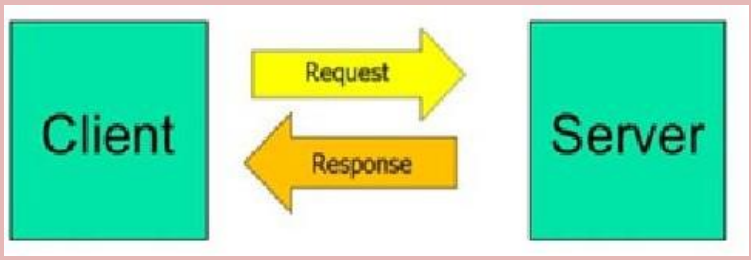
Chapter 4

Networking(10M)

Java Networking is a concept of connecting **two or more computing devices** together so that we can **share resources**.



Advantage of Java Networking

1. sharing resources
2. centralize software management

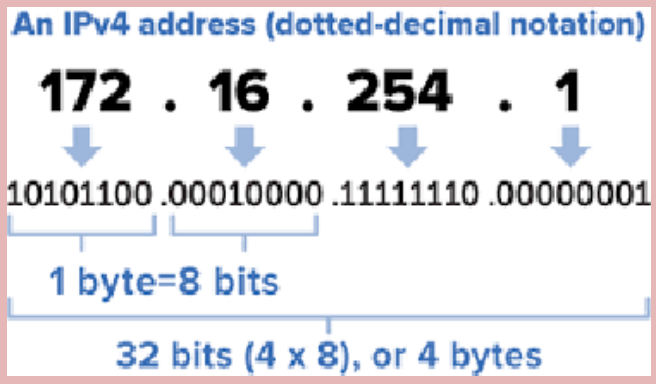
Networking Terminology

The widely used java networking terminologies are given below:

1. IP Address
2. Protocol
3. Port Number
4. MAC Address
5. Connection-oriented and connection-less protocol
6. Socket

1) IP Address

- IP address is a unique number assigned to a node of a network **e.g. 192.168.0.1** . It is composed of octets that range from **0 to 255.**
- It is a logical address that **can be changed**.
- IPv4 uses a **32-bits** value to represent an address**.**



- IPv6 uses a **128-bits(16 bytes)** value to represent an address**.**

2) Protocol

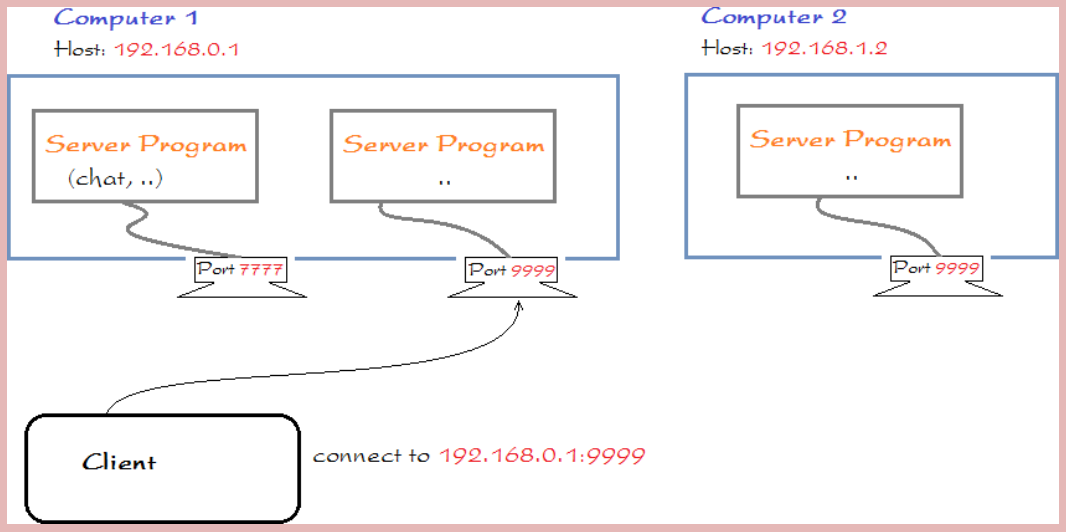
A protocol is a **set of rules** basically that is followed for communication. For example:

- TCP/UDP
- FTP
- Telnet
- SMTP
- POP etc.

3) 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.



