

# Unit-1: Java Networking

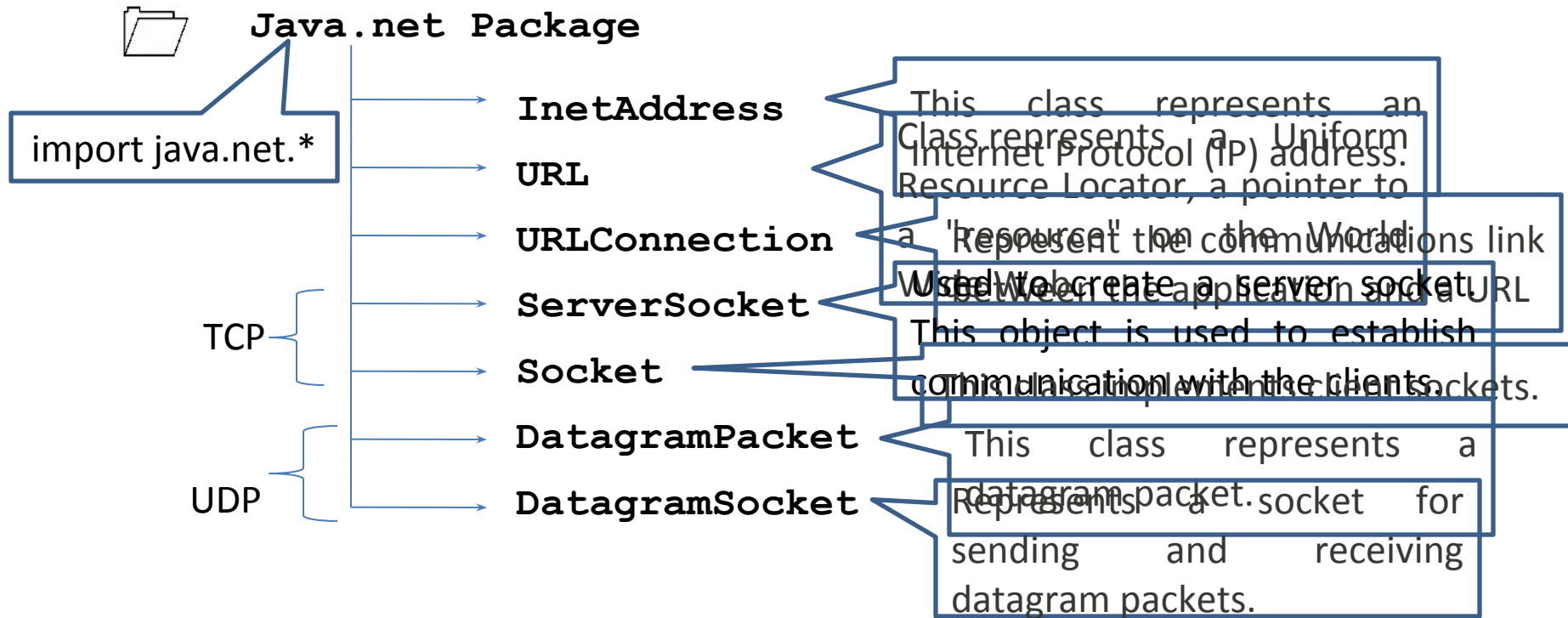
---

## Topic

1. **Network Basics and Socket overview**
2. InetAddress
3. TCP/IP server sockets
4. TCP/IP client sockets
5. Datagrams
6. URL
7. URLConnection

# Network Basics: java.net package

- The term *network programming* refers to writing programs that execute across multiple devices (computers), in which the devices are all connected to each other using a network.

