Networking:

If we design and execute any application on the basis of client-server architecture or by distributing application logic over multiple machines then that application is called Distributed applications.

There are two types of Distributed applications.
1. Web based Distributed applications.
2. Remote Based Distributed Applications.

Web Based Distributed Applications:

It is a Distributed application, where we will use a browser as a client and a java program as a Server side application.

To prepare Web Based Distributed applications we have to use the technologies like Servlets, JSPs, JSTL and Expression Language.

Remote Based Distributed Applications:

It is a distributed application, where we will use a java program with main method , a servlet , a JSP page , a Framework applications struts, JSF or a spring as client and a java program as a server side application.

In Remote Based Distributed applications, we will define services at a remote machine[Server Machine] called Remote Services and these Services are consumed by the program at a local machine[Client machine] called Local Application or consumer.

To prepare distributed applications , JAVA has provided the following technologies.
1. Socket programming
2. RMI[Remote Method Invocation]
3. CORBA[Common Object request Broker Arch]
4. EJBs[Enterprise Java Beans]
5. Web Services

Distributed Applications Through Socket programming:

To prepare Distributed applications, if we use Socket programming then we have to use Sockets , where the provided Sockets will establish the connection on the basis of the IP address and port number.

Q)What is the difference between IP Address and Port number?

Ans:

IP Address is an unique identity for each and every machine in the network and which will be provided by the Network Manager at the time of Network configuration.

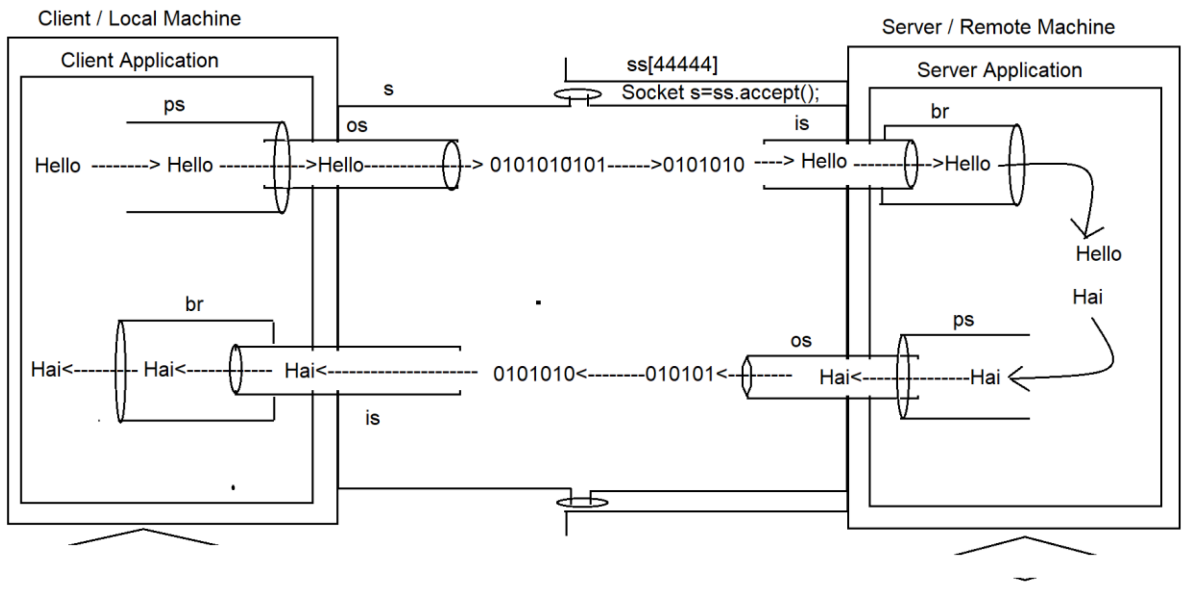
Port number is an unique identity for each and every process being executed in a single computer , where all the port numbers are managed by the operating system.

Socket: Socket is a channel or medium , it can be used to transfer data from one machine to another machine over the network.

There are two types of Sockets.

1. ServerSocket
2. Socket

Socket Programming Arch:


Client / Local Machine
Client Application
ps
Hello --------> Hello ---------->Hello----------> 0101010101------>0101010 ----> Hello -------->Hello
s
os
ss[44444]
Socket s=ss.accept();
Server / Remote Machine
Server Application
is
br
Hello
Hai
br
ps
os
Hai<-------- Hai<--------- Hai<-------------------- 0101010<--------010101<-------- Hai<---------------Hai
is

Steps to prepare Distributed Applications by Using Socket Programming
----
Client Side Application:
1. Create Client Side Socket:
```
      Socket s = new Socket("localhost", 4444);
```

With the above Socket creation, Socket is requesting the ServerSocket which is running the current system at the port number 4444 about to establish connection with the Server machine.