- Port number is a two bytes unsigned number ranging from 0-65535.
- **TCP/IP reserves the lower 1,024 ports for specific protocols.**

21 is for FTP,
23 is for Telnet,
25 is for e-mail,
79 is for finger,
80 is for HTTP,
119 is for netnews

4) <u>MAC Address</u>

MAC (Media Access Control) Address is a unique identifier of NIC (Network Interface Controller). A network node can have multiple NIC but each with unique MAC.

5) Connection-oriented and connection-less protocol

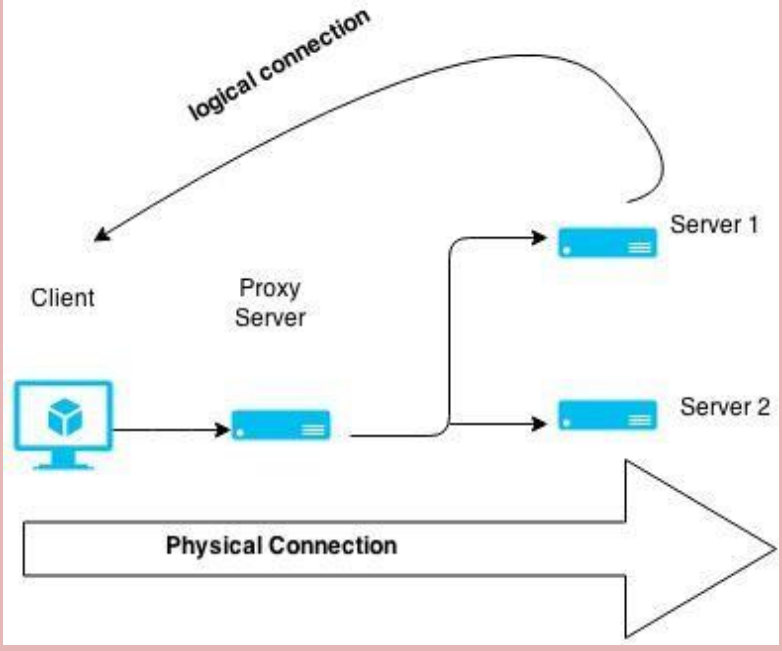
- In connection-oriented protocol, **acknowledgement is sent by the receiver**. So it is **reliable** but slow. The example of **connection-oriented protocol is TCP.**
- But, in **connection-less protocol**, **acknowledgement is not sent by the receiver**. So it is **not reliable but fast.** The example of **connection-less protocol is UDP**.

6) Proxy Server

Proxy server is an intermediary server between client and the internet. Proxy servers offers the following basic functionalities:

- Firewall and network data filtering.
- Network connection sharing
- Data caching

Proxy servers allow to hide, conceal and make your network id anonymous by hiding your IP address.



7) **Domain Naming Service /system(DNS).**

The internet world is completely based on IP (Internet Protocol) address.To access any website you need to know its **IP address which is a long numeric code and is not possible to learn**.

Now, here comes the role of DNS. A DNS is an internet service that **translates a domain name into corresponding IP address**.

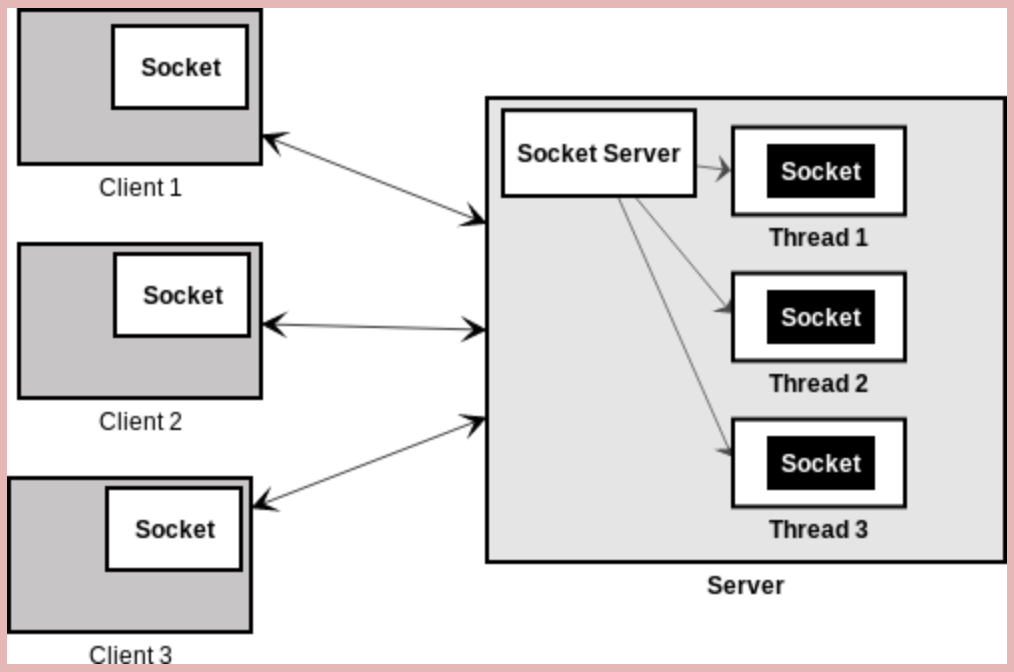
Domain name used here is alphabetic and can be easily remembered.