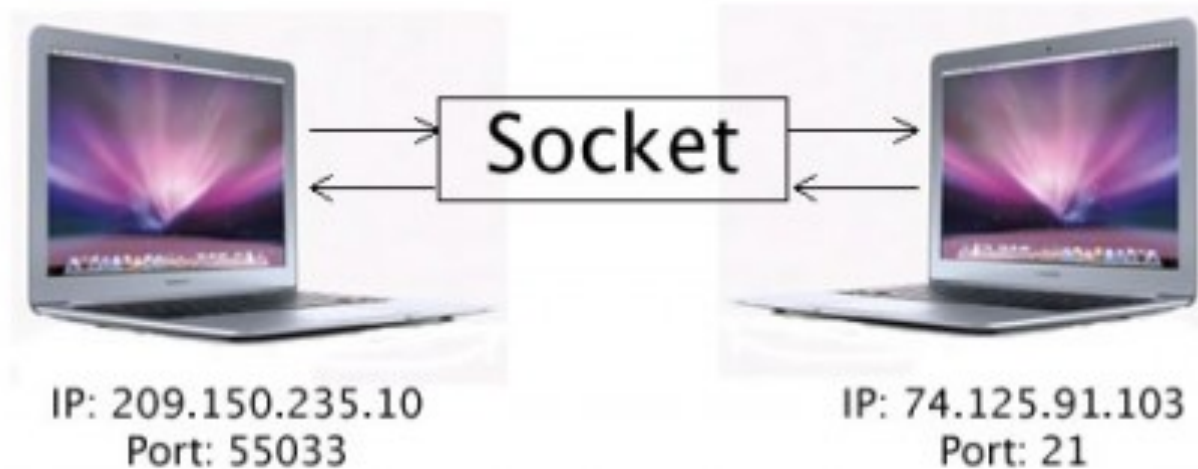


# Socket overview

---

## Socket

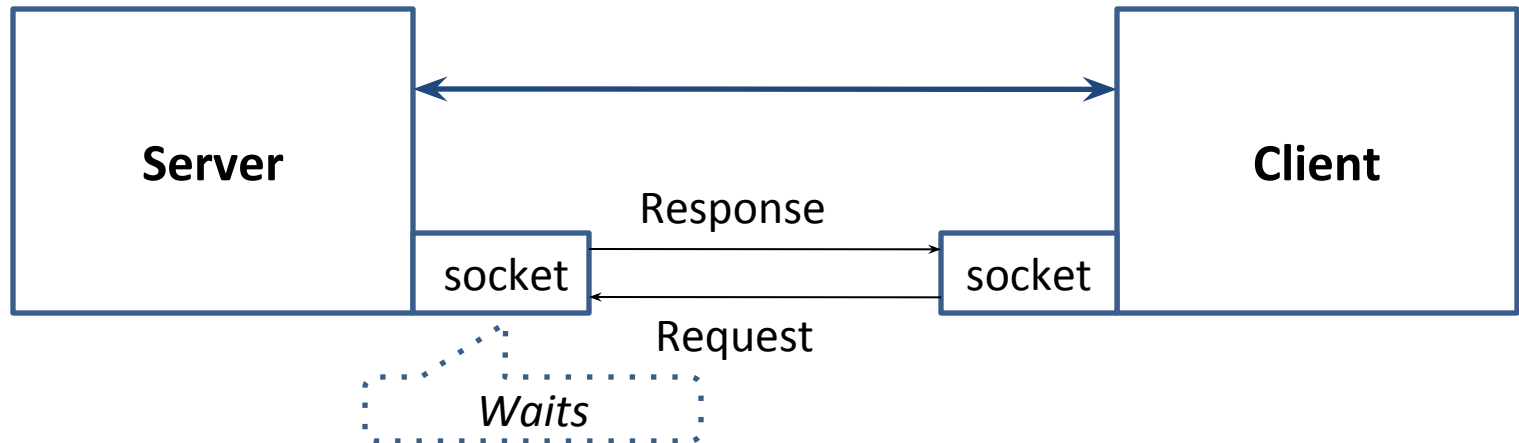
- “A **socket** is one endpoint of a two-way communication link between two programs running on the network.”
- A Socket is combination of an IP address and a port number.



# Client – Server Communication

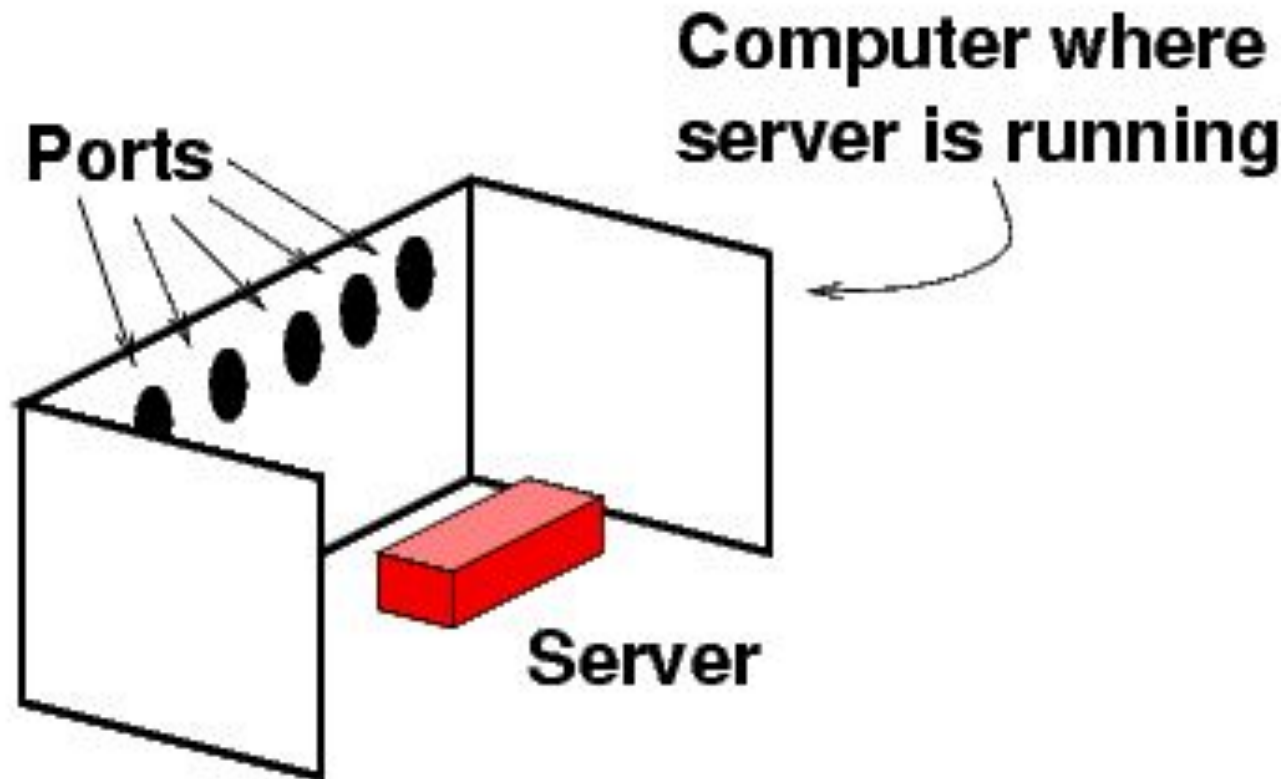
---

- Two machines must connect
- Server waits for connection
- Client initiates connection
- Server responds to the client request



# Socket overview

- The server is just like any ordinary program running in a computer.
- Each computer is equipped with some ports.

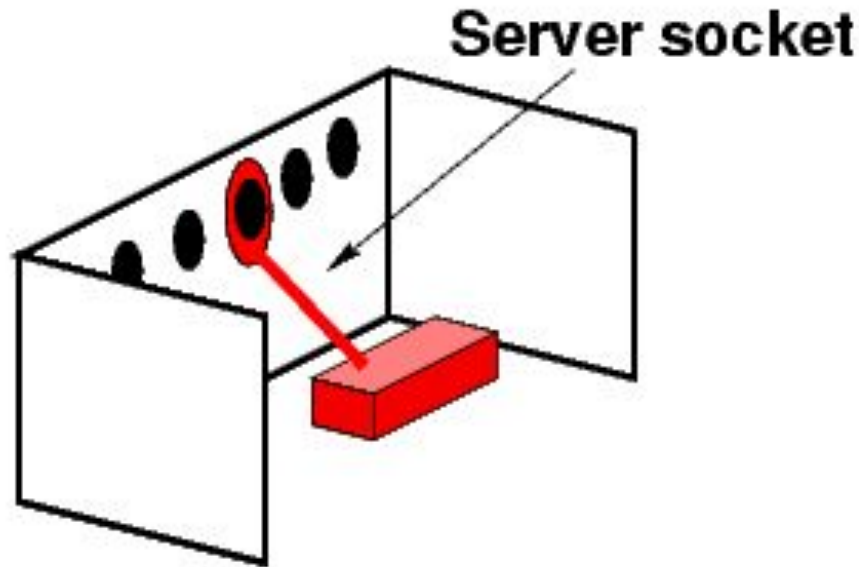


# Socket overview

- The server connects to one of the ports.
- This process is called **binding** to a port.
- The connection is called a server socket.

