified host and port, creating a socket on the local host at the specified address and port.
<code>public Socket(InetAddress host, int port, InetAddress localAddress, int localPort) throws IOException</code>	This method is identical to the previous constructor, except that the host is denoted by an <code>InetAddress</code> object instead of a String.
<code>public Socket()</code>	Creates an unconnected socket. Use the <code>connect()</code> method to connect this socket to a server.



# Class Socket

Following are the methods present in the **Socket** class.:

Method	Description
<code>public void connect(SocketAddress host, int timeout) throws IOException</code>	This method connects the socket to the specified host. This method is needed only when you instantiate the Socket using the no-argument constructor.
<code>public InetAddress getInetAddress()</code>	This method returns the address of the other computer that this socket is connected to.
<code>public int getPort()</code>	Returns the port the socket is bound to on the remote machine.
<code>public int getLocalPort()</code>	Returns the port the socket is bound to on the local machine.
<code>public SocketAddress getRemoteSocketAddress()</code>	Returns the address of the remote socket.
<code>public InputStream getInputStream() throws IOException</code>	Returns the input stream of the socket. The input stream is connected to the output stream of the remote socket.
<code>public OutputStream getOutputStream() throws IOException</code>	Returns the output stream of the socket. The output stream is connected to the input stream of the remote socket.
<code>public void close() throws IOException</code>	Closes the socket, which makes this Socket object no longer usable for connecting again to any server.