For example, **www.example.com** is a **domain name of a site**. And with the help of DNS it will get translate into **its IP address 198.105.232.4**.

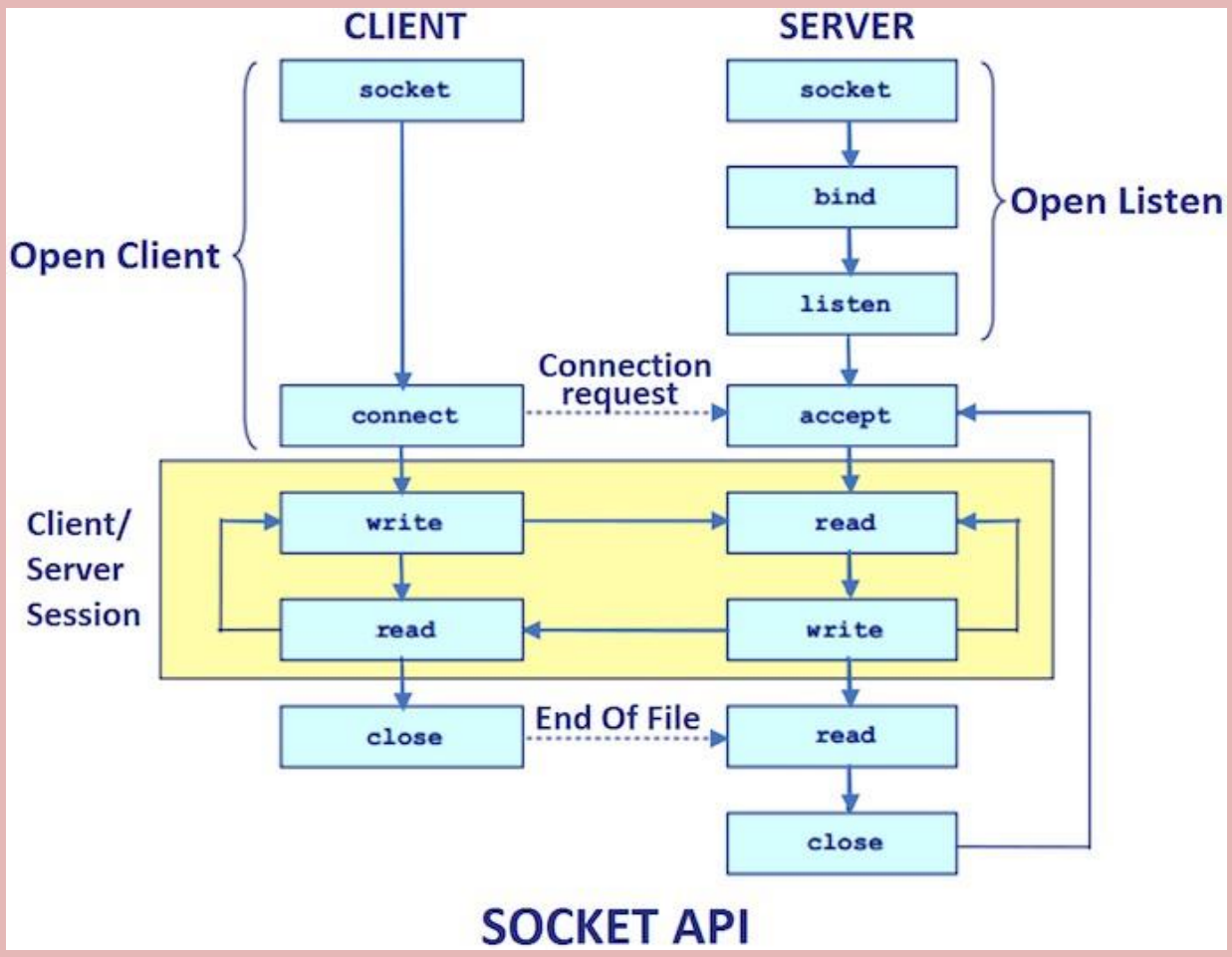
8) Socket

A **socket** is one **endpoint of a two-way communication lin**k between two programs running on the network. ...