*The Java server code that does this is:*

```
ServerSocket ss = new ServerSocket(1234) ; //1234 is port number
```

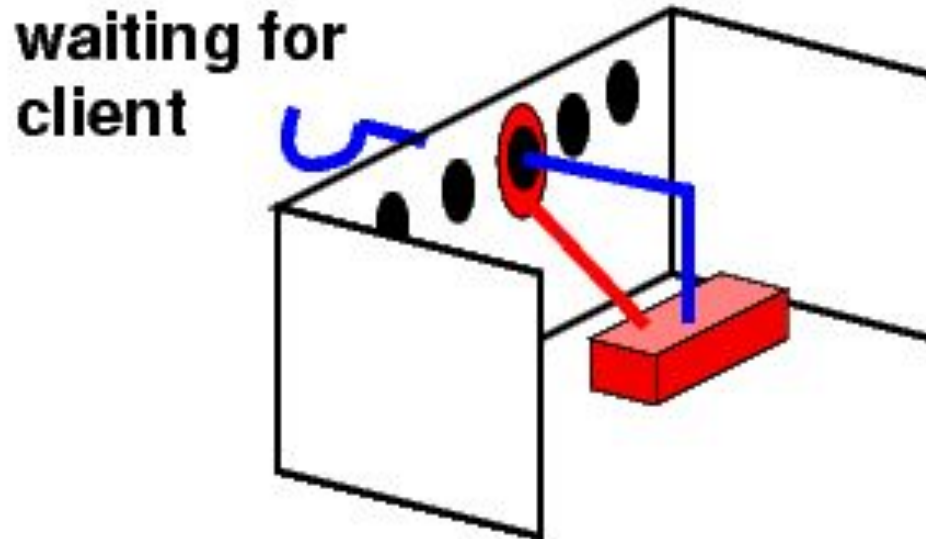


*Reference: isinotes/javatut/net*

# Socket overview

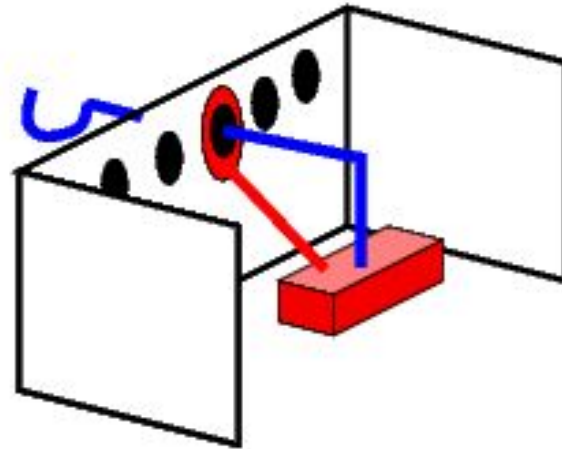
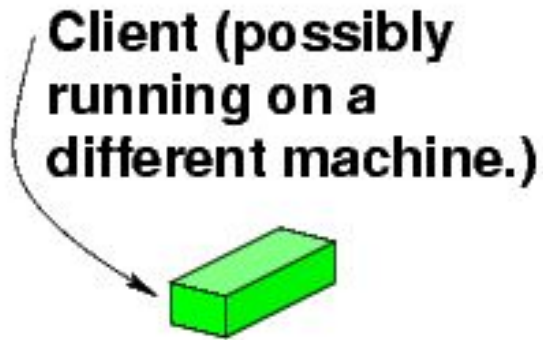
---

- Server is waiting for client machine to connect.