IIT KHARAGPUR

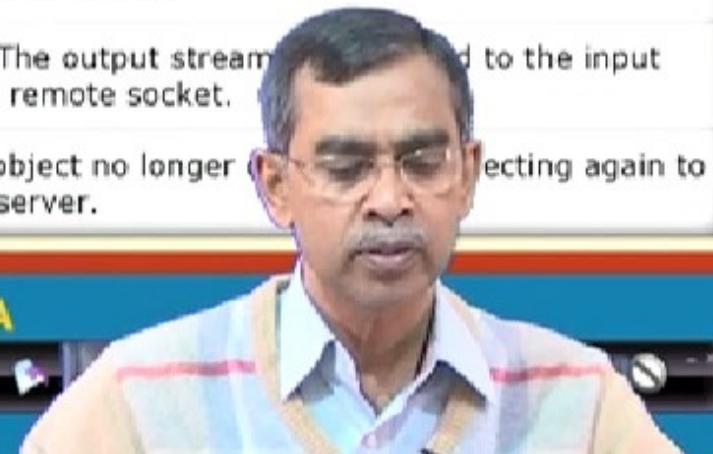


NPTEL  
ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR

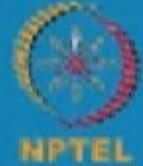




# Simple Client-Server Example



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA





# Example 1 : Simple Client-Server communication

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

public class MyServer {
    public static void main(String[] args) {
        try{
            ServerSocket ss = new ServerSocket(6666);
            Socket s = ss.accept(); //establishes connection
            DataInputStream dis = new DataInputStream(s.getInputStream());
            String str = (String)dis.readUTF();
            System.out.println("message= "+str);
            ss.close();
        }catch(Exception e){
            System.out.println(e);
        }
    }
}
```

Server

Let's see a simple example of Java socket programming in which client sends a text and server receives it. Shows the message and then closes the connection and execution stops.

This is the **server program**.



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



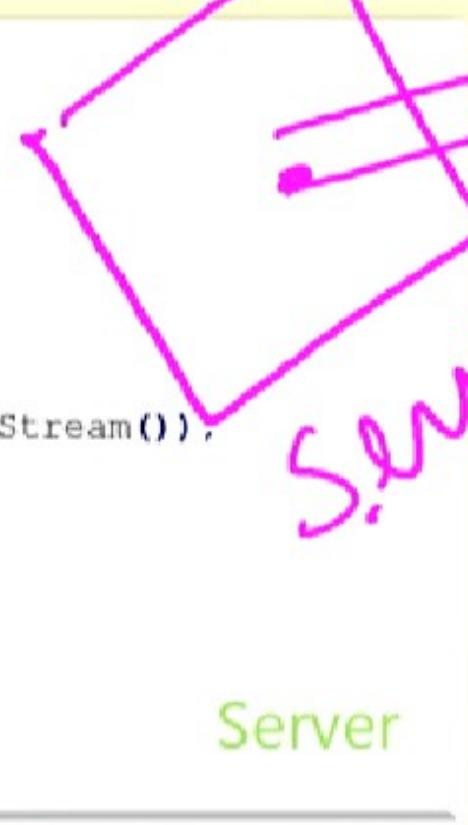
IIT KHARAGPUR



# Example 1 : Simple Client-Server communication

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

public class MyServer {
    public static void main(String[] args) {
        try{
            ServerSocket ss = new ServerSocket(6666);
            Socket s = ss.accept(); //establishes connection
            DataInputStream dis = new DataInputStream(s.getInputStream());
            String str = (String)dis.readUTF();
            System.out.println("message= "+str);
            ss.close();
        }catch(Exception e){
            System.out.println(e);
        }
    }
}
```



Server

Let's see a simple example of java socket programming in which client sends a text and server receives it. Shows the message and then closes the connection and execution stops.

This is the **server program**.

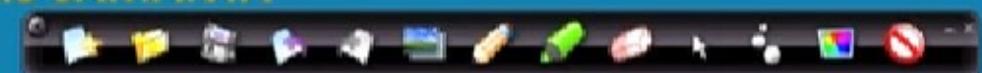


IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA





# Example 1 : Simple Client-Server Communication

This is the **client program**.

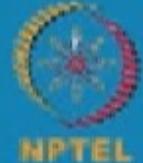
```
import java.io.*;
import java.net.*;

public class MyClient {
    public static void main(String[] args) {
        try{
            Socket s = new Socket("localhost",6666);
            DataOutputStream dout = new DataOutputStream(s.getOutputStream());
            dout.writeUTF("Hello Server");
            dout.flush();
            dout.close();
            s.close();
        }catch(Exception e){
            System.out.println(e);
        }
    }
}
```

Client



IIT KHARAGPUR



NPTEL  
NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR



# Example 1 : Simple Client-Server Communication

This is the **client program**.

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

public class MyClient {
    public static void main(String[] args) {
        try{
            Socket s = new Socket("localhost", 6666);
            DataOutputStream dout = new DataOutputStream(s.getOutputStream());
            dout.writeUTF("Hello Server");
            dout.flush();
            dout.close();
            s.close();
        }catch(Exception e){
            System.out.println(e);
        }
    }
}
```

Client



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



# Example 1 : Simple Client-Server communication

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

public class MyServer {
    public static void main(String[] args) {
        try{
            ServerSocket ss = new ServerSocket(6666);
            Socket s = ss.accept(); //establishes connection
            DataInputStream dis = new DataInputStream(s.getInputStream());
            String str = (String)dis.readUTF();
            System.out.println("message= "+str);
            ss.close();
        }catch(Exception e){
            System.out.println(e);
        }
    }
}
```

Server

Let's see a simple example of Java socket programming in which client sends a text and server receives it. Shows the message and then closes the connection and execution stops.

This is the **server program**.



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR



## Example 1 : Simple Client-Server communication

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

public class MyServer {
    public static void main(String[] args) {
        try {
            ServerSocket ss = new ServerSocket(6666);
            Socket s = ss.accept(); //establishes connection
            DataInputStream dis = new DataInputStream(s.getInputStream());
            String str = (String)dis.readUTF();
            System.out.println("message= "+str);
            ss.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

## Server

Let's see a simple example of java socket programming in which client sends a text and server receives it. Shows the message and then closes the connection and execution stops.

This is the **server** program.