The **java**.net package in the **Java** platform provides a class, **Socket** , that implements one side of a two-way connection between your **Java** program and another program on the network.



Client Server Communication



---

java.net package

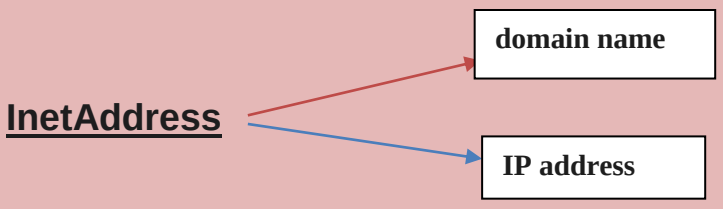
The java.net package provides many classes to deal with networking applications in Java. A list of these classes is given below:

- Authenticator
- CacheRequest
- CacheResponse
- ContentHandler
- CookieHandler
- CookieManager
- **DatagramPacket**
- **DatagramSocket**
- DatagramSocketImpl
- InterfaceAddress

- JarURLConnection
- MulticastSocket
- InetSocketAddress
- **InetAddress**
- Inet4Address
- Inet6Address
- IDN
- HttpURLConnection
- HttpCookie
- NetPermission
- NetworkInterface
- PasswordAuthentication
- Proxy
- ProxySelector
- ResponseCache
- SecureCacheResponse
- **ServerSocket**
- **Socket**
- SocketAddress
- SocketImpl
- SocketPermission
- StandardSocketOptions
- URI
- **URL**
- URLClassLoader
- URLConnection
- URLDecoder
- URLEncoder
- URLStreamHandler

InetAddress

Whether you are making a phone call, sending mail, or establishing a connection across the Internet, addresses are fundamental. The **InetAddress** class is used **to encapsulate** both the numerical IP address and the domain name for that address.



Factory Methods

The **InetAddress** class has **no visible constructors**. To create an **InetAddress object,you have to use one of the available factory methods**

InetAddress provides 3 factory methods are shown here.

getLocalHost( )
getByName(String *hostName*)

getAllByName(String *hostName*)

If these methods are **unable to resolve the host name**, they throw an **UnknownHostException**.

The **getLocalHost( )** method simply returns the **InetAddress** object that represents the **local host**.

The **getByName( )** method returns an **InetAddress** for a host name passed to it.

The **getAllByName( )** factory method returns **an array of InetAddresse**s that represent all of the addresses that a particular name resolves to.

```
import java.net.*;

class InetAddressTest

{

public static void main(String args[]) throws UnknownHostException

{
```

```
InetAddress address = InetAddress.getLocalHost();
System.out.println(address);

address = InetAddress.getByName("yahoo.com");
System.out.println(address);

InetAddress SW[] = InetAddress.getAllByName("www.gmail.com");
 for (int i=0; i<SW.length; i++)
System.out.println(SW[i]);


}
}
```

Acer-PC/103.31.147.90

yahoo.com/206.190.36.45

www.gmail.com/74.125.236.117
www.gmail.com/74.125.236.118
www.gmail.com/2404:6800:4009:800:0:0:0:1015

When laptop is connected to internet

```
/* output
Acer-PC/103.31.147.90
yahoo.com/206.190.36.45

www.gmail.com/74.125.236.117
www.gmail.com/74.125.236.118
www.gmail.com/2404:6800:4009:800:0:0:0:1015
*/
```

When laptop is not connected to internet

```
LAPTOP-ESA250HF/127.0.0.1
Exception in thread "main" java.net.UnknownHostException: yahoo.com
        at java.net.Inet6AddressImpl.lookupAllHostAddr(Native Method)
        at java.net.InetAddress$2.lookupAllHostAddr(Unknown Source)
        at java.net.InetAddress.getAddressesFromNameService(Unknown Source)
```

at java.net.InetAddress.getAllByName0(Unknown Source)
at java.net.InetAddress.getAllByName(Unknown Source)
at java.net.InetAddress.getAllByName(Unknown Source)
at java.net.InetAddress.getByName(Unknown Source)
at InetAddressTest.main(InetAddressTest.java:11)

What is localhost?