# Socket overview

---

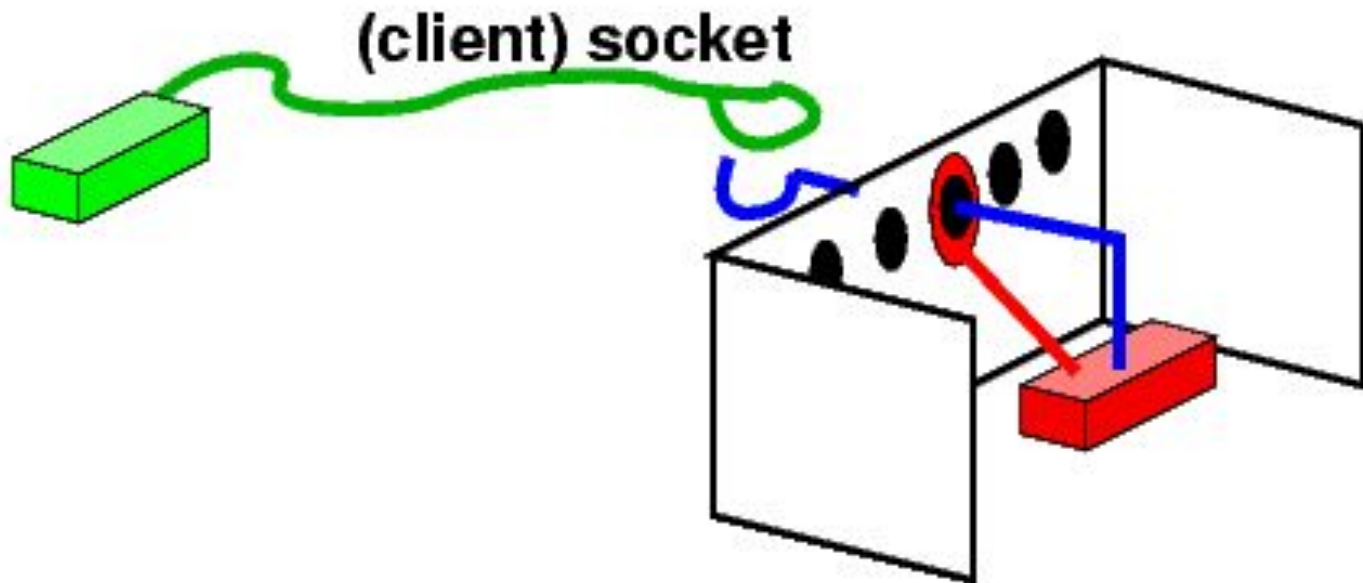




# Socket overview

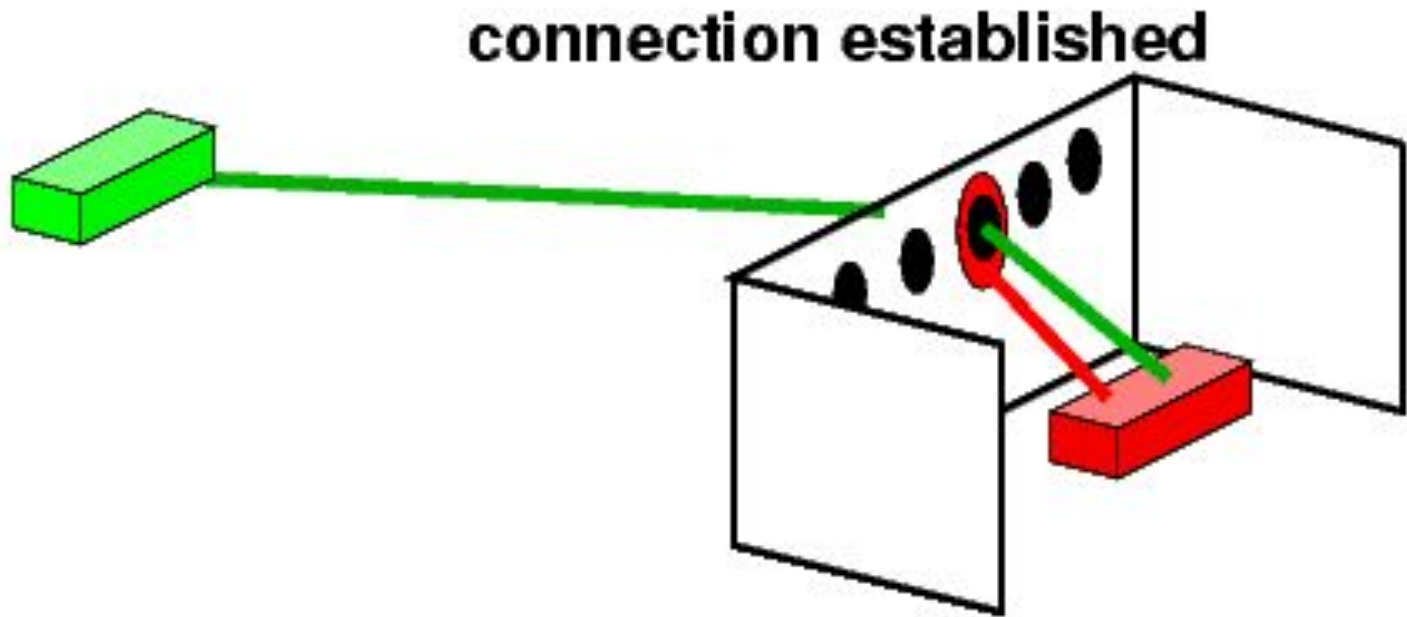
- In the next step the client connects to this port of the server's computer.
- The connection is called a (client) socket.

```
Socket sock = new Socket("www.indusuniversity.in", 80);  
/* The client knows the number 80 */
```



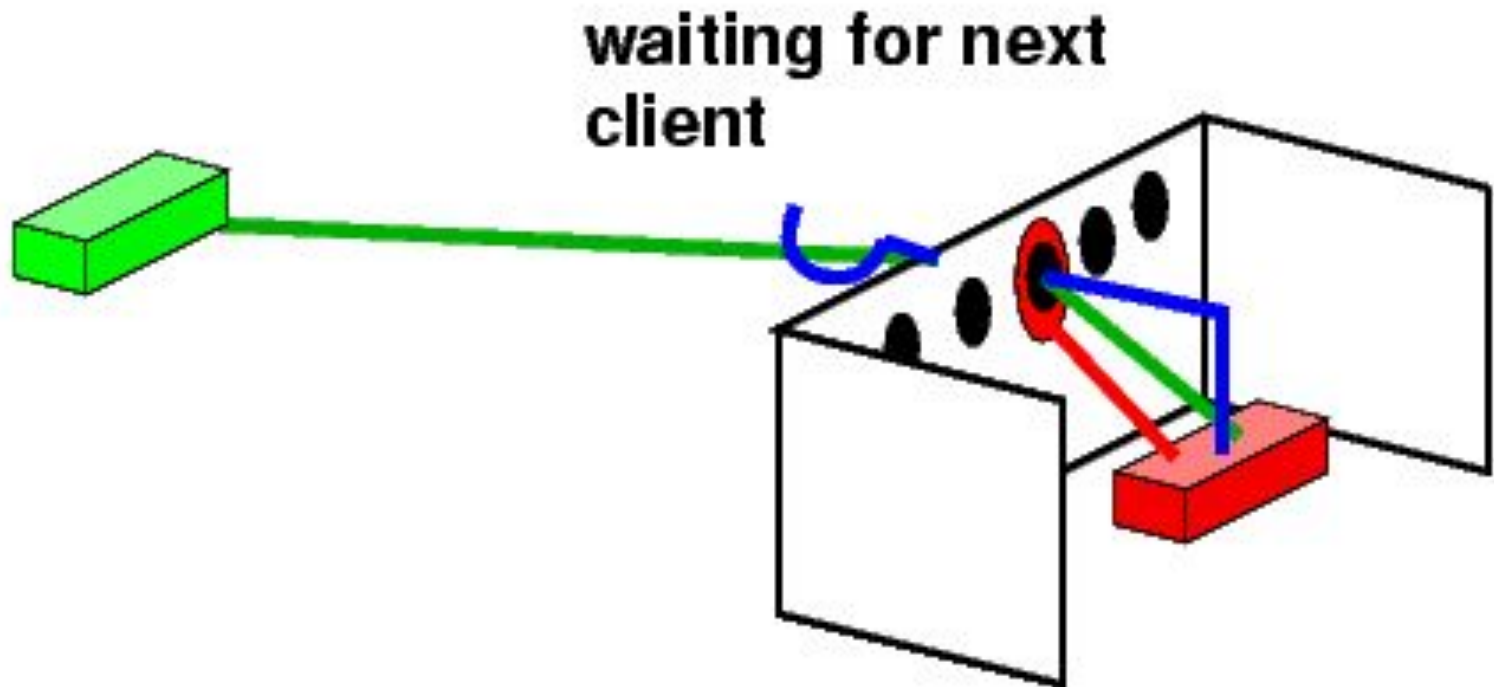
# Socket overview

- Now, connection is established between client and server.



