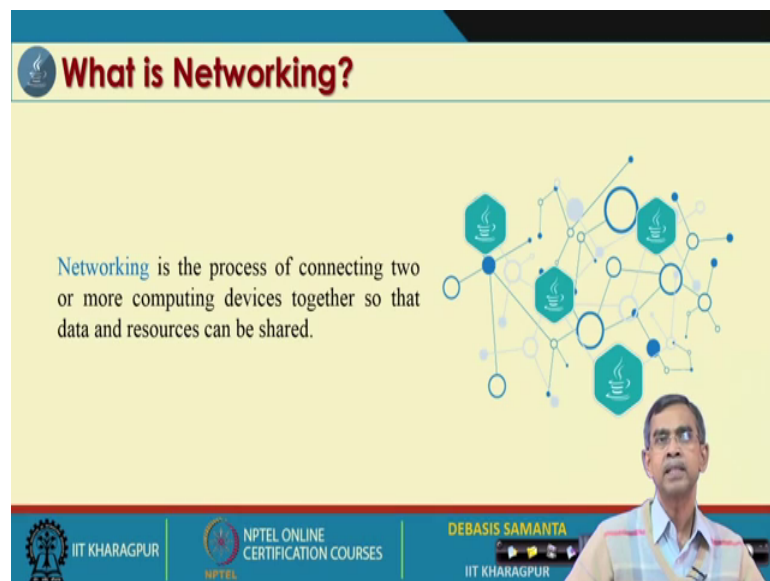


Programming in Java
Prof. Debasis Samanta
Department of Compute Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 48
Java Networking

So, now we are in a position to discuss about network related programming in Java. So, basically the idea of the networking is useful to I mean do certain what is called the programming, so that the communication between two so users can be achieved. So, here we have to cover how this network related programming can be done, what are the different supports the those are available in Java networking, we will discuss. Before going to actual networking related programming, we have to have certain familiarity related to the network related terminologies.

(Refer Slide Time: 01:11)



What is Networking?

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

The slide features a network diagram with nodes and connections, and a small inset image of Prof. Debasis Samanta in the bottom right corner.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

And then we will discuss those things, then these are the things that is required to understand the next discussion. So, we will quickly cover the basic concepts of networking and then finally, the Java. Now, you know exactly what is the process of networking, it is basically is a process of communication between the two computers we can say and they can communicate by this ok, if they are the connected through net network actually, communication channel right; they can communicate, they can share the file, they can talk to each other's and like this one. So, these are the things it is there.

And the networking is an issue, because as the time passing on the number of users are more those are connected in the network and it becomes really very complex and complicated, how this programming issues can be handled. And this is (Refer Time: 01:56) it has been evolved many ideas that how this networking can be done, the many tools, there many what is called the environments are available, out of which Java gives a very good one system, which basically allows you to write your own program and using this program you can communicate ah communicate to any parties actually. So, this is basically the basic idea about the Java networking.

(Refer Slide Time: 02:23)

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

KEY TERMS

DEBASIS SAMANTA
IIT KHARAGPUR

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NPTEL

Now, I will just quickly recapitulate the different terminology those are relevant to the networking related issues are there. And probably you know all those terminology, I will just quickly touch there, so like LAN-MAN-WAN, WWW, IP address, port number, URL, then MAC address and sockets, because all those things will be referred into our Java based network programming concept is there.

(Refer Slide Time: 02:49)

The slide is titled "LAN-MAN-WAN" and is divided into three columns. The first column defines a LAN (Local Area Network) as a group of computers and network devices connected together, usually within the same building/campus. It notes that connections must be high speed and relatively inexpensive (e.g., token ring or Ethernet) and that most university departments are on LANs. The second column defines a MAN (Metropolitan Area Network) as a larger network that usually spans several buildings in the same city or town, with cable TV and cable internet as examples. The third column defines a WAN (Wide Area Network) as a network not restricted to a geographical location, though it might be confined to a state or country. It notes that the technology is high speed and relatively expensive, and that the Internet is an example of a worldwide public WAN. Each column includes a small diagram: LAN shows a central hub connected to several desktop computers; MAN shows several buildings connected to a central hub; WAN shows a globe with lines connecting different geographical locations. The slide footer includes logos for IIT KHARAGPUR, NPTEL ONLINE CERTIFICATION COURSES, and DEBASIS SAMANTA.

Now, so for the local area network is concerned. It is basically the network is a connection of computers of course, but this is called a local network if the network spans within say maximum 1 to 10 kilometers within the range. So, usually it is basically in a campus or even a large building, then the network that needs to be carried out is called the local area network; it is simply called the LAN.

On the other hand, medium area network a little bit larger than this one, this span is usually 10 to 40 kilometers. So, if you have to carried out certain communication within the city say suppose or a innovative metropolitan area, then it is called the medium metropolitan area network that is why it is called the metropolitan area network or sometime medium network also we can say. And is example is that cable TV connection and or cable internet connection, all these things are basically an example of metropolitan area network.

On the other hand, wide area network if the network ah what is called a connectivity extends up to a state or even the entire country, then it is called the wide area network. And the span is basically 40 kilometers and above. So, usually internet is a good example of the public worldwide area network, there are may be public network, there may be private network. Private network is limited to among the peer to peer communications only on the other hand, public means anybody can be connected and then they can share them use they can use this one.

(Refer Slide Time: 04:29)

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.

Key Layers of the Internet

early milestones		milestones
email@-1971 Ray Tomlinson	CONTENT	1987-HyperCard Bill Atkinson
Archie-1990 Emlage & Deutsch	SEARCH ENGINE	1998-Google Brin & Page
DOS Houdini-1988 Neil Larson	BROWSERS	1993-Mosaic Marc Andreessen
(Vannevar Bush, Ted Nelson, Douglas Engelbart)	WORLD WIDE WEB	1990-http:// Tim Berners-Lee
ARPANET-1969 J.C.R. Licklider	INTERNET	1975-TCP/IP Cerf & Kahn
SAGE-1956 George Valley	NETWORKS	1973-Ethernet Robert Metcalfe
Z3-1941 Konrad Zuse	COMPUTERS	1976-Apple Jobs & Wozniak

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

Now, WWW as you have heard about it, it is basically a popularly called the W3 or simply web is a World Wide Web, is a web means it just like a net actually you can see, who is basically spread. And the worldwide web is basically related to internet also, because internet is a part of the worldwide. So, what is the World Wide Web? Actually, it is a collection of huge what is called the documents, which are spread over the entire what is called the network or you can more precisely call the internet. So, sometimes worldwide web is also called network of networks that means, is there are a lot of networks which are connected to each other's is called that worldwide.