- **localhost** refers to **"this computer"** or even more accurately **"the computer I'm working on."**
- The **localhost** is the default name describing **the local computer address** also known as the **loopback address**.
- the local IP address is **127.0.0.1** (the loopback address).

TCP/IP Client Server Communication

- TCP/IP sockets are used to implement **reliable, idirectional, persistent, point-to- point,stream-based connections** between hosts on the Internet.

- There are **two kinds of TCP sockets** in Java. **One is for servers, and the other is for clients**.

- The **ServerSocket(Server)** class is designed to be a **"listener,"** which waits for clients to connect before doing anything.

- The **Socket** class is designed for (Client).

- The creation of a **Socket** object implicitly establishes a connection

between the client and server.

- **ServerSocket** has a method called **accept( )**,which is a blocking call that will wait for a client to initiate communications,

//TCPIP Server client Communication

Server Program

```
import java.net.*;
import java.io.*;
public class SimpleServer
{
public static void main(String args[])
{
ServerSocket ss=null;
try
{
 ss=new  ServerSocket(132);
}
catch(IOException E)
{
}
while (true)
{
try
{
Socket s=ss.accept();
OutputStream os=s.getOutputStream();
BufferedWriter bw=new BufferedWriter(new OutputStreamWriter(os));
```

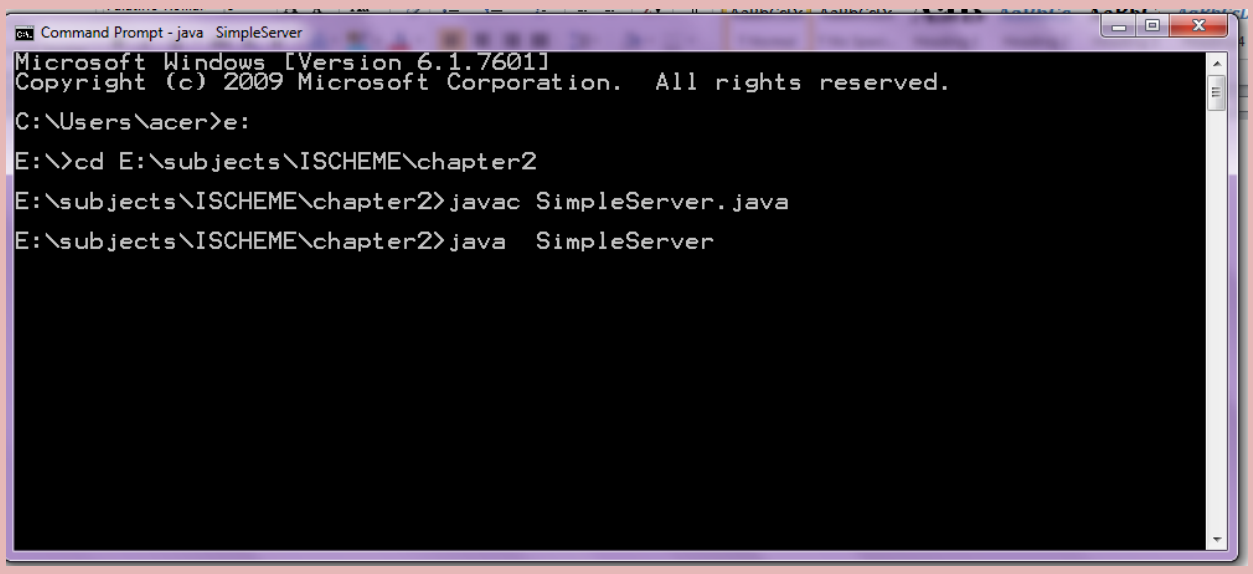
```
bw.write("Hello.....");
bw.close();
s.close();
}
catch(IOException e)
{
}
}
}
}
```