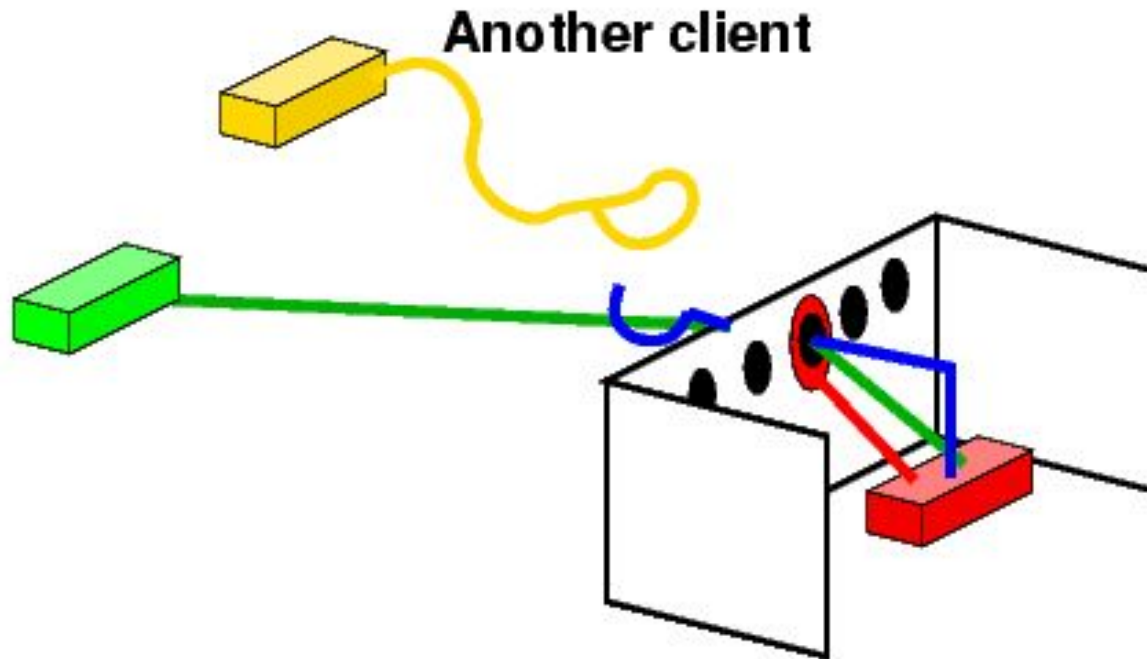
*Reference: isinotes/javatut/net*

# Socket overview



# Socket overview

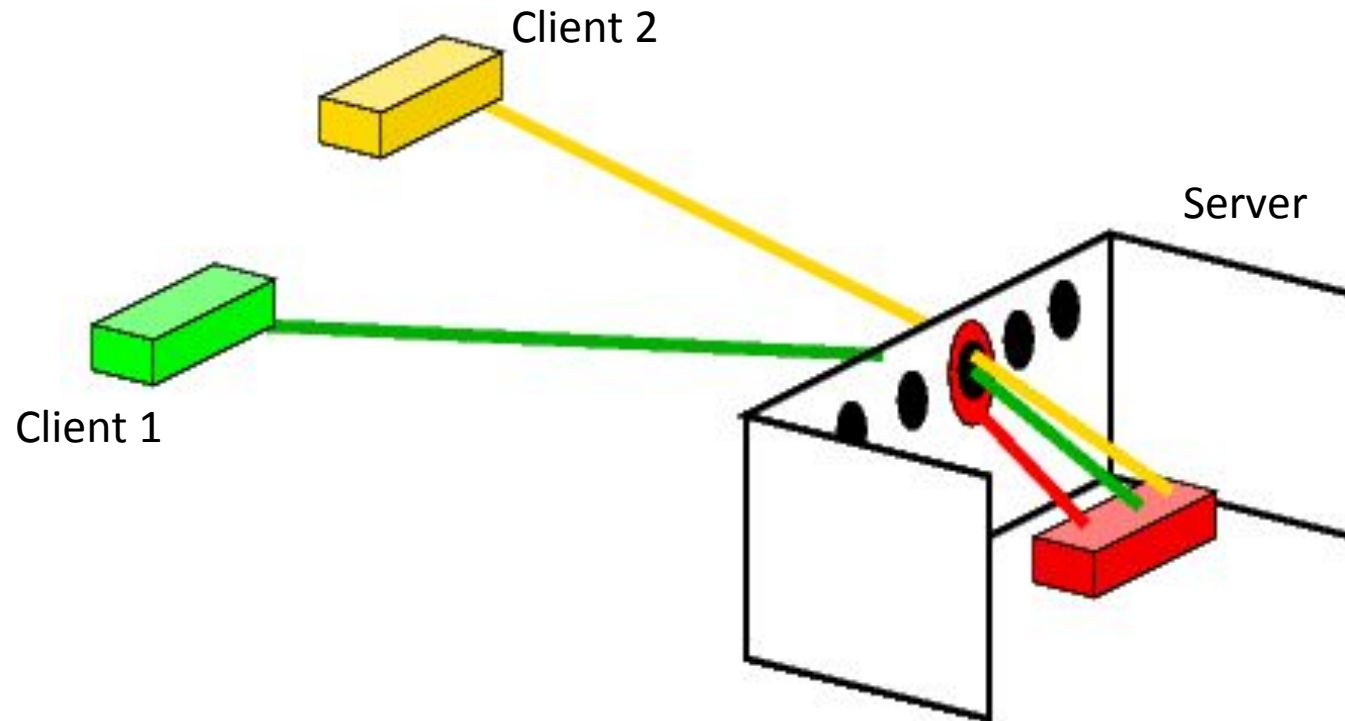
Everytime a client is found, its Socket is extracted, and the loop again waits for the next client.



Reference: [isinotes/javatut/net](http://isinotes/javatut/net)

# Socket overview

---



*Reference: isinotes/javatut/net*

# Unit-1: Java Networking

---

## Topic

1. Network Basics and Socket overview
2. **InetAddress**
3. TCP/IP server sockets
4. TCP/IP client sockets
5. Datagrams
6. URL
7. URLConnection

# InetAddress

---

## java.net package

- This class represents an Internet Protocol (IP) address.
- The **java.net.InetAddress** class provides methods to get an IP of host name.

### *Example*

```
InetAddress ip  
=InetAddress.getByName("www.indusuniversity.in");  
System.out.println("ip="+ ip);
```

### *Output:*

```
ip: www.indusuniversity.in/89.238.188.50
```

# InetAddress : Method

Method	Description
<code>public static InetAddress <b>getByName</b>(String host) throws UnknownHostException</code>	Determines the IP address of a given host's name.

*Example*

```
InetAddress ip  
=InetAddress.getByName("www.indusuniversity.in");  
System.out.println("ip="+ ip);
```

*Output:*

```
ip: www.indusuniversity.in/89.238.188.50
```



# InetAddress : Method

Method	Description
<code>public static InetAddress <b>getLocalHost()</b> throws UnknownHostException</code>	Returns the address of the local host.