And related to World Wide Web, there are many keywords which has been evolved and first said, 50 years or so, which we have mention like browsers, the search engine, the content, hypertext documents, the computer, networks and so many things are there. All those things evolved not in a day of course gradually, and today basically we see exactly what is the World Wide Web to and today World Wide Web is basically indispensable in our day to day activities.

(Refer Slide Time: 05:47)

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.

192 . 168 . 1 . 34

Network ID Host ID

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES DEBASIS SAMANTA
IIT KHARAGPUR

Now, so far the different computer connectivity's are concerned a every computer should be uniquely identified, there should be some numbers for that and that is what the W3 consortium they proposed one mechanism, it is called the IP address concept. So, it is basically IP stands for Internet Protocol.

So, IP address usually is a 4 byte address segmented into 8 byte each, and then the last 8 byte is basically host id that means, if one computer is connected to a network and that network may have its own id, usually that is decided by the first 3 bytes; in the last byte is for the computer which is connected to this network. So, it is basically as it is a 8 by 2, so each byte value if you express it is basically in the range 0 to 255 as an example 192.168 dot 0 is the basically one network id, and then dot 1 is basically the host id means id of a computer. So, these address is not exactly the physical address, logical address.

Logical I say in the sense that whenever one user wants to connect to a network, he or she should approach to the network administrator, the network administer will assign one id; whether it is a network service provider or your network administrator in your organization, whatever it is there.

(Refer Slide Time: 07:19)

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.

IP Address	Port Number
200.23.56.8	69
200.23.56.8	69

--- Socket Address ---

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

So, this is the IP address and in addition to this IP address, there is also on another identity is required it is called the port number. So, port number related to a particular and application. So, whenever you run one program, whether it is a Gmail program or is a browser program or whatever it is there. So, it is basically created an application.

So, every application should be uniquely identified, usually it is that identification it by means of it is called the port number. So, usually port number is related to the machine itself. So, every machine is given a unique port number and together the IP address and port number actually called the socket address. So, socket address nothing but this is an address of an application.

(Refer Slide Time: 08:03)

The slide is titled "URL" and features a blue header with a logo. The main text explains that URL stands for Uniform Resource Locator and is the global address for documents on the World Wide Web. It provides an example of a URL: <http://www.google.com>. A diagram breaks down the URL into its components: "http://" is the prefix, "www" is the sub-domain, "example" is the domain name, and ".com" is the extension. A list of three bullet points explains each part: 1. The first part is the protocol identifier, indicating the protocol to use. 2. The second part is the resource name, specifying the IP address or domain name. 3. The protocol identifier and resource name are separated by a colon and two forward slashes. The slide also includes logos for IIT Kharagpur and NPTEL, and a small video inset of a man speaking.

And then there is also the concept of URL, probably we familiar to an example of URL like `www dot google dot com`. So, this is the one example of URL it is the full form is Uniform Resource Locator. Now, here it is basically one idea about by which a particular what is called the document, we can say the hypertext document like or a particular server can be identified.

Is basically if we have this URL, then we will be able to quickly look at a computer where it is located. And for this location as you see there are many parts are involved, one is called the domain name. And the domain name has the 3 components, is the basically connected to which computer for you `www` this means is connected to web that means, internet. And `dot example` is basically the name of the server, where that particular file or document is located. And `dot com` is the name of the organization and that means, it is basically `dot com` or is basically say `dot i n` that means, it is connected to India or `dot a u` Australia or `dot u a` USA these kind of things are there `dot UK` United Kingdom like this one.

So, the different country has the different what is called the extension of these domain name. And then private organization `dot com` is basically for all commercial documents related and like this one, `dot org` the for educational institution, `dot edu` also other additional institution like. So, these are the different organized wise the naming is there.

Any way in addition to this, so this is basically called the location of a particular file and there is also another is called the protocol. So, which protocol that it should use in order to access that document, it is specified by that one for example, http it is basically protocol. So, it is basically (Refer Time: 10:03) protocol and then followed by the resource name, resource name is basically www dot example dot com for example in this case, is basically both together includes the URL.

(Refer Slide Time: 10:29)

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

00	1A	3F	F1	4C	C6
----	----	----	----	----	----

Organizationally Unique Identifier Universally Administered Address

DEBASIS SAMANTA
IIT KHARAGPUR

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES NPTEL

So, there are different instead of protocol, here we have use example one protocol http, near there may be ftp or ftp or smtp like this. So, different protocol is for dual discuss about the different protocol shortly. And then MAC address is basically is a another physical address, where the IP address is a logical. So, it is physical address means whenever you purchase a computer, the manufacturer assigns on unique number to this computer and these are this is called the MAC address, the full form of this is Media Access Control address. And this is also a unique for every machine whichever it is manufactured in till time has the unique number.

So, this MAC address is specified by 48 bits number, again each of 8 there are 6 8 bytes segmentation which is basically usually expressed in a hexadecimal number like, so usually max at this for example, I have given the MAC address of this one.

(Refer Slide Time: 11:23)

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

So, MAC address is also required again to identify uniquely a location of a particular machine. And then socket as the socket is basically is a is an application program, you have a software you can say is a port we can say which basically use for the communication, is basically is an application program which is responsible for connecting two endpoints.

So, if you want to connect two computers in a network, this means that that connection can be controlled by means of two programs at the two ends. So, two computer which should run two programs, they are called the socket programs as you know the socket is basically connected to a computer. So, every socket should be uniquely identified and that for this the IP address in combination with the port number both together call the socket address. For example, as you see the socket address of this machine is 10.14.90.85 and the port number is 27017 like. So, there so including these two things basically combined the socket address. And therefore, socket is not a hardware device, it is basically software.

(Refer Slide Time: 12:29)

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)*

DEBASIS SAMANTA
IIT KHARAGPUR

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

And now let us come to the communication protocol, there are many communication protocols has been proposed so far people you are using, like say UDP protocol it is that all connectionless protocol whereas, the TCP is very popular one protocol Transmission Control Protocol is called the connection oriented protocol.

And there is a another protocol, which is connection oriented as well as connectionless it is called the TCP-IP. In fact, IP is the one protocol just like a UDP it is called the connectionless protocol. So, TCP-IP is a combination of both the thing and FTP is a protocol, which is basically used for transferring the file from one machine to another.