Client Program

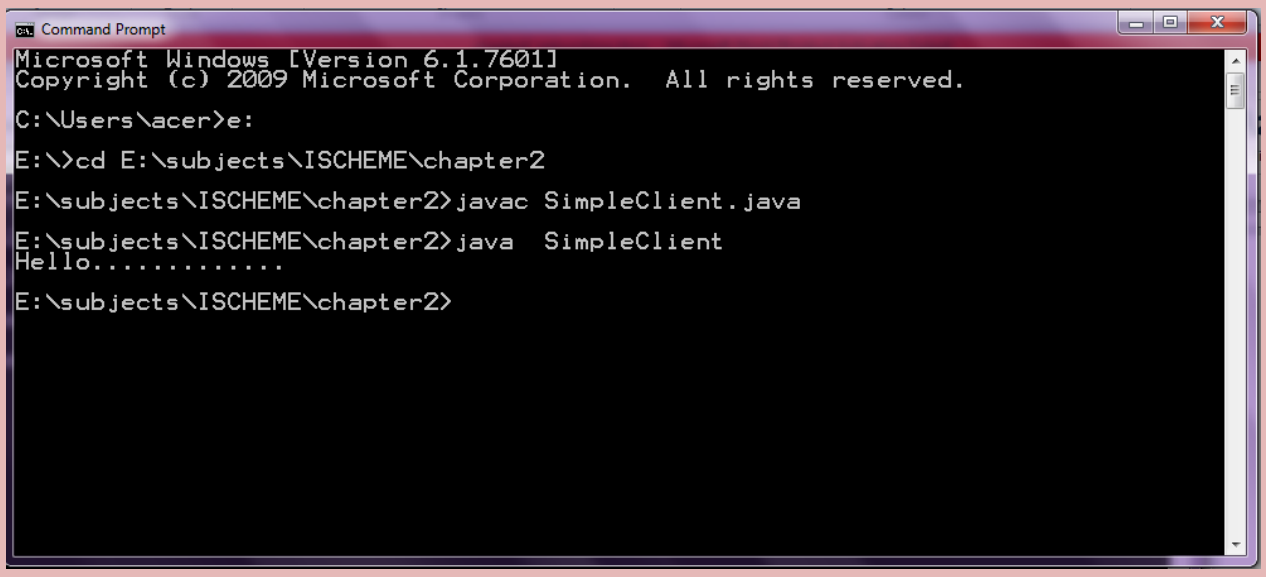
```
import java.net.*;
import java.io.*;
public class SimpleClient
{
public static void main(String args[])
{
try
{
Socket s1=new Socket("127.0.0.1",132);
InputStream is=s1.getInputStream();
BufferedReader br=new BufferedReader(new InputStreamReader
```

```
(is)); System.out.println(br.readLine());
br.close();
s1.close();
}
catch(Exception e)
{}
}
}
```

Server Window



Client Window



Here are two constructors used to create client **sockets:**

Socket(String *hostName*, int *port*)

Socket(InetAddress *ipAddress*, int *port*)

ServerSocket constructors are:

ServerSocket(int *port*)
ServerSocket(int *port*, int *maxQueue*)

ServerSocket(int *port*, int *maxQueue*,
InetAddress *localAddress*)

Chatting application

1)**Server program**

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

```
public class  chatserver
{
        public static void main(String args[]) throws Exception
        {
                ServerSocket ss=new ServerSocket(2000);
                Socket sk=ss.accept();

                BufferedReader cin=new BufferedReader(new
                InputStreamReader(sk.getInputStream()));
                     //for reading from client cmd prompt

                PrintStream cout=new PrintStream(sk.getOutputStream());
                    //for printing on client cmd

                     BufferedReader stdin=new BufferedReader(new
                    InputStreamReader(System.in));
                    //for reading from same cmd prompt

                    String s;
                 while ( true )
                {
                        s=cin.readLine();
```

```
                if (s.equalsIgnoreCase("END"))
                {
                        cout.println("BYE");
                        break;
                  }
                System.out.print("Client :"+s+"\n");
                System.out.print("Server: ");
                s=stdin.readLine();
                cout.println(s);
            }
            ss.close();
            sk.close();
            cin.close();
            cout.close();
            stdin.close();
        }
}
```

2)**Client Program**

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

```
public class  chatclient
{
        public static void main(String args[]) throws Exception
        {
                Socket sk=new Socket("localhost",2000);
                BufferedReader sin=new BufferedReader(new
          InputStreamReader(sk.getInputStream()));
                        //for reading from server cmd prompt


                PrintStream sout=new PrintStream(sk.getOutputStream());//for
printing on server cmd
                BufferedReader stdin=new BufferedReader(new
InputStreamReader(System.in));
                      //for reading from same cmd prompt String s;
                while ( true )
                {
                        System.out.print("Client : ");
                        s=stdin.readLine();
                        sout.println(s);
                        s=sin.readLine();
                        System.out.print("Server : "+s+"\n");
```

```
                if ( s.equalsIgnoreCase("BYE") )
                   break;
            }
             sk.close();
             sin.close();
             sout.close();
            stdin.close();
        }
}
```

Whois

The whois command displays information about a **website's record**.