*Example*

```
InetAddress ip=InetAddress.getLocalHost();  
System.out.println("LocalHost:"+ip);
```

*Output:*

```
LocalHost:milan-PC/10.254.3.34
```

# InetAddress : Method

---

Method	Description
public String <b>getHostName()</b>	it returns the host name of the IP address.

## *Example*

```
InetAddress ip=InetAddress.getByName("10.254.3.34");  
System.out.println("Hostname:"+ip.getHostName());
```

## *Output:*

```
Hostname: milan-PC
```

# InetAddress : Method

Method	Description
public String <b>getHostAddress()</b>	it returns the IP address in string format.

## *Example*

```
InetAddress ip=InetAddress.getByName("www.indusuniversity.in");  
System.out.println("HostAddress:"+ip.getHostAddress());
```

## *Output:*

```
HostAddress:89.238.188.50
```

# InetAddress: Program

---

```
import java.net.*; //required for InetAddress Class
public class InetDemo{
    public static void main(String[] args){
        try{
            InetAddress ip
=InetAddress.getByName("www.indusuniversity.in");

            System.out.println("Host Name: "+ip.getHostName());
            System.out.println("IP Address:"+ip.getHostAddress());
        }
        catch(Exception e){System.out.println(e);}
    }
}
```

*Output:*

Host Name: www.indusuniversity.in

IP Address: 89.238.188.50

# InetAddress: Method Summary

static InetAddress <b>getByName</b> (String host)	Determines the IP address of a given host's name.
static InetAddress <b>getLocalHost</b> ()	Returns the address of the local host.
public String <b>getHostName</b> ()	it returns the host name of the IP address.
public String <b>getHostAddress</b> ()	it returns the IP address in string format.
String <b>toString</b> ()	Converts this IP address to a String.
boolean <b>equals</b> (Object obj)	Compares this object against the specified object.
static InetAddress[] <b>getAllByName</b> (String host)	Returns an array of its IP addresses, based on the configured name service on the system.
static InetAddress <b>getLoopbackAddress</b> ()	Returns the loopback address.

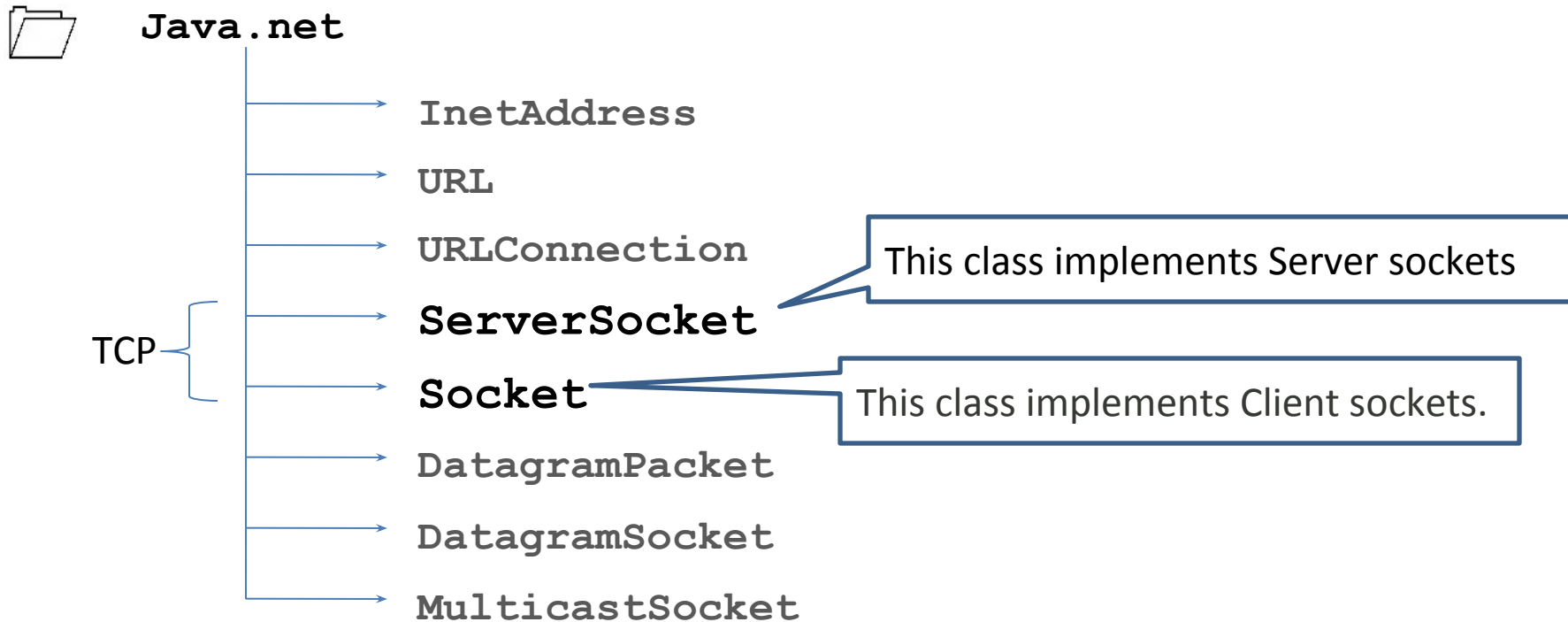
# Unit-1: Java Networking

---

## Topic

1. Network Basics and Socket overview
2. InetAddress
- 3. TCP/IP server sockets**
- 4. TCP/IP client sockets**
5. Datagrams
6. URL
7. URLConnection

# TCP/IP Client-Server sockets



# TCP/IP ServerSocket Class

---

- The ServerSocket class (java.net) can be used to create a server socket.
- This object is used to establish communication with the clients.