And HTTP is the file to accessing web documents, like say HTML page like and it is the full form of Hypertext Transfer Protocol and HTTPS is the one it advanced version is basically Hypertext Transfer Protocol will follow the simple document transmission whereas, the HTTP is follows the secure document transmission as the security is there is basically security is enforced by means of some encryption and decryption. And SMTP is also just like a HTTP it is, but it is related to only application for the mail transfer. So, if you want to communicate your peers using mail and everything, so usually you follow SMTP for which you have to follow one mail server. So, mail server usually follow SMTP.

Now, here is again connectionless and connection-oriented, the two protocols by name you can understand that if it is a connection-oriented protocol means before the

communication, there should be the connection between the two computers should be established, so that is the one idea. There is a mechanism by which the communication or connection can be established prior to the communication begin. So, there is called the communication-oriented protocol.

On the other hand, connection-less protocol means, if the two party wants to communicate themselves, that they do not record any prior and at the prior connection to be established, they simply start sending their documents. Now, it is just like a sender sending a letter to his friend how he can send, you just simply write the postal address of the friend and then drop it to the post box, and then there is a postal service is there. They will channelize this letter into as a particular frame like. So, it is the procedure is called the connectionless protocol.

Anyway, so these are the protocol and more interesting fact is that Java supports any one protocol whatever we have mentioned, whether it is a UDP or TCP TCP-IP or HTTP HTTPS absolutely not the issues there.

(Refer Slide Time: 15:13)

Connection-oriented protocol

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

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

TCP follows connection-oriented protocol.

The diagram shows a network topology with a central cloud labeled 'IP/RATM Core'. Two 'IPsec/GRE and Sub' nodes are connected to the core. Each of these nodes is further connected to a 'VPN CE' node. A box labeled 'Connection-Oriented' is positioned between the two 'IPsec/GRE and Sub' nodes, indicating the path of the connection-oriented protocol.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

And so these are convection-less.

(Refer Slide Time: 15:13)

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**.

The diagram shows a network topology with an MPLS Core and Connectionless links between VPN CE and MPLS VPN nodes.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

And connection-oriented protocol, as you have discussed about.

(Refer Slide Time: 15:15)

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

And then TCP-IP is also one protocol which is basically more reliable as well as fast usually connectionless protocol is not reliable; however, it is fast whereas, connection oriented protocol is highly reliable, but it is not so fast. On the other hand, TCP-IP is a combination of the both connection-less and connection-oriented protocol which is reliable as well as very fast.

(Refer Slide Time: 15:41)

FTP

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.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

Now, FTP protocol is for the transferring file.

(Refer Slide Time: 15:43)

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.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

And HTTP is also transferring the hypertext document. For example, say document is an HTML page who is hosted some scraped, PHP, Java script or even a plate like this one. So, this basically how these files the documents can be coming I mean, can be from on server, can be brought to a browser machine here for maybe see your local machine and so and so on. So, these are protocol that it follows.

(Refer Slide Time: 16:07)

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 ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

SMTP is for transferring the file document.

(Refer Slide Time: 16:09)

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

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

And there is obviously, some salient points the between HTTP-FTP and SMTP, I have listed all these points here for the summarization of all these aspects you can take your own time to go through this and then can understand it.

(Refer Slide Time: 16:25)

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).

DEBASIS SAMANTA
IIT KHARAGPUR

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NPTEL

Now, I will come to that Java networking which is more important things to understand at the moment. And for the Java networking there are many ways actually and all the things can be categorized into two categories. They basically called the stream class and then the datagram sockets or stream sockets and datagram sockets, as you know socket means a program actually. So, the program should be written in Java language.

So, there are two ways the program can be written, one way is called the concept using stream sockets and another way is considering your datagram sockets. So, actually these are the two different programs, if we use the two different classes for communications among the two parties. Now, so far the stream sockets is concerned, it basically the related to the connection oriented service like say TCP.

ah So, according to these sockets it basically first establishes a connection between the two communicating parties. And then whenever the connection is established, then they will start streaming the data from one machine to another as a continuous that is why it is called the stream sockets. And here as I already told you, this is a connection oriented protocol that mean TCP is followed here.

On the other hand, datagram sockets is the another way of sending I mean communicating between the two parties, it is based on the connectionless protocol. Here actually what is that the entire documents or the content that needs to be channelized,

needs to be send from one into another is basically divided into small parts, each part is called the packets or more precisely it is called the datagram.

A datagram is just like a packets, like a packet consists of maybe say 64 bytes or each 1024 bytes the network admitters can decide that what should be the size, but the standard size is 64 bytes like. So, is a smaller bytes packets are created. So, entire documents if it is a 1 MB, it will be fragmented into 64 bytes each, each, each, each. And then it starts sending one bytes at a rate, and then whenever a packet is there or a datagram packet it should include; who is the sender and who is the receiver all at this everything is there.

Now, it starts sending start dropping this packet into the network. And then the network a controller is there, which will basically decide that whether if this packet is available to him, when to which direction he should forward this packet, so that ultimately it will reach to the destination ah points. So, this is the idea about this one and this protocol popularly called UDP – User Datagram Protocol.

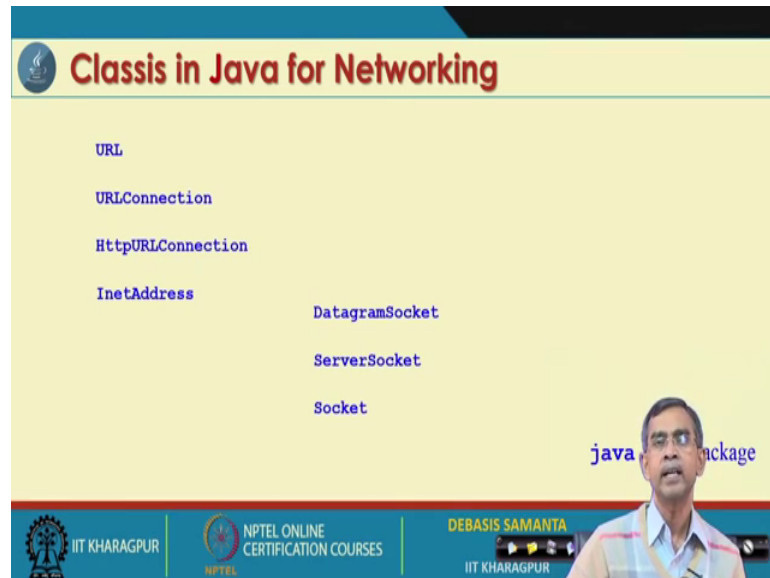
(Refer Slide Time: 19:07)