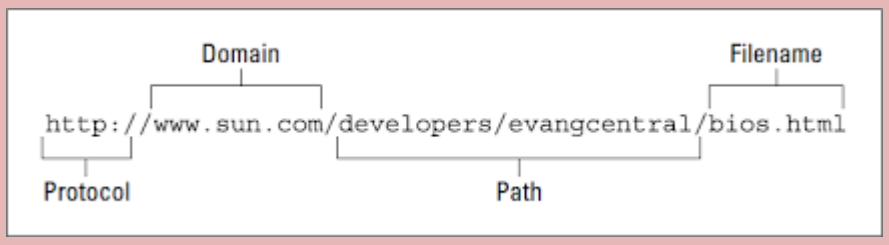
You may get all the information about a website regarding **its registration and owner's information.**

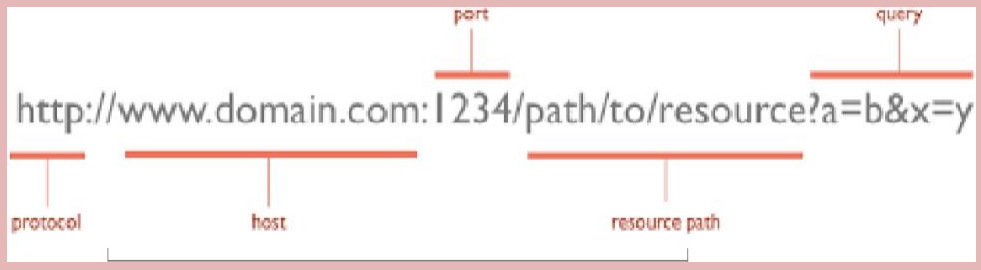
URL

The **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:

1. https://www.google.com/index.html





A URL contains many information(components of URL):

1. **Protocol:** In this case, http is the protocol.
2. **Server name or IP Address:** In this case, www.google.com is the server name.
3. **Port Number:** It is an optional attribute. If we write http//ww.google.com:80/index, 80 is the port number. If port number is not mentioned in the URL**, it returns -1.**
4. **File Name or directory name:** In this case, index.html is the file name.

// Demonstrate URL.

import java.net.*;

class URLDemo

{

public static void main(String args[]) throws MalformedURLException

```
{
URL hp = new URL("http://www.google.com/downloads");
System.out.println("Protocol: " + hp.getProtocol());
System.out.println("Port: " + hp.getPort());
System.out.println("Host: " + hp.getHost());
System.out.println("File: " + hp.getFile());
System.out.println("Ext:" + hp.toExternalForm());
}
}
```

output:

Protocol: http
Port: -1
Host: www.google.com
File: /downloads
Ext:http://www.google.com/downloads

- The next two forms of the constructor allow you to break up the URL into its component parts:

URL(String *protocolName*, String *hostName*, int *port*, String *path*)
URL(String *protocolName*, String *hostName*, String *path*)
URL(URL *urlObj*, String *urlSpecifier*)

- **URL** class has several constructors, and each can throw a **MalformedURLException**

URLConnection

URLConnection is a general-purpose class for accessing the attributes of a remote resource.

In the following example, we create a **URLConnection** using the **openConnection( )**method of a **URL** object and then use it to examine the document's properties and content:

```
import java.net.*;

import java.io.*;

 import java.util.Date;


class UCDemo

{

public static void main(String args[]) throws Exception

{

int c;

URL hp = new URL("http://www.gmail.com");

URLConnection hpCon = hp.openConnection();


// get date

long d = hpCon.getDate();

if(d==0)
```

```
System.out.println("No date information.");
else
System.out.println("Date: " + new Date(d));

// get content type
System.out.println("Content-Type: " + hpCon.getContentType());
// get expiration date
d = hpCon.getExpiration();

if(d==0)
System.out.println("No expiration information."); else
System.out.println("Expires: " + new Date(d));

// get last-modified date
d = hpCon.getLastModified();
if(d==0)
System.out.println("No last-modified information."); else
System.out.println("Last-Modified: " + new Date(d));
```

```
// get content length
int len = hpCon.getContentLength();
if(len == -1)
System.out.println("Content length unavailable."); else
System.out.println("Content-Length: " + len);
}
}
/*
E:\imp\co6c\chapter2>java UCDemo

Date: Wed Oct 20 10:50:26 IST 2020
Content-Type: text/html; charset=UTF-8
Expires: Mon Jan 01 05:30:00 IST 1990
No last-modified information.
Content-Length: 352
*/
```

Datagrams(UDP)

- *Datagrams* are **bundles of information(packets)** passed between machines.
- Once the datagram has been released to its intended target, there is no assurance that it will arrive or even that someone will be there to catch it.(connection less protocol)

- when the datagram is received, there is no assurance that it hasn't been damaged in transit or that whoever sent it is still there to receive a response.