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA





# Example 1 : Simple Client-Server communication

To execute this program open two command prompts and execute each program at each command prompt as displayed in the below figure. *First run server then client.*

The image shows two separate Command Prompt windows side-by-side. Both windows have a title bar labeled "Command Prompt". The left window displays the output of running the "MyServer" application, showing the message "Hello Server". The right window displays the output of running the "MyClient" application, showing the message "Hello Client". Both windows are running from the same directory path: H:\WORK\IMP\All Final\Week 10\Lecture Slides\Networking Demos\7. Socket - One way communication>java MyServer or MyClient.

```
H:\WORK\IMP\All Final\Week 10\Lecture Slides\Networking Demos\7. Socket - One way communication>java MyServer
message= Hello Server

H:\WORK\IMP\All Final\Week 10\Lecture Slides\Networking Demos\7. Socket - One way communication>java MyClient
message= Hello Client
```

After running the client application, a message will be displayed on the server and client console.



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



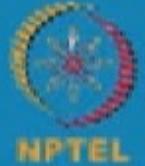
IIT KHARAGPUR



# Dialogue Client-Server Example (in the same machine)



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR

## Example 2 : Dialogue Client-Server

```
import java.net.*;
import java.io.*;
class MyServer{
    public static void main(String args[]) throws Exception{
        ServerSocket ss = new ServerSocket(3333);
        Socket s = ss.accept();
        DataInputStream din = new DataInputStream(s.getInputStream());
        DataOutputStream dout = new DataOutputStream(s.getOutputStream());
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String str1 = "",str2= "";
        while(!str1.equals("stop")){
            str1 = din.readUTF();
            System.out.println("client says: "+str1);
            str2 = br.readLine();
            dout.writeUTF(str2);
            dout.flush();
        }
        din.close();
        s.close();
        ss.close();
    }
}
```

Server

In this example, client will write first to the server then server will receive and print the text.

Then server will write to the client and client will receive and print the text. The step goes on.

This is the **server program** and next slide contains the **client program**.



## Example 2 : Dialogue Client-Server

```
import java.net.*;
import java.io.*;
class MyClient{
public static void main(String args[])throws Exception{
Socket s=new Socket("localhost", 3333);
DataInputStream din=new DataInputStream(s.getInputStream());
DataOutputStream dout=new DataOutputStream(s.getOutputStream());
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
String str1="",str2="";
while(!str1.equals("stop")){
Str1 = br.readLine();
dout.writeUTF(str1);
dout.flush();
Str2 = din.readUTF();
System.out.println("Server says: "+str2);
}
dout.close();
s.close();
}}
```

Client

This is the **client program**.

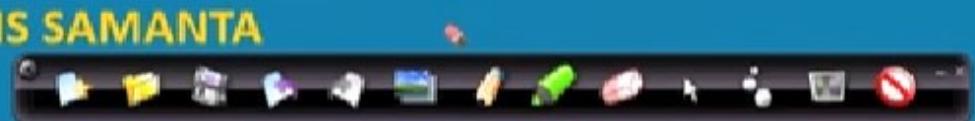


IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



03:57 PM  
22-12-2008



## Example 2 : Dialogue Client-Server

To execute this program open two command prompts and execute each program at each command prompt as displayed in the below figure. *First run server then client.*

The image shows two separate Command Prompt windows side-by-side. The top window is titled "Command Prompt - java MyServer" and the bottom window is titled "Command Prompt - java MyClient". Both windows have a black background and white text. In the "MyServer" window, the user types "java MyServer" and the server responds with "client says: Hi server! How are you? Hey, I'm fine, What about you client?". In the "MyClient" window, the user types "java MyClient" and the client responds with "Hi server! How are you? Server says: Hey, I'm fine, What about you client?". This illustrates a two-way communication between the client and the server.

```
C:\Users\Nilanjan Sinhababu\Desktop\IIT\NPTEL\Slides Final\Week 10\Codes\Socket - Two way communication>java MyServer
client says: Hi server! How are you?
Hey, I'm fine, What about you client?

C:\Users\Nilanjan Sinhababu\Desktop\IIT\NPTEL\Slides Final\Week 10\Codes\Socket - Two way communication>java MyClient
Hi server! How are you?
Server says: Hey, I'm fine, What about you client?
```

After running the client application, a message will be displayed on the server and client console.