TRANSMISSION CONTROL PROTOCOL (TCP)	USER DATAGRAM PROTOCOL (UDP)
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.	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.
TCP is reliable as it guarantees delivery of data to the destination router.	The delivery of data to the destination cannot be guaranteed in UDP.
TCP provides extensive error checking mechanisms. It is because it provides flow control and acknowledgment of data.	UDP has only the basic error checking mechanism using checksums.
Sequencing of data is a feature of Transmission Control Protocol (TCP); this means that packets arrive in-order at the receiver.	There is no sequencing of data in UDP. If ordering is required, it has to be managed by the application layer.
TCP is comparatively slower than UDP.	UDP is faster, simpler and more efficient than TCP.
Retransmission of lost packets is possible in TCP, but not in UDP.	There is no retransmission of lost packets in User Datagram Protocol (UDP).
TCP header size is 20 bytes. TCP is heavy-weight.	UDP Header size is 8 bytes. UDP is lightweight.
TCP is used by HTTP, HTTPS, FTP, SMTP and Telnet	UDP is used by DNS, DHCP, TFTP, SNMP, RIP, and VoIP.

So, these are the two protocol that is follows here. And again there is obviously, some comparisons between the TCP versus UDP. The basic ways that TCP is connection yes or else the UDP connection-oriented, where is the UDP is connection-less. And there are some other more implicated the differences are there which has been listed here, again I advise you to just go through all the points and try to understand that is the difference is

there, but at the moment all those difference are not so much important. So, for our network each programming is concerned.

(Refer Slide Time: 19:41)



Now, I will come to the discussion on what are the different process so that we can utilize, because you know Java is the object oriented programming. So, for implementation of anything, we have to rely on the different classes. So, I will just discuss what are the different classes which basically you should learn, so that you can find yourself comfortable with the network related programming with Java.

Now, here there are few classes related the connection and then identifying a connection like. So, we have listed here like URL is a class, URLConnection is also another class, and then HttpURLConnection is a another class, InetAddress is class, we will go quickly go through the different class, they are compositions and then they are utilization and everything also illustration. And then so these are the different classes by which the connections related information's can be maintained or can be controlled.

And then regarding these connections and then making the communications actually, so that means sending and receiving and everything, it has been followed by few more classes, like DatagramSockets, ServerSocket and then Socket. Now, DatagramSocket is basically the concept that followed is a datagram streaming that means for UDP protocol like, where as ServerSocket and Socket is basically for the TCP related protocol that means, connection oriented protocol like.

So, ServerSocket is basically whenever we have to make one machine as a server and then socket is basically for making some machines as a client. So, it is basically for client server computing or distributed computing line. Anyway we will discuss the different ah concept of all the classes that is basically very much essentials here, relate to the Java networking program.

Anyway all these classes definitely are Java is very organized on system, they are defined in a package. The package, which basically used for this is Java dot net dot package. Java dot net dot package is basically is the package mean for including all these classes, where user can get it and then you can use it.

(Refer Slide Time: 21:49)

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

Now, I will first discuss each one class one by one followed by some illustration, so that you can understand the concept that what are these classes has these facilities and in what occasion which class should be used like this. So, the class URL and I know you have already mentioned that what exactly and URL look like. So, URL first point is that which protocol that it follows.

For example, in this example as you see there are two parts in this URL, the `https` is the protocol that it should follows. And then followed by the what is called the resource location, this resource location is basically `nptel dot ac dot in slash course dot PHP`, this means that in this location, there is a file available means a user, a browser, a party can access on tile the name of the file is `course dot PHP`. So, the `course dot PHP` can be

accessed by means of this URL and here another thing is that from which machine that this file can be accessed.

So, the middle part that is nptl nptel.ac.in indicates in which server, this file is located. And then if this file and this server is identified, then they are definitely what type of protocol transmission should be followed in order to get access of this file into your own machine. So, here therefore the protocol, the server name and sometimes the server name should be associated with port number also. For example, nptel dot ac dot in and then colon and the port number say 5 0 5 0 that also can be specified; if no port number is mentioned, then by default the port number will be considered as a minus 1, so this is the default standard and like this one.

And then finally, the last component is the actual document specialist at the document. Like say if we say there, test dot Java. So, this is the file that you can have and then we can access it. So, it is like this so this way the URL basically imply it, now whenever we can define a class. So, a URL basically this is an object as you see, so the this is basically an object and we can say that URL object that means, URL consists of so many information; what is the protocol, what is the name of the server, what is the file name or maybe. So, this is the name of the server that we can understand, but in new of this on the server name can be also uniquely identified by IP address say 155.18.20.31, so that is also this one, so logically address is there.

So, it is in and there and there may be one machine that proxy we can say, proxy has the tank mapping that if this is the name domain name or the server name, then corresponding this which is the IP address. So, there is a table maintained for every network there is a proxy machine, which basically includes everything if this is the domain name then what is the IP address like this one.

So, so this is a very important one concept about regarding the location or a precisely specifying the machine that is there. So, this basically the URL location and the URL classes is basically has many other constructors as well as method is there.

(Refer Slide Time: 25:27)

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.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

Ok.

(Refer Slide Time: 25:37)

Class URL : Methods

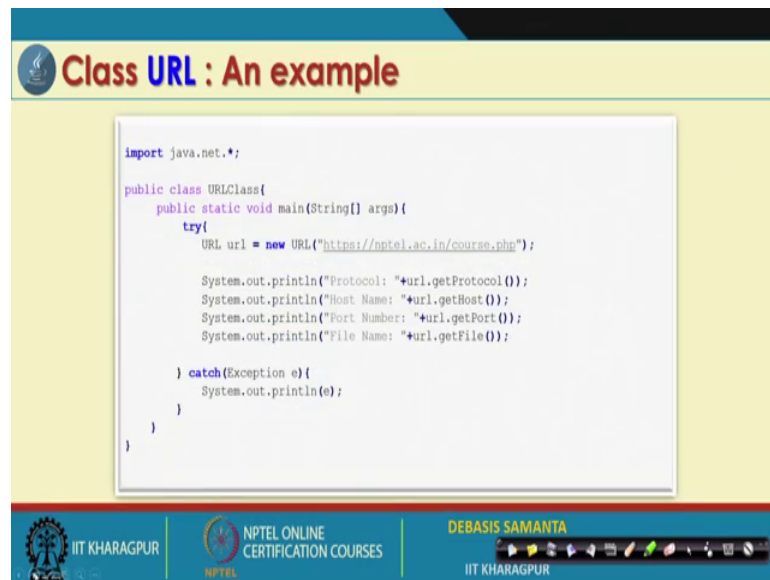
The `java.net.URL` class provides many methods. The important methods of URL class are given below.