## *Constructor*

ServerSocket(int port)	Creates a server socket, bound to the specified port.
------------------------	---

## *Method*

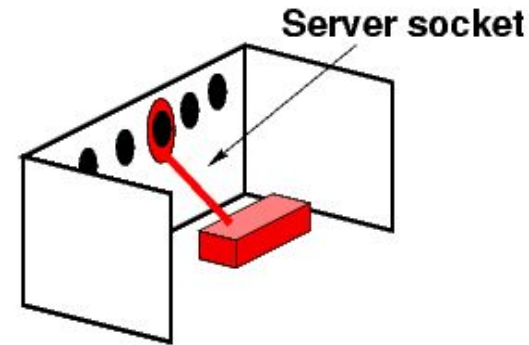
public Socket <b>accept()</b>	returns the socket and establish a connection between server and client.
-------------------------------	--



# TCP/IP Server Sockets

## *Syntax*

```
ServerSocket ss;  
ss=new ServerSocket (Port_no) ;
```



## *Example*

```
ServerSocket ss=new ServerSocket (1111) ;
```

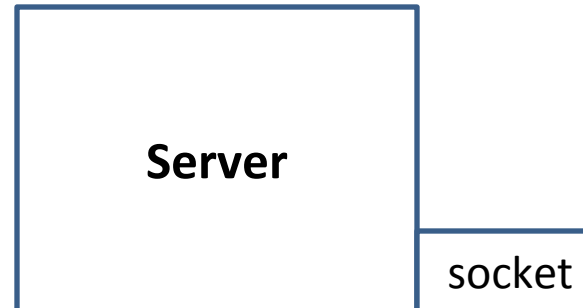
# TCP/IP Server program

## *MyServer.java*

```
1. import java.io.*; // required data input/output stream
2. import java.net.*; //required for Socket Class
3. public class MyServer {
4.     public static void main(String[] args){
5.         try{
6.             ServerSocket ss=new ServerSocket(1111);
7.             Socket s=ss.accept();//establishes connection
8.             DataInputStream dis=
9.                 new DataInputStream (s.getInputStream());
10.            String str=(String)dis.readUTF();
11.            System.out.println("message= "+str);
12.            ss.close();
13.        }
14.        catch(Exception e){System.out.println(e);}
15.    }
16. }
```

## Output

message= Hello Server



# TCP/IP Client Sockets: Socket Class

---

The client in socket programming must know two information:

1. IP Address of Server
2. Port number.

***Constructor***

***Method***

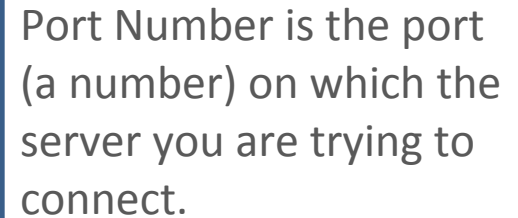
# TCP/IP Client Sockets

## *Syntax*

```
Socket myClient; //Creates object of Socket Class  
myClient = new Socket("Machine name", PortNumber);
```



DNS or IP Address




Port Number is the port (a number) on which the server you are trying to connect.

## **Example**

```
Socket s;  
s=new Socket("localhost",1111);
```

# TCP/IP Client Sockets: Program

```
1. import java.net.*; //required for Socket Class
2. import java.io.*; // required data input/output stream
3. public class MyClient {
4.     public static void main(String[] args)
5.     { try{
6.         Socket s = new Socket("localhost",1111);
7.         DataOutputStream dout= new
DataOutputStream(s.getOutputStream());
8.         dout.writeUTF("Hello Server");// Writes a
string to the underlying output stream
9.     }catch(Exception e)
10.     {System.out.println(e);}
11. }
12. }
```



*Object of Socket class*

# TCP/IP Client-Server program

## *MyServer.java*

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

public class MyServer {
public static void main(String[] args){
try{
ServerSocket ss=new ServerSocket(1111);

Socket s=ss.accept();

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);}
} //psvm
} //class
```

Output  
message= Hello Server

## *MyClient.java*

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

public class MyClient {
public static void main(String[] args){
    try {
        Socket s=new Socket("localhost",1111);

        DataOutputStream dout=new
DataOutputStream(s.getOutputStream());

        dout.writeUTF("Hello Server");
        //Writes string to underlying o/p
        stream

    } //try
    catch(Exception e)
    {System.out.println(e);}
    } //psvm
} //class
```

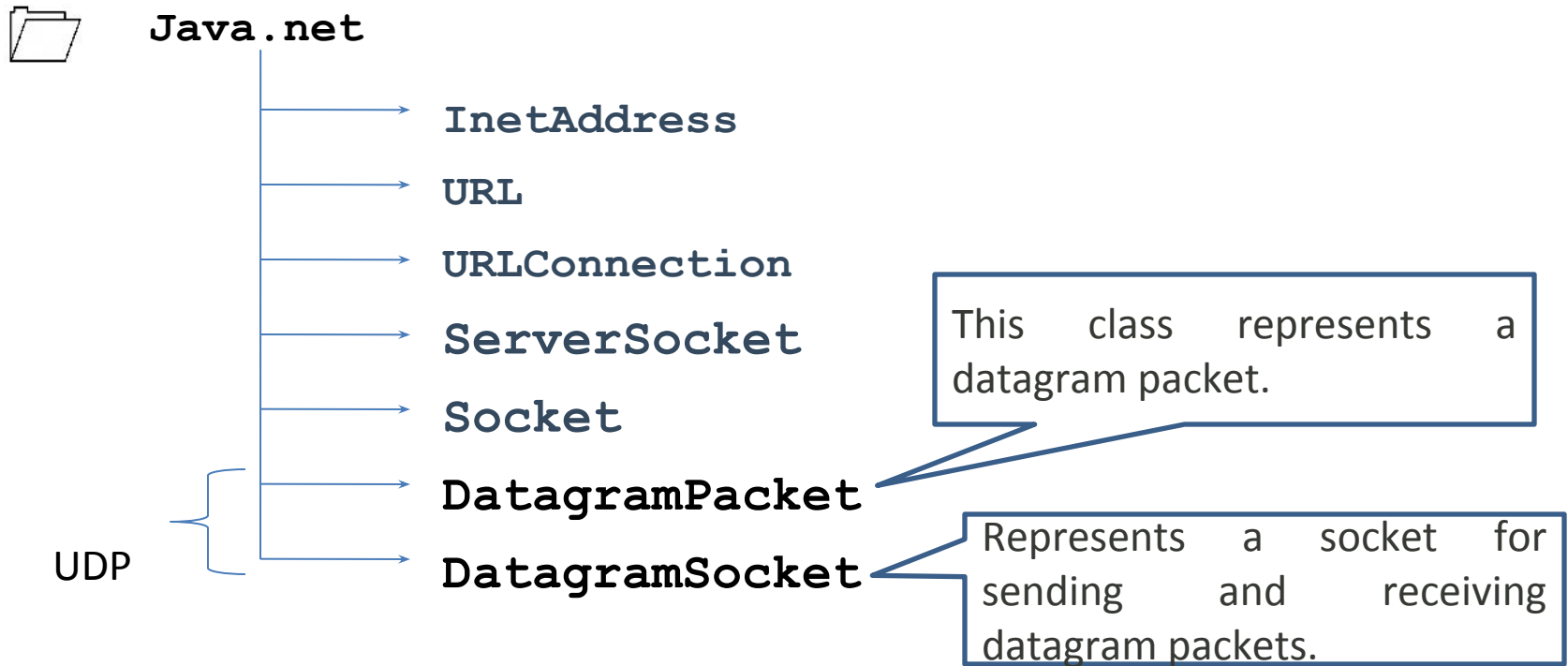
# Unit-1: Java Networking

---

## Topic

1. Network Basics and Socket overview
2. InetAddress
3. TCP/IP server sockets
4. TCP/IP client sockets
- 5. Datagrams**
6. URL
7. URLConnection