## Dialogue Client-Server Example (between two remote machines)



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA

IIT KHARAGPUR





## Example 3 : Remote Client-Server dialogue

```
C:\Users\Nilanjan.Sinha\bash>ipconfig

Windows IP Configuration

Wireless LAN adapter Wi-Fi:
  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . :

Wireless LAN adapter Local Area Connection* 3:
  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . :

Wireless LAN adapter Local Area Connection* 3:
  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . :

Ethernet adapter Ethernet:
  Connection-specific DNS Suffix . : sit.iitkgp.ernet.in
  Link-local IPv6 Address . . . . . : fe80::44ce:6a7e:101a:3065%3
  IPv4 Address . . . . . : 10.14.97.253
  Subnet Mask . . . . . : 255.255.0.0
  Default Gateway . . . . . : 10.14.1.1

Ethernet adapter Bluetooth Network Connection:

  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . :

C:\Users\Nilanjan.Sinha\bash>
```

This is the same example as the previous program. The only difference is that *the server and the client are on different machines.*

In this case, the only difference is that the server should listen to the LAN IP address instead of local host and the client should know the IP and port of the server

This is the **server program** and next slide contains the **client program**.



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR

# Example 3 : Remote Client-Server dialogue

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

class MyServer{
    public static void main(String args[]) throws Exception{
        ServerSocket ss = new ServerSocket(3333, 2, InetAddress.getByName("10.14.97.211"));
        Socket s = ss.accept();
        DataInputStream din=new DataInputStream(s.getInputStream());
        DataOutputStream dout=new DataOutputStream(s.getOutputStream());
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        String str1 = "",str2="";
        while(!str1.equals("stop")){
            str1 = din.readUTF();
            System.out.println("client says: "+str1);
            str2 = br.readLine();
            dout.writeUTF(str2);
            dout.flush();
        }
        din.close();
        s.close();
    }
}
```

Server

This is the same example as the previous program. The only difference is that ***the server and the client are on different machines.***

In this case, the only difference is that the server should listen to the LAN IP address instead of local host and the client should know the IP and port of the server

This is the **server program** and next slide contains the **client program**.



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR



## Example 3 : Remote Client-Server dialogue

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

class MyServer{
    public static void main(String args[])throws Exception{
        ServerSocket ss = new ServerSocket(3333,7,InetAddress.getByName("10.14.97.21"));
        Socket s = ss.accept();
        DataInputStream din=new DataInputStream(s.getInputStream());
        DataOutputStream dout=new DataOutputStream(s.getOutputStream());
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        String str1 = "",str2="";
        while(!str1.equals("stop")){
            str1 = din.readUTF();
            System.out.println("client says: "+str1);
            str2 = br.readLine();
            dout.writeUTF(str2);
            dout.flush();
        }
        din.close();
        s.close();
    }
}
```

Server

This is the same example as the previous program. The only difference is that ***the server and the client are on different machines.***

In this case, the only difference is that the server should listen to the LAN IP address instead of local host and the client should know the IP and port of the server

This is the **server program** and next slide contains the **client program**.



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA





# Example 3 : Remote Client-Server dialogue

```
import java.net.*;
import java.io.*;
class MyClient{
    public static void main(String args[]) throws Exception{
        Socket s = new Socket("10.14.97.211", 3333);
        DataInputStream din=new DataInputStream(s.getInputStream());
        DataOutputStream dout=new DataOutputStream(s.getOutputStream());
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        String str1 = "",str2 = "";
        while(!str.equals("stop")){
            str=br.readLine();
            dout.writeUTF(str);
            dout.flush();
            str2 = din.readUTF();
            System.out.println("Server says: "+str2);
        }
        dout.close();
        s.close();
    }
}
```

Client

This is the client program.