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 | IIT KHARAGPUR

So, the methods those are they are in URL by there are few more methods also, but I have mention only those are most important methods are there; `getProtocol`, `getHost` that means, the name of the host and then `getPort`, `getFile`, and then `openConnection`, like this one. So, I have already mentioned at the about the URL. So, if it is URL object is given to you, and then you can get although the information from there.

(Refer Slide Time: 26:05)



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");

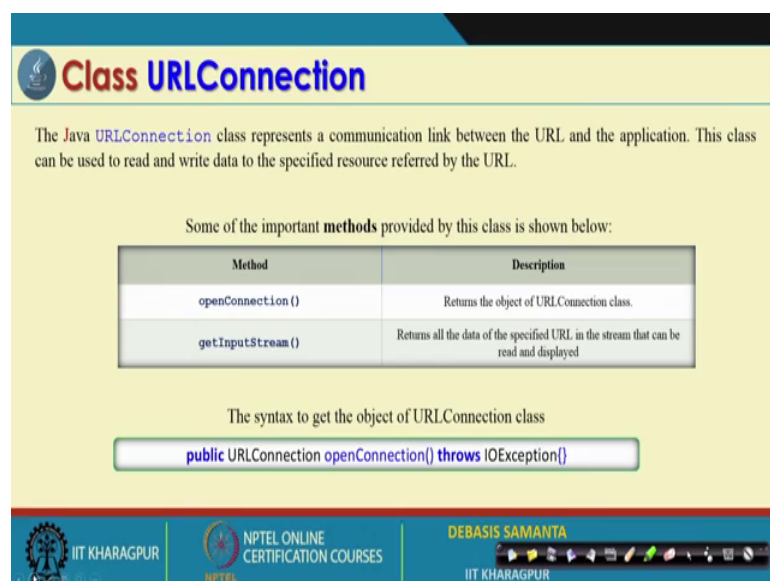
            System.out.println("Protocol: "+url.getProtocol());
            System.out.println("Host Name: "+url.getHost());
            System.out.println("Port Number: "+url.getPort());
            System.out.println("File Name: "+url.getFile());

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

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

And so this is an example you can understand, this is very separating a simple example. So, we can create an URL object like this one, sending this information. And then from this URL class and using this getProtocol method that we have already discussed, we can have all the information related to this object. So, this is a simple example if you can run it, you can get an idea. So, instead of any other URL you can mention any URL of your own and you can find the execution of this thing you can understand about these one. And so this is the URL.

(Refer Slide Time: 26:41)



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 <code>URLConnection</code> 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

And then the next class is called the URL class, URL connection class. It has the two methods mainly, the open connection and get input stream and this open connection is basically to establish a connection from one machine to another machine from the current socket to the distant socket like. And then get input stream basically, how the data can be read from the distant socket or from the another socket to this current socket. So, this is the two methods usually very much useful, there are some other methods also those are very much essential right, but those are the methods we have we will be referring here, we have mentioned here only.

(Refer Slide Time: 27:15)

```
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);
        }
    }
}
```

And this is an example that you can check it quickly. And so this is basically URL object is created, and then we just make a connection and this is the method by which we can make the connection. So, this name of the connection, connection again is an object everything in Java theater engine objects. So, URL connection and then once the connection is there, then we can create a stream and this stream is basically get input stream for this URL connection that means, if this is the machine and we can connect this another machine like right from so this is basically the distance machine.

So, this machine address is this on, it is your own program from where I am running this one, so this one. And then from there, I am get a get input stream means I am reading the data from this machine. So, this is the idea about each here that and then we can start reading this data in the stream ah. So, long the end of file is occurred that mean entire

file will be read from the distance machine and will be stored into the local machine like. So, this is a very simple example that explain these concept that how the URL connection can be utilized to fetch data from the distance machine through the communication.

(Refer Slide Time: 28:21)

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 | IIT KHARAGPUR

Now, `HttpURLConnection` is very similar to URL connection only, but only in this case the protocol, that it should follow HTTP protocol only.

(Refer Slide Time: 28:29)

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
<code>getHeaderFieldKey(int n)</code>	Returns the information in the header field <code>n</code> .
<code>getHeaderField(java.lang.String name)</code>	Returns all the header field.

Note: There are many more methods in this class

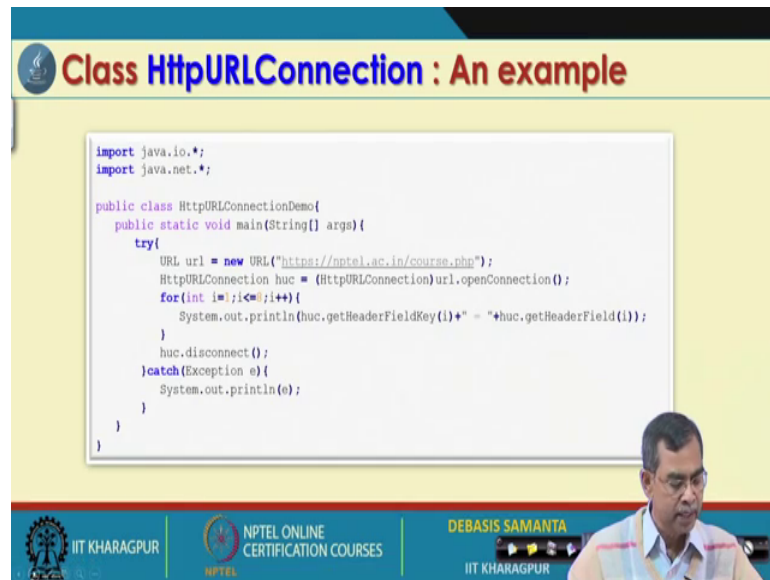
The syntax to get the object of `URLConnection` class

```
public URLConnection openConnection() throws IOException;
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

And here is a rather different methods that is included in the HttpURLConnection, the huge number of methods are there, but I have mentioned only two methods other methods you can obtain from the core Java material, tutorial that is there.

(Refer Slide Time: 28:41)



The slide displays a Java code example for the HttpURLConnection class. The code is as follows:

