# Networking Basics

**Addressing computers in a network:**

Using TCP/IP protocol computers in a network are addressed by using IP address.

The IP address changes from one IP version to another IP version.

**There are two ip versions:**

IPV4 and IPV6

* In IPV4 ip address size is 4 bytes
* In IPV6 ip address size is 16 bytes

**IPV4 address format is:**

xxx.xxx.xxx.xxx

each xxx will occupy 1byte

**IPV6 address format is:**

xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx

each xxxx will occupy 2bytes

**IPV4 address starts from:**

0.0.0.0

0.0.0.1

0.0.0.2

0.0.0.3

....

....

0.0.0.255

0.0.1.0

0.0.1.1

0.0.1.2

...

...

0.0.1.255

0.0.2.0

....

....

255.255.255.255

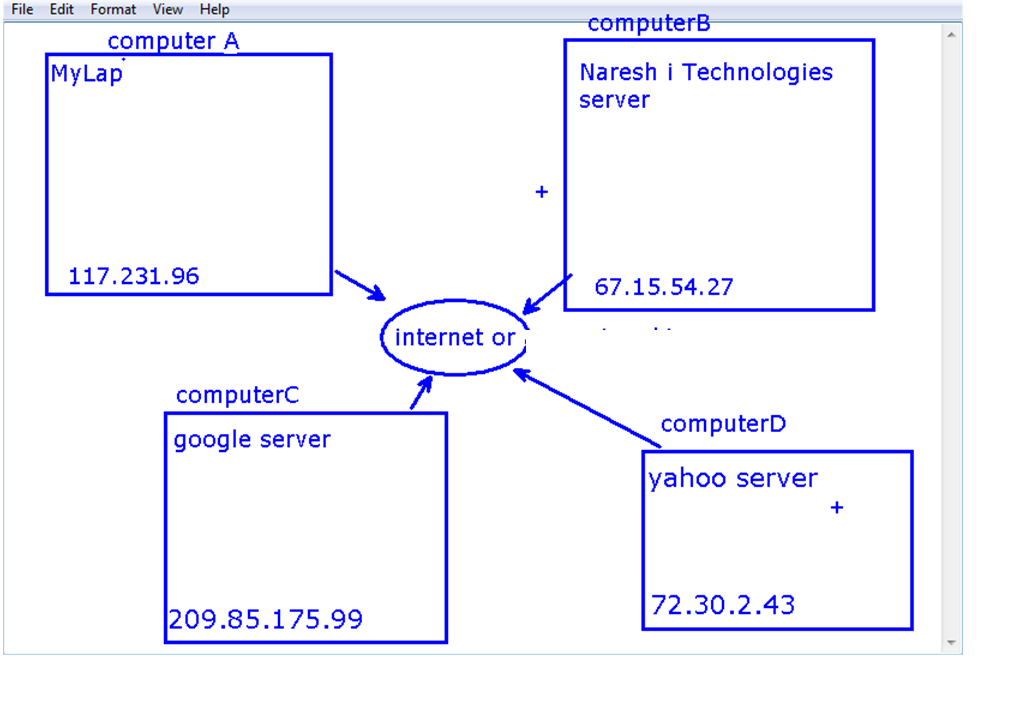
**All these address are divided into**

class A IP

class B IP

class C IP

class D IP



**How a computer can send message to another computer?**

A computer as hardware component cannot send message to another computer

So a program running one computer can send message to another program running in another computer.

This kind of program is called as networking program.

**Addressing networking programs running in a computer:**

Each networking program running in one computer can communicates with another networking program running in the same computer or running in the other computer

The networking programs must have unique addresses.

* The address assigned to a networking program is called port number.
* Port number size is 2 bytes.
* So port numbers ranges from 0 to 65535.

**note:** from 0 to 1024 are reserved by operating system for predefined network services.

Few services examples:

https->443, http->80, telnet->23, ftp->21

**How can we develop networking programs in java?**

ans: We can develop networking programs in java by using java.net package classes and interfaces.

**Networking programs are two types:**

* server programs
* client programs

**What is server?**

ans: A server is a networking program that can receive a request from client and it can send response to the client.