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA

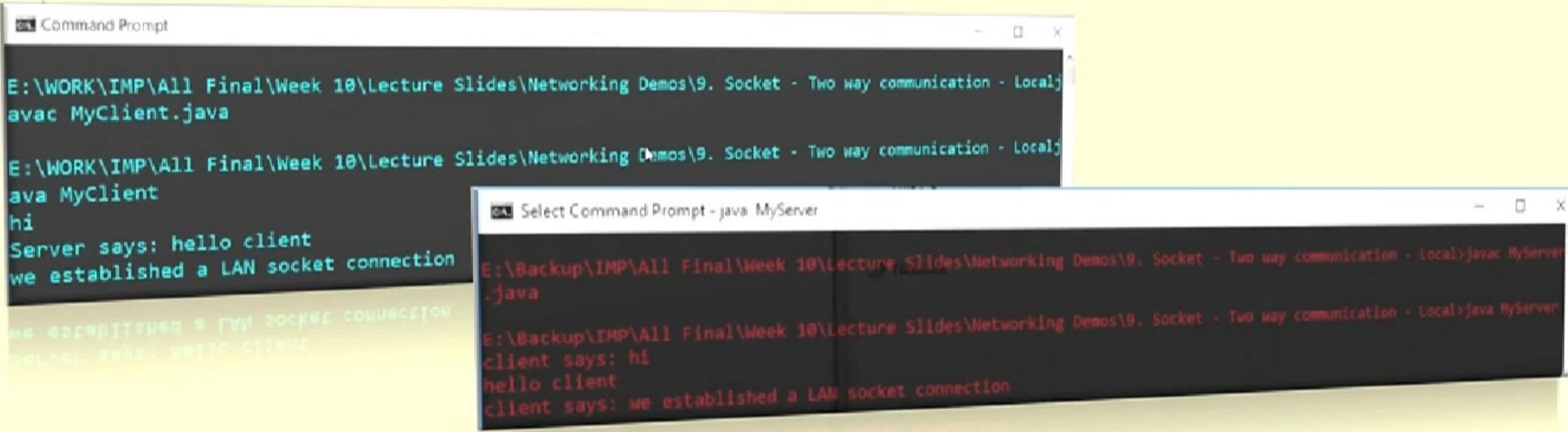


IIT KHARAGPUR



## Example 3 : Remote Client-Server dialogue

To execute this program open two command prompts and execute each program at each command prompt as displayed in the below figure. *First run server then client.*



```
Command Prompt
E:\WORK\IMP\All Final\Week 10\Lecture Slides\Networking Demos\9. Socket - Two way communication - Localjava MyClient.java

E:\WORK\IMP\All Final\Week 10\Lecture Slides\Networking Demos\9. Socket - Two way communication - Localjava MyClient
hi
Server says: hello client
we established a LAN socket connection
we established a LAN socket connection
client says: we established a LAN socket connection

Select Command Prompt - java MyServer
E:\Backup\IMP\All Final\Week 10\Lecture Slides\Networking Demos\9. Socket - Two way communication - Local\java MyServer.java

E:\Backup\IMP\All Final\Week 10\Lecture Slides\Networking Demos\9. Socket - Two way communication - Local\java MyServer
client says: hi
hello client
client says: we established a LAN socket connection
```

After running the client application, a message will be displayed on the server and client console.