```
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);
        }
    }
}
```

The slide also features the NPTEL logo, the text "IIT KHARAGPUR", "NPTEL ONLINE CERTIFICATION COURSES", and the name "DEBASIS SAMANTA" in the bottom right corner.

Now, let us come to the example here as you see, so this is the url connection, url objects. And these basically the idea about, how the HttpURLConnection can be established. And once these connection is established rest of the things is same as the previous one. So, this is the idea about so only these basically make distinguish or simple communication or the URL HTTP communication like.

(Refer Slide Time: 29:07)

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 www.google.com 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

And InetAddress also similar to the URL address, but it has the different way the internet address related composition. So, he is basically IP, Port, everything can be there, it can be understood. And as you have seen these are the different methods is very similar to the URL like, the different methods are there by which you can access the different information related to the different connection of a connection with the about the different connection.

(Refer Slide Time: 29:33)

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);
        }
    }
}
```

`F:\WORK\DP\aii Final\week 10\Lecture Slides\Networking Demos\4. InetAddress Class\java InetAddressClass`
Host Name: www.nptel.ac.in
IP Address: 14.139.169.71

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA

And here is another example that you can see. So, InetAddress just like a similarly URL, this is basically the URL we can say I have specified and get by name and InetAddress. So, IP is basically the I InetAddress in this case, one this InetAddress is there, then we can have the different method getHostname, getHost local host and everything of this method. And then we can obtain the different information, the concept is very similar to the URL connection and the idea it is like this. And here is an example that you can run this program.

Now, I will come to the communication. So, using the sockets basically and there are two type of sockets, rather 3 socket DatagramSockets, server socket and socket only.

(Refer Slide Time: 30:17)

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

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

So, DatagramSocket is basically follow the UDP protocol and the different methods those are defined there in DatagramSockets, these are the basically constructor by which a socket can be created actually that they are different constructor means, I can either we can mentioned the port number or we can means at the port number as well as either at this or you cannot mean some anything. So, this socket is basically if you want to communicate within the same local host that means, maybe that within the same machine like one machine can be connected by the two users.

So, it is basically shear single shear machine is sheared form and then there protocol can be followed, but here if the distance machine from this machine a to another machine b.

And it is also distance machine, but if you know the InetAddress quickly early. So, these are the different I mean constructed those are useful for the DatagramSocket like.

(Refer Slide Time: 31:13)

Class DatagramSocket : Methods

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

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

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

And these are the methods `getAddress`, `getPort`, and then `getData` and `getLength`. `getData` is very important in order to get the data I mean ah get fetching the data from the machine to that machine here.

(Refer Slide Time: 31:27)

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();
    }
}

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.getByHost("127.0.0.1");
        DatagramPacket dp = new DatagramPacket(str.getBytes(), str.length(), ip, 3000);
        ds.send(dp);
        ds.close();
    }
}
```

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.

Receiver

Sender

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA

And this example is very easy to understand and very simple example I have planned for you. So, in this example as we see DatagramSockets ds object is created, and here you

have to mention the port that is the port number which we and this machine this example is basically, in the same machine how it can be shared the documents actually. So, the idea it is there you can run the two programs, socket program so that means, two different threads are there.

And that two threads is just you can simulate that ok, this is the one party and this is the another party; like so one party is called the receiver, another party is called the sender. And this is a receiver socket and this is the Sender sockets. Idea is very simple, first you have to keep create a DatagramSocket objects as you have created here. And then to do this thing, I have mentioned here port number. So, port number should be same as these two machines, these two programs who run in the same machine. So, for both server and send receiver and sender it has the same port number.

And then here you just create a DatagramPacket. So, DatagramSockets and DatagramPackets are basically to see that whatever the message that you want to communicate, how the entire message can be created into a number of packets, a packets may be of 64 bytes as I have told you. So, these object we will do these things for you. The entire buffer if you give it, then this dp will create the packets that needs to be transmitted from one party to another here.

And then once it is there, then this basically dp dot getData can be converting the string and the same string can be transmitted by this means of method. So, receive the dp that means, it basically receive packets that can be a send that that has been send by some machine and it will receive. And then after receiving this packet dp, it will basically process that means, you can process shown or you can display whatever it is there.

So, this is so further receive is concerned. And now here is basically sending, as you know the sending here. So, suppose I want to send only this text, Welcome to a NPTEL. So, first I have to create a DatagramSocket ds, as you have to send in the same machine. So, port number is basically same thing that you can do, otherwise you can do not use anything also. And then this is the message that I want to send it, is very small message of course in this case, but actually it can be very large file in that case I can mention that this is a file that you have to store, but using say file input stream the file can be converted in the stream format and that stream can be putted into that datagram like.

And then this is the ip at this from here you have to send it. Now, here is the one thing is that or it is a default that if the same machine it will already assign one ip address that it is this one 127.0.0.1, it is the standard default only. So, this is the default ip address if you have to communicate within the same machine only. And then the DatagramPacket that will be transmitted about this string. So, it is a string we convert in the DatagramPacket, and then we send this DatagramPacket to this port and then is basically send.

So, the DatagramPacket has been ready with these destination address and everything and therefore send. So, this way the sender will send, it will go to the receiver, receiver will receive, and then it will be process. So, this is basically the communication between the two parties, which are basically using the same port we can say. Same port in the sense that they are the same machine.

(Refer Slide Time: 35:01)

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, you must call the <code>bind()</code> method when you are ready to bind the server socket.

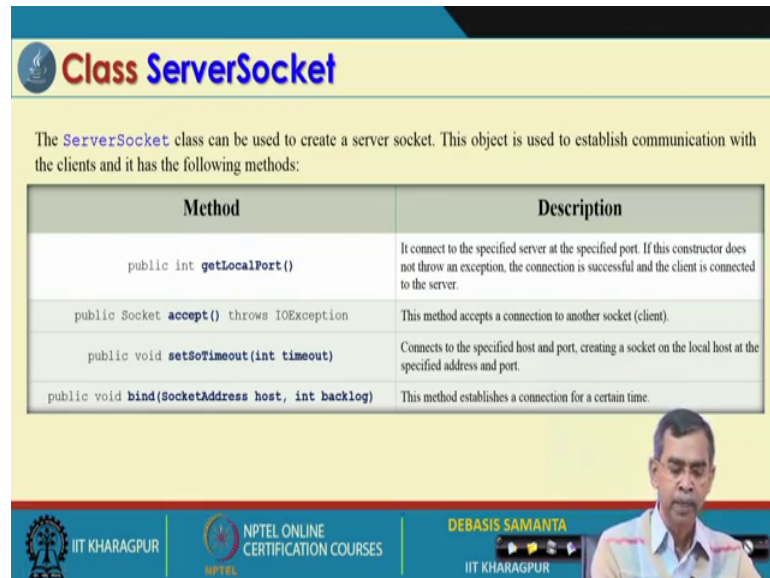
DEBASIS SAMANTA
IIT KHARAGPUR

Now, so another is a class ServerSockets, these has the few constructor as we have mentioned here. Again the port, the InetAddress and then time that needs to be queued and so many things are the basic information that needs to be provided in order to create the object.

As I have already mentioned you, the server socket is basically the program that basically suitable for maintaining one machine as a server that means, this machine may be connected with many clients, those are either in the same or you may same system or

maybe in the remote systems whatever it is here. And so idea it is like this, so server socket is there. And the constructor it is these are the constructor.

(Refer Slide Time: 35:41)



Class ServerSocket

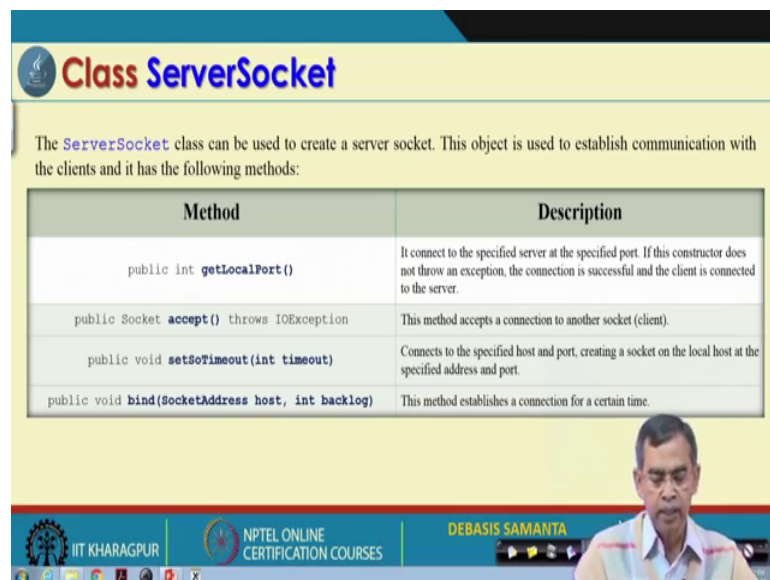
The `ServerSocket` class can be used to create a server socket. This object is used to establish communication with the clients and it has the following methods:

Method	Description
<code>public int getLocalPort()</code>	It connect 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.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

There are many more methods are there, I have mention few methods here.

(Refer Slide Time: 35:49)



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 connect 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.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

I have mention few methods, those are very important here. For example, getting the information about the port, and connecting that accept I mean accepting a communication connection, and then how much time that it should way to listen to

others, those are the different information which is basically can be maintained by and then can be obtained from the from this method for a ServerSocket object like.

(Refer Slide Time: 36:19)

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 connect 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 InetAddress 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 InetAddress object instead of a String
<code>public Socket()</code>	Creates an unconnected socket. Use the connect() method to connect this socket to a server.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA

And then Class Sockets, very similar to this server socket itself only the thing is that socket can be used to represents a client machine, so that is why and it has the different constructor as I have mentioned here. And in addition to these constructors, there are many other methods also involved.

(Refer Slide Time: 36:37)

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

So, these are the methods very similar to the server sockets. And then I will quickly conclude this presentation with more few examples, 2 3 examples I will give it. So, simple client server.

(Refer Slide Time: 36:49)

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);
        }
    }
}
```