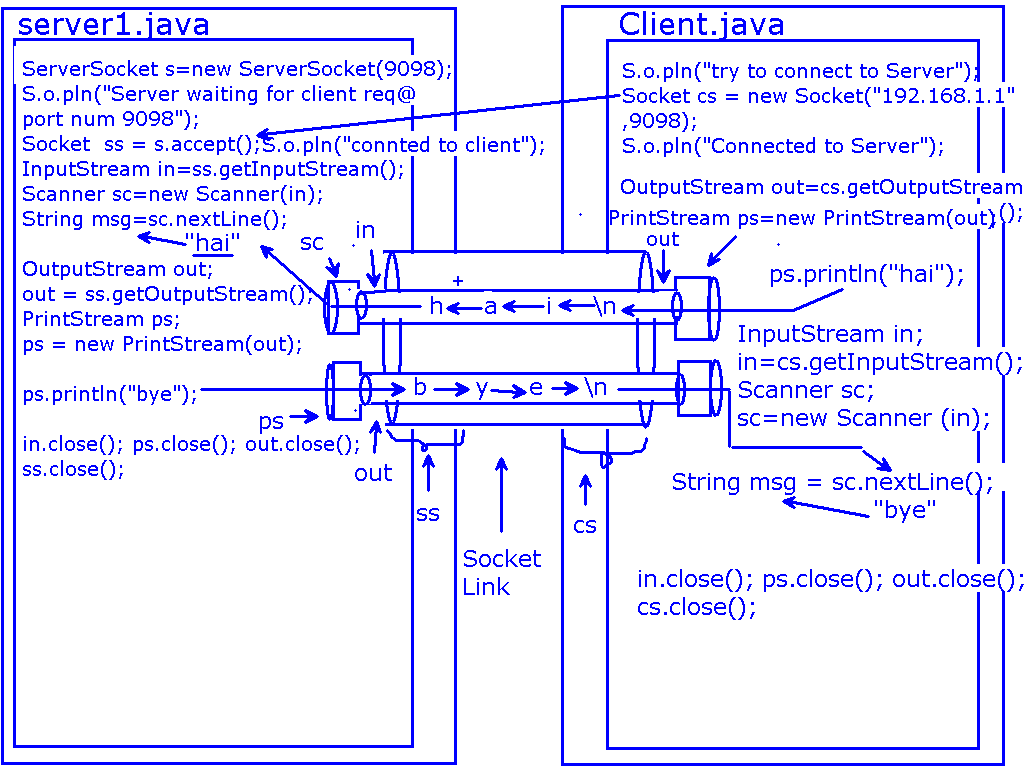
**What is client?**

ans: A client is also a networking program that can send a request to the server and it can receive a response from the server.

**How can we develop client and server programs?**

We can develop client and server program by using java.net.ServerScoket and java.net.Socket classes of networking API.

|  |  |
| --- | --- |
| **Steps to create a server program**  ***step1:***  **Register server port number or Bind server port number**  *ServerSocket server = new ServerSocket(7070);*  ***step2:* wait for client connection request & accept it**  *Socket ss = server.accept();*  ***step3:* getinput stream and connect to scanner and read msg given by client**  *InputStream in = ss.getInputStream();*  *Scanner sc = new Scanner(in);*  *String msg = sc.nextLine();*  ***step4:* get output stream and connect to printstream and send bye dear to client**  *ouputStream out = ss.getOuputStream();*  *PrintStream ps = new PrintStream(out);*  *ps.println("byedear");*  *S.o.pln(msg);*  ***step5:***  *in.close(); ps.close(); out.close(); ss.close();* | **Steps to create a client program**  ***step1:* give connection request to server program**  *Socket cs = new Socket("192.168.1.1",7070);*  ***step2:* get output stream and connect it to print stream and send "hai" to server**  *OuputStream out = cs.getOutputStream();*  *PrintStream ps = new PrintStream(out);*  *ps.println("hai");*  ***step3:***  **getinput stream and connect to scanner and read msg given by server**  *InputStream in = cs.getInputStream();*  *Scanner sc = new Scanner(in);*  *String msg = sc.nextLine();*  *S.o.pln(msg);*  ***step4:***  *in.close(); ps.close(); out.close(); cs.close();* |



**Chatting Server code**

|  |
| --- |
| import java.util.\*;  import java.io.\*;  import java.net.\*;  public class ChattingServerA{  public static void main(String[] args)throws Exception{  ServerSocket s=new ServerSocket(9098);  System.out.println("wainting for client request @ port num 9098");  Socket ss = s.accept();  System.out.println("connected to client");  //prepare scanner to read msgs from client  InputStream in = ss.getInputStream();  Scanner sc = new Scanner(in);  //prepare printStream to send msg to the client  OutputStream out = ss.getOutputStream();  PrintStream ps = new PrintStream(out);  //prepare scanner to read text from keypad  Scanner scanf = new Scanner(System.in);  String smsg="";//to hold sending msg  String rmsg="";//to hold receving msg  System.out.printf("\n wating for msg from client...");  rmsg =sc.nextLine();//reading msg from client  while(rmsg.equalsIgnoreCase("bye")==false){  System.out.printf("\nmsg from client:"+rmsg);  //read msg to send to client  System.out.printf("\nEnter msg to send to client:");  smsg = scanf.nextLine();  //send it to client  ps.println(smsg);  System.out.printf("\n wating for msg from client...");  rmsg =sc.nextLine();//reading msg from client  }  System.out.printf("\n msg from client:"+rmsg);//displaying bye  ps.println("bye");//sending bye to client  in.close();  ps.close();  out.close();  ss.close();  }  } |

**ChattingClientCode**

|  |
| --- |
| import java.util.\*; import java.io.\*; import java.net.\*;  public class ChattingClientA{  public static void main(String[] args)throws Exception{  //prepare scanner to read input from keyboard  Scanner scanf = new Scanner(System.in);  System.out.printf("\nTrying to Connect to Server");  Socket cs = new Socket("localhost",9098);  System.out.printf("\n connected to the server");  //prepare Scanner to read msg from server  InputStream in = cs.getInputStream();  Scanner sc = new Scanner(in);  //prepare printStream to send msg to server  OutputStream out = cs.getOutputStream();  PrintStream ps = new PrintStream(out);  String rmsg="";//to hold received msg  String smsg="";//to hold sending msg  while(rmsg.equalsIgnoreCase("bye")==false){  System.out.printf("\n enter msg to send to the server:");  smsg = scanf.nextLine();  ps.println(smsg);//sending to the server  System.out.printf("\nwaiting for msg from server....");  rmsg = sc.nextLine();  System.out.printf("\nmsg from server="+rmsg);  }  ps.println("bye");//sending bye to server  // close all  in.close(); ps.close(); out.close(); cs.close(); } } |