# Concurrent Server

4



IIT KHARAGPUR



NPTEL  
NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR

# Concurrent server : An example

```
import java.io.*;
import java.text.*;
import java.util.*;
import java.net.*;
// Server class
public class MyServer{
    public static void main(String[] args) throws IOException{
        // server is listening on port 5056
        ServerSocket ss = new ServerSocket(5056);
        // running infinite loop for getting
        // client request
        while (true){
            Socket s = null;
            try{
                // socket object to receive incoming client requests
                s = ss.accept();
                System.out.println("A new client is connected : " + s);
                // obtaining input and out streams
                DataInputStream dis = new DataInputStream(s.getInputStream());
                DataOutputStream dos = new DataOutputStream(s.getOutputStream());
                System.out.println("Assigning new thread for this client");
                // create a new thread object
                Thread t = new ClientHandler(s, dis, dos);
                // Invoking the start() method
                t.start();
            }
        }
    }
}
```

Server

In this example, we will create a Date-Time server, and clients can either view '**Date**' or '**Time**' as per their selection. Also, Clients can close the connection by typing '**Exit**'.

Since we will be using threads in the program, multiple clients can be connected and request information from the server simultaneously.

This is the **server program** followed by the **client program**.



IIT KHARAGPUR



NPTEL  
NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR

# Example 3 : Concurrent server

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

// ClientHandler class
class ClientHandler extends Thread
{
    DateFormat fordate = new SimpleDateFormat("yyyy/MM/dd");
    DateFormat fortime = new SimpleDateFormat("hh:mm:ss");
    final DataInputStream dis;
    final DataOutputStream dos;
    final Socket s;
    // Constructor
    public ClientHandler(Socket s, DataInputStream dis, DataOutputStream dos){
        this.s = s;
        this.dis = dis;
        this.dos = dos;
    }

    public void run(){
        String received;
        String toreturn;
    }
}
```

Server

This is the **server program**.

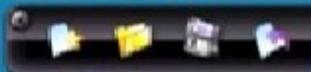


IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



# Concurrent server : An example

```
import java.io.*;
import java.text.*;
import java.util.*;
import java.net.*;
// Server class
public class MyServer{
    public static void main(String[] args) throws IOException{
        // server is listening on port 5056
        ServerSocket ss = new ServerSocket(5056);
        // running infinite loop for getting
        // client request
        while (true){
            Socket s = null;
            try{
                // socket object to receive incoming client requests
                s = ss.accept();
                System.out.println("A new client is connected : " + s);
                // obtaining input and out streams
                DataInputStream dis = new DataInputStream(s.getInputStream());
                DataOutputStream dos = new DataOutputStream(s.getOutputStream());
                System.out.println("Assigning new thread for this client");
                // create a new thread object
                Thread t = new ClientHandler(s, dis, dos);
                // Invoking the start() method
                t.start();
            }
        }
    }
}
```

Server

In this example, we will create a Date-Time server, and clients can either view '**Date**' or '**Time**' as per their selection. Also, Clients can close the connection by typing '**Exit**'.

Since we will be using threads in the program, multiple clients can be connected and request information from the server simultaneously.

This is the **server program** followed by the **client program**.



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



# Example 3 : Concurrent server

```
while (true){  
    try {  
        // Ask user what he wants  
        dos.writeUTF("What do you want? [Date | Time]..\n"+  
                    "Type Exit to terminate connection.");  
        // receive the answer from client  
        received = dis.readUTF();  
        if(received.equals("Exit")){  
            System.out.println("Client " + this.s + " sends exit...");  
            System.out.println("Closing this connection.");  
            this.s.close();  
            System.out.println("Connection closed");  
            break;  
        }  
        // creating Date object  
        Date date = new Date();  
        // write on output stream based on the  
        // answer from the client  
        switch (received) {  
            case "Date" :  
                toreturn = fordate.format(date);  
                dos.writeUTF(toreturn);  
                break;  
        }  
    }  
}
```

Server

This is the **server program**.



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR

# Example 3 : Concurrent server

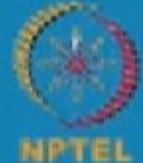
```
        case "Time" :  
            toreturn = fortome.format(date);  
            dos.writeUTF(toreturn);  
            break;  
        default:  
            dos.writeUTF("Invalid input");  
            break;  
    }  
} catch (IOException e) {  
    e.printStackTrace();  
}  
}  
try{  
    // closing resources  
    this.dis.close();  
    this.dos.close();  
}catch(IOException e){  
    e.printStackTrace();  
}  
}  
}
```

Server

This is the **server program**.

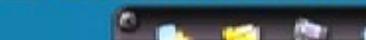


IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR





# Example 3 : Concurrent server (Client)