- The **DatagramPacket** object is the **data container**,while the **DatagramSocket** is the mechanism used to **send or receive** the **DatagramPacket**s.
- DatagramSocket is used to create socket for communication.

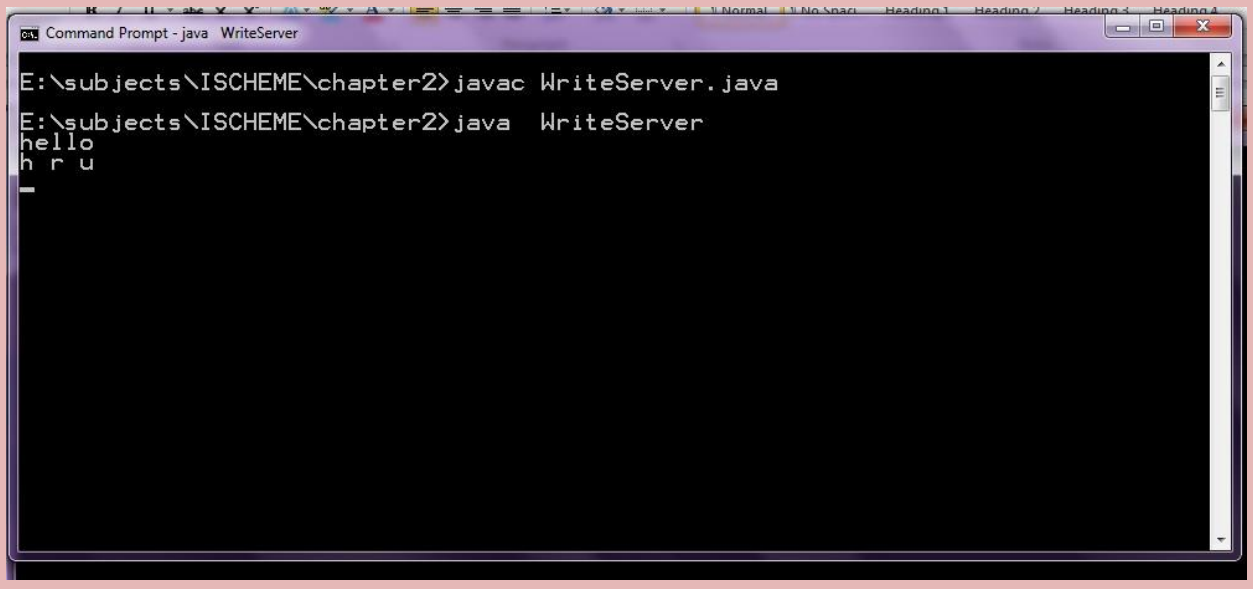
```
//
 Demonstrate Datagrams.
import java.net.*;
class WriteServer
{
public static int serverPort = 999;
public static int clientPort = 998;
public static int buffer_size = 1024;
public static DatagramSocket ds;
public static byte buffer[] = new byte[buffer_size];

public static void theServer() throws Exception
{
int pos=0;
```

```
while (true)
{
int c = System.in.read();

switch (c)
{
case -1: System.out.println("Server
quit"); return;

case '\n':
ds.send(new DatagramPacket(buffer,pos,
InetAddress.getLocalHost(),clientPort));
pos=0;
break;

default:
buffer[pos++] = (byte) c;
}
}
}
```

```
public static void theClient() throws Exception
{
while(true)
{
DatagramPacket p = new DatagramPacket(buffer, buffer.length); ds.receive(p);
System.out.println(new String(p.getData(), 0, p.getLength()));
}
}

public static void main(String args[]) throws Exception
{
if(args.length == 1)
{
ds = new DatagramSocket(serverPort);
theServer();
}

else {
ds = new DatagramSocket(clientPort);
theClient();
```

```
}

}


}
```


Command Prompt - java WriteServer 1
E:\subjects\ISCHEME\chapter2>javac WriteServer.java
E:\subjects\ISCHEME\chapter2>java WriteServer 1
hello
h r u


Command Prompt - java WriteServer
E:\subjects\ISCHEME\chapter2>javac WriteServer.java
E:\subjects\ISCHEME\chapter2>java WriteServer
hello
h r u