# Datagrams





# Datagrams: DatagramSocket class

---

- **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.
- *Constructor*

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

# Datagrams: DatagramPacket class

- **Java DatagramPacket** is a message that can be sent or received.
- If you send multiple packet, it may arrive in any order.
- Additionally, packet delivery is not guaranteed.

**Constructor**

DatagramPacket(byte[] barr, int length)	It creates a datagram packet. This constructor is used to receive the packets.
DatagramPacket(byte[] barr, int length, InetAddress address, int port)	It creates a datagram packet. This constructor is used to send the packets.

## Example of Sending DatagramPacket by DatagramSocket

---

```
import java.net.*; //required for Datagram Class
public class DSender{
    public static void main(String[] args)
        throws Exception
    {
        DatagramSocket ds = new DatagramSocket();
        String str = "Message sent by Datagram socket";
        InetAddress ip = InetAddress.getByName("127.0.0.1");

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

# Example of Receiving DatagramPacket by DatagramSocket

---

```
import java.net.*;
public class DReceiver{
    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();
    }
}
```

*Output*

Message sent by Datagram socket

# Example of Sending and Receiving DatagramPacket

## *DSender.java*

```
import java.net.*;
public class DSender{
public static void main(String[] args)
throws Exception{

DatagramSocket ds=
    new DatagramSocket();
String str = "Message sent by Datagram
    socket";
InetAddress ip =
InetAddress.getByName("127.0.0.1");

DatagramPacket dp = new DatagramPacket
(str.getBytes(),str.length(),ip,3000);

ds.send(dp);
ds.close();

}
}
```

## *DReceiver.java*

```
import java.net.*;
public class DReceiver{
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();
}
}
```

*Output*

Message sent by Datagram socket

# Unit-1: Java Networking

---

## Topic

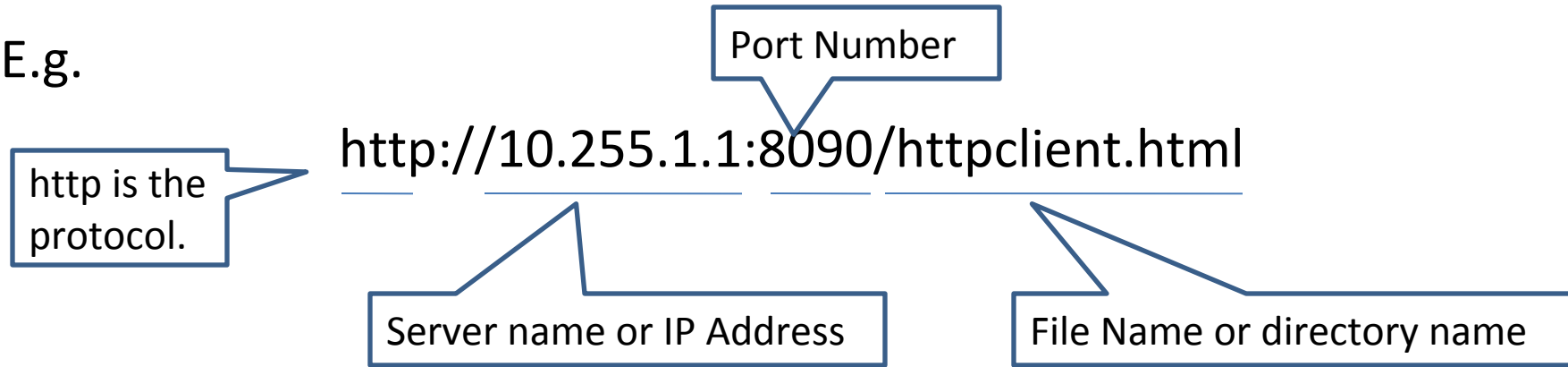
1. Network Basics and Socket overview
2. InetAddress
3. TCP/IP client sockets
4. TCP/IP server sockets
5. Datagrams
- 6. URL**
7. URLConnection

# URL: Uniform Resource Locator

---

- The **Java URL** class represents an URL.
- This class is pointer to “resource” on the World Wide Web.

E.g.



# URL

---

## *Constructor*

<code>URL(String url)</code>	Creates a URL object from the String representation.
------------------------------	--

## *Example*

```
URL url=new URL("http://www.indusuniversity.in");
```

## *Method*

<code>public URLConnection <b>openConnection()</b> throws IOException</code>	This method of URL class returns the object of URLConnection class
--	--

## *Example*

```
URLConnection urlcon=url.openConnection();
```



# Unit-1: Java Networking

---

## Topic

1. Network Basics and Socket overview
2. InetAddress
3. TCP/IP client sockets
4. TCP/IP server sockets
5. Datagrams
6. URL
- 7. URLConnection**

# URLConnection

---

- URLConnection is the superclass of all classes that represent a communications link between the application and a URL.
- Instances of this class can be used both to read from and to write to the resource referenced by the URL.

## *Constructor*

URLConnection(URL <i>url</i> )	Constructs a URL connection to the specified URL
--------------------------------	--

# URLConnection

---

## Method

public InputStream <b>getInputStream()</b> throws IOException	Returns an input stream that reads from this open connection.
public OutputStream <b>getOutputStream()</b> throws IOException	Returns an output stream that writes to this connection.

# URLConnection : Program

---

```
import java.io.*; //required for input stream
import java.net.*; //required for URL & URLConnection
public class URLConnectionDemo {
    public static void main(String[] args){
        try{
            URL url=new URL("http://www.indusuniversity.in");
            URLConnection urlcon=url.openConnection();
            InputStream stream=urlcon.getInputStream();
            int i;
            while((i=stream.read())!=-1) Object of URLConnection Class
                System.out.print((char)i);
        }
    } catch (Exception e){System.out.println(e);}
}
```