**Java Networking**: ( java.net package):

* Is a concept of connecting two or more computing devices together so that we can share resource.
* Java Socket programming provides facility to share data between different computing devices.

Java Networking Terminology:

* **IP Address:**

Is a unique number assigned to a node of a network (logical address can be changed).

* **Protocol:**

Is a set of rules that followed for communication. (TCP, FTP, HTTP, SMTP etc.)

* **Port Number:**

Is a used to uniquely identify different applications, (it acts as communication endpoint associated with IP Address for communication between two applications).

* **MAC Address:**

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

* **Connection-oriented and Connection less protocol:**

In connection-oriented protocol acknowledgement is sent by the receiver, it is reliable but slow. Eg: TCP. In connection-less protocol, acknowledgement is not sent by receiver, not reliable but fast Eg: UDP.

* **Socket:**

A socket is an endpoint between two communications.

**Java Socket Programming:**

* Is used for communication between the applications running on different JRE.
* Can be connection-oriented or connection-less

**Socket** and **ServerSocket classes** are used for **connection-oriented** socket programming**.**

**DatagramSocket** and **DatagramPacket classes** used for **connection-less** socket programming**.**

**[**Note: Socket class can be used to create a Socket, it is simply an endpoint for communication

ServerSocket object is used to establish connection Server and Client]

The client in socket programming must know two information:

* IP Address of Server and
* Port Number.

1.Simple client-server application, server receives string from client: use two command prompts.

**Creating Server:**

*//Creating a Server*ServerSocket server=**new** ServerSocket(6666);  
Socket s=server.accept();  
DataInputStream dis=**new** DataInputStream(s.getInputStream());  
String str=(String) dis.readUTF();  
System.***out***.println(**" received message = "**+str);  
s.close();

**Creating Client:**

Socket client = **new** Socket(**"localhost"**, 6666);  
DataOutputStream dout=**new** DataOutputStream(client.getOutputStream());  
dout.writeUTF(**"HELLO SERVER"**);  
dout.flush();  
dout.close();

2. Simple client-server application read and write (chatting app):

**Creating Server:**

ServerSocket srv=**new** ServerSocket(5555);  
Socket sc=srv.accept();  
DataInputStream din=**new** DataInputStream(sc.getInputStream());  
DataOutputStream dout=**new** DataOutputStream(sc.getOutputStream());  
BufferedReader reader=**new** BufferedReader(**new** InputStreamReader(System.***in***));  
String input=**""**;  
String output=**""**;  
**while** (!input.equals(**"stop"**)){  
 input=din.readUTF();  
 System.***out***.println(**"Client says : "** + input);  
 output=reader.readLine();  
 dout.writeUTF(output);  
 dout.flush();  
}  
din.close();  
sc.close();  
srv.close();

**Creating Client:**

Socket cl=**new** Socket(**"localhost"**,5555);  
DataInputStream din=**new** DataInputStream(cl.getInputStream());  
DataOutputStream dout=**new** DataOutputStream(cl.getOutputStream());  
BufferedReader reader=**new** BufferedReader(**new** InputStreamReader(System.***in***));  
String input=**""**;  
String output=**""**;  
**while** (!input.equals(**"stop"**)){  
 input=reader.readLine();  
 dout.writeUTF(input);  
 dout.flush();  
 output=din.readUTF();  
 System.***out***.println(**"Server says : "**+output);  
}  
dout.close();  
cl.close();

**DatagramSocket and DatagramPacket classes:** represents a connection less socket for sending and receiving datagram packets. (there is no guarantee of contents arrival or arrival time).

**Sending DatagramPacket by DatagramSocket:**

DatagramSocket ds = new DatagramSocket();

String str = "Welcome java";

InetAddress ip = InetAddress.getByName("127.0.0.1");

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

ds.send(dp);

ds.close();

**Receiving DatagramPacket by DatagramSocket:**

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();