```
import java.io.*;
import java.net.*;
import java.util.Scanner;
// Client class
public class MyClient{
    public static void main(String[] args) throws IOException{
        try{
            Scanner scn = new Scanner(System.in);
            // getting localhost ip
            InetAddress ip = InetAddress.getByName("localhost");
            // establish the connection with server port 5056
            Socket s = new Socket(ip, 5056);
            // obtaining input and out streams
            DataInputStream dis = new DataInputStream(s.getInputStream());
            DataOutputStream dos = new DataOutputStream(s.getOutputStream());
            // the following loop performs the exchange of
            // information between client and client handler
            while (true){
                System.out.println(dis.readUTF());
                String tosend = scn.nextLine();
                dos.writeUTF(tosend);
            }
        }
    }
}
```

Client

This is the **client program**.



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA





# Example 3 : Concurrent server (Client)

```
// If client sends exit,close this connection  
// and then break from the while loop  
if(tosend.equals("Exit")){  
    System.out.println("Closing this connection : " + s);  
    s.close();  
    System.out.println("Connection closed");  
    break;  
}  
// printing date or time as requested by client  
String received = dis.readUTF();  
System.out.println(received);  
}  
// closing resources  
scn.close();  
dis.close();  
dos.close();  
}catch(Exception e){  
    e.printStackTrace();  
}  
}  
}
```

Client

This is the **client program**.



IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR



# Example 3 : Dialogue Client-Server

The screenshot shows three Command Prompt windows illustrating a client-server dialogue:

- Server:** Shows the server application running. It logs new client connections and handles their requests. The log includes:

```
Windows\IMP\All Final\Week 10\lecture Slides\Networking Demos\10. Concurrent Server>java MyServer
A new client is connected : Socket[addr=127.0.0.1,port=50447,localport=5056]
Assigning new thread for this client
A new client is connected : Socket[addr=127.0.0.1,port=50450,localport=5056]
Assigning new thread for this client
Client Socket[addr=127.0.0.1,port=50450,localport=5056] sends exit...
Closing this connection.
Connection closed.
```
- Client 1:** Shows a client application interacting with the server. It asks for a date/time, receives it, and then exits.

```
Windows\IMP\All Final\Week 10\lecture Slides\Networking Demos\10. Concurrent Server>java MyClient
What do you want?(Date | Time)..
Type Exit to terminate connection.
date
Invalid input.
What do you want?(Date | Time)..
Type Exit to terminate connection.
Date
2018/12/28
What do you want?(Date | Time)..
Type Exit to terminate connection.
```
- Client 2:** Shows another client application interacting with the server. It asks for a time, receives it, and then exits.

```
Windows\IMP\All Final\Week 10\lecture Slides\Networking Demos\10. Concurrent Server>java MyClient
what do you want?(Date | Time)..
Type Exit to terminate connection.
Time
05:55:23
what do you want?(Date | Time)..
Type Exit to terminate connection.
Exit
Closing this connection : Socket[addr=localhost/127.0.0.1,port=5056,localport=50450]
Connection closed
```

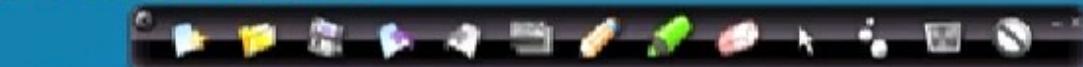


IIT KHARAGPUR



NPTEL ONLINE  
CERTIFICATION COURSES

DEBASIS SAMANTA



IIT KHARAGPUR