Let's see a simple 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.

Server

Handwritten notes: 'S.S.W' and 'S.S.' with arrows pointing to the ServerSocket and Socket lines in the code.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA

Example, I have already given an idea about the sender and receiver. It is very similar tune only, how this program basically explained, how one machine can be made as server, another machine can be made as a client. So, the here basically the ServerSocket to be created and here the socket is created. Once it is created is basically by means of getInputStream, it will read it; and by means of getOutputStream, then it will basically send it. So, these are the two concept by which the communication can be establishes. So, this is the idea about the communication here.

Now, the server as you see in this program we have created a socket as a ss and ServerSocket ss also. Now, here ss this basically we create ss dot accept means, the server accepts any response that is basically made by on a request that is made by a client, so that it will accept and then the DataInputStream is a class by which it will read data from the client actually. So, it is a right s is the socket program in the client and ss is the socket program in the server in this case and then it will basically send. And finally, when the message receiving is complete, so it is basically closed. So, this is the idea it is there.

(Refer Slide Time: 38:11)

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

And then once it is this is the server program and similarly the client program it looks like, so this is the client program. And in this client program, we create a socket program here, so s is the socket objects. And then DataOutputStream, because here client we will send some data to the server, and this is the message that the client suppose to send it. And this basically message will be transmitted, I mean will be send to the net and then it will go to the sockets and then this one. And here is the port number, this port number should be the port number of the server to which it wants to communicate actually. And it should be same is in case of both server and then socket as you see here, the port number that we have used here.

(Refer Slide Time: 38:53)

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 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

Is basically 6 6 6 also, it is basically in the server. Now, so this is the idea about that ok, here in this particular example how a client machine can send a document to server, and then server can received it and process it.

And our next example is basically to give the dialogue that means client will send something, server responds to that; again client will send something, server will respond to that this is called the dialogue process. And this is again simple program to understand this concept.

(Refer Slide Time: 39:25)

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("#")){
            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.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

And here is basically the server is basically this is the server, and then server as the ServerSockets and these basically accept and this is basically input, output mechanism that is read from the client right through the client like this one. So, for which data input and DataOutputstream, d in and d out has been created, and then it will be for reading if the large message needs to be read. So, data to buffer reader should be used, so it is a buffer reader and from the buffer reader it basically is reading, and whenever there is a message that from the client that is a stop, then it will close the service, then server will stop it there.

So, this is the server side program. Is very simple program actually as you see, and you can understand easily just you have to remember the input output stream, we are once we have discussed about our IO stream concept like.

(Refer Slide Time: 40:17)

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();
}}
```

This is the client program.

Client

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES DEBASIS SAMANTA

And this is the client side program. In this client side the same port number it should use. So, it is based socket and the local host means is the local actually in the same machine. So, in the same machine two programs we will learn. So, it is quite possible multiple program can be executed in the same machine, Java allows these things very heavily, so it will there.

And then data this is basically input output that means, client will send something, client will receive something from the server. So, this one again the buffer will be there, to read the network line actually channel, and then this is the communication continue until the

server wants to continue it there. So, this is the idea about the dialogue client-server as you have so it is a basically dialogue between the two machines. And this is the one example here, in the last example we have considered that in the same machine, but in the in this example if the two distance machines are there.

(Refer Slide Time: 41:09)

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(1333, 5, InetAddress.getByAddress("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**.

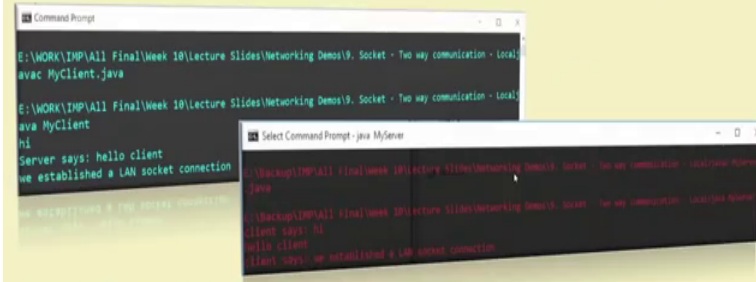
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

So, obviously you should know exactly what is the IP address of the distance machine. The program is very similar to the previous one, only the difference is that we have to explicitly mention, the IP address and then these are port number and number of client that it can support in this case. So, this is the only change in the server program, otherwise everything remains same. The client program particularly is the same program as it is the earlier one.

(Refer Slide Time: 41:35)

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.*



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

So, this is the client program already we have discussed and you can run it, we can simulate it in the same machine or in at the distance machine also the output that you will can obtain. And then again the communication, dialogue, chat, and everything whatever you want to do, you can do that. So, this is basically the remote communication between the two machines.

(Refer Slide Time: 41:55)

Concurrent server : An example

```
import java.io.*;
import java.net.*;
import java.util.*;
// Server class
public class MyServer {
    public static void main(String[] args) throws IOException {
        // server is listening on port 8080
        ServerSocket ss = new ServerSocket(8080);
        // 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();
            }
        }
    }
}
```

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.

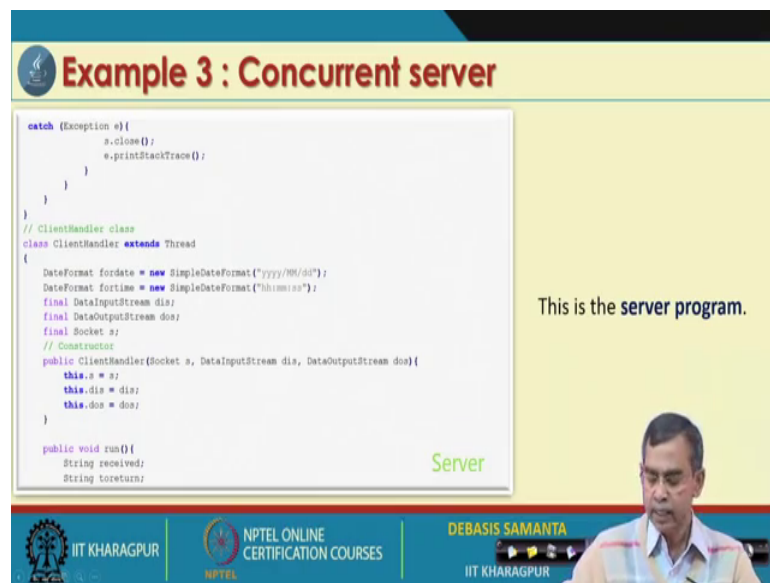
This is the server program followed by the client program.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA | IIT KHARAGPUR

And then concurrent server is basically one server can process more than one client at a time, so it is called the concurrent server. In this case, the threads to be maintained and if

there should be some method should be by which the client handling client can be handled like. So, here again this is an example, so this basically creating the server as a socket and for this server socket we accept any client that it send the request, and so this is the same thing as earlier.

(Refer Slide Time: 42:25)



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;
```

This is the server program.

Server

DEBASIS SAMANTA
IIT KHARAGPUR

IIT KHARAGPUR
NPTEL ONLINE
CERTIFICATION COURSES
NPTEL

And then the next thing is basically here is the thread that you have to create it. And for this thread, this is the one simple intermediate procedure about input output stream that needs to be created. And we create a thread here; this is because you have to establish a thread there. And this basically ah so thread is basically here, the implementation of thread method in the last example we created thread actually.

(Refer Slide Time: 42:59)

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 5054
        ServerSocket ss = new ServerSocket(5054);
        // 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();
            }
        }
    }
}
```

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.

Server

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES DEBASIS SAMANTA

So, here we see we create a thread here, so a server is a thread. It will run for so long, and the thread needs to be handled by the ClientHandle by means of input stream and output stream, those are things to be mentioned there. So, this is basically created thread. So, in this case the server is a thread and we start the thread. Once the thread is started, we have to define the thread.

(Refer Slide Time: 43:23)

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;
        }
    }
}
```

This is the server program.

Server

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES DEBASIS SAMANTA

And thread define means all the method you have to define it here. And the run method is basically is defined the thread how the thread needs to be run actually. And here is the

server part next part of the server; that means, we just continuation of this one while is basically this is a infinite execution, while it is true like.

So, until you do not at a disconnect the connection, it will continue this one. And then it will receive the message, send the message, all these things will carried out in a synchronized manner actually, so that program is like this, so that concurrently that means it does this basically constantly attend attending on client request if it is there; once it is there, it responds immediately for which the thread is basically mean for that.

(Refer Slide Time: 44:11)

Example 3 : Concurrent server

```
        case "time" :
            toreturn = foctime.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

And here this is the continuation of this program.

(Refer Slide Time: 44:13)

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);
            }
        }
    }
}
```

This is the client program.

Client

DEBASIS SAMANTA
IIT KHARAGPUR

And the client program as usual the similar one, absolutely there is no same where different. It is basically same, we have to make this as connection as a 5 0 5 6 is the port number of the server itself, and then input output stream for the dialogue communication if you want to maintain. Then it basically than any other protocol say maybe say, here we are using the ip i InetAddress, basically TCP-IP protocol, but you can mean some the datagram protocol dp and you can use it.

So, in that case of message needs to be first packet it using the data gram packets and it can be sent here. So, this is the idea about the concurrent servers both in the client side and then server side. And this basically explained the different mechanism by which the communication can be done and as in the class of learning right. So, we are basically learning, how the communication can be written using Java program, so that is the only focus we have. Now, in our next we will discussed the some demonstration of this program, so that you can see exactly how they run and then how they can execute, and then how they can see the result actually.

So, we will discuss about the demonstration of this program in our next module that we will give an idea about, how we can implement all these programs and then you can execute it and you can see the output.

Thank you very much.