2. Create an OutputStream from the Socket:
```
      OutputStream os = s.getOutputStream();
```

3. Create PrintStream to improve Output operation performance.
```
      PrintStream ps = new PrintStream(os);
```

4. Send data to the PrintStream:
```
      String data = "Hello";
      ps.println(data);
```
When we send data to the PrintStream , PrintStream will take data , it will send data to the OutputStream, where OutputStream will send data to the Client Side Socket, where Client Side Socket will send data to the Server side Socket, where Server Side socket will send data to the Server program.

5. Create InputStream from Socket:
```
      InputStream is = s.getInputStream();
```

6. Create BufferedReader with InputSTream:
```
      BufferedReader br = new BufferedReader(new InputStreamReader(is));
```

7. Read Data from BufferedReader:
```
      String data = br.readLine();
      System.out.println(data);
```

Server Side Application:
1. Create Server Socket :
```
      ServerSocket ss = new ServerSocket(4444);
```

2. Accept the Client side sockets requests about the connections:
```
      Socket s = ss.accept();
```

3. Create an InputSTream from Socket:

```
      InputStream is = s.getInputStream();
```

4. Create a BufferedReader with InputSTream to improve Input operation performance.
```
      BufferedReader br = new BufferedReader(new InputStreamReader(is));
```

5. Read data from BufferedReader:
```
      String data = br.readLine();
      System.out.println(data);
```

6. Create OutputStream from Socket:
```
      OutputStream os = s.getOutputStream();
```

7. Create PrintStream with the OutputStream:
```
      PrintStream ps = new PrintStream(os);
```

8. Send data to the PrintSTream:
```
      String data = “Hai”;
      ps.println(data);
```

With the above instruction, data will be send to PrintStream, where PrintStream will send data to the OutputSTream, where OutputStream will send data to the Server Socket, where Server side socket will send data to the Client Side Socket, where Client Side Socket will send data to the CLient Application.

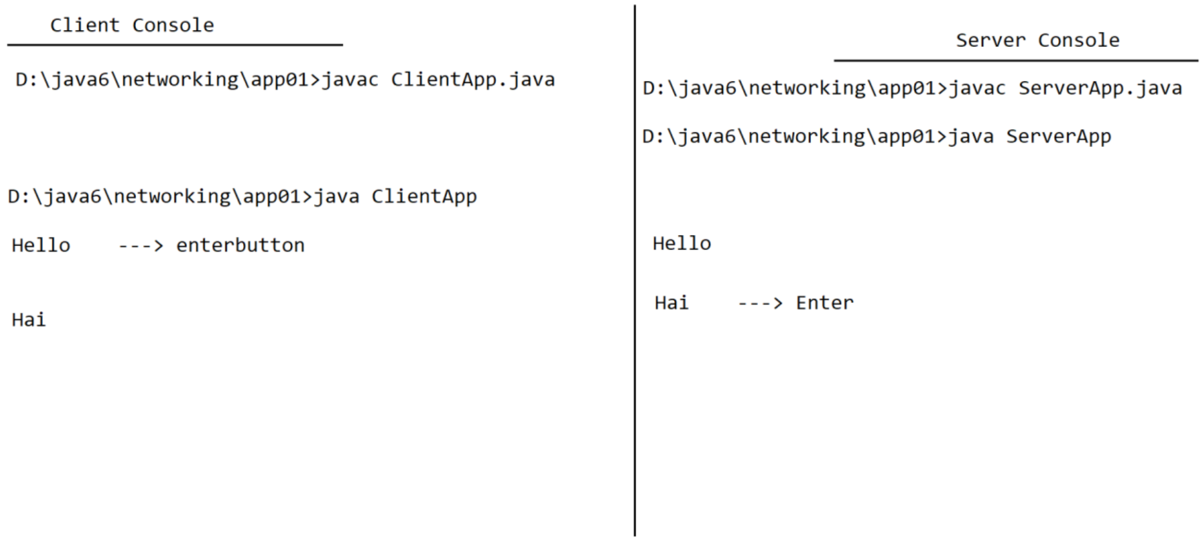
Execution Process.
1. Open a separate console for Client Application.
2. Open a separate console for Server application.
3. Compile both the programs .
4. Run Server side program in Server console.
5. Run Client Side program in Client Console
6. Provide messages at both consoles and see the messages exchange.

EX:D:\java6\networking\app01

```
ClientApp.java
import java.io.*;
import java.net.*;
class ClientApp{
      public static void main(String[] args)throws Exception{
            Socket s = new Socket("localhost", 4444);
            OutputStream os = s.getOutputStream();
            PrintStream ps = new PrintStream(os);
            BufferedReader br1 = new BufferedReader(new
InputStreamReader(System.in));
            String data1 = br1.readLine();
            ps.println(data1);

            InputStream is = s.getInputStream();
            BufferedReader br2 = new BufferedReader(new
InputStreamReader(is));
            String data2 = br2.readLine();
            System.out.println(data2);


      }
}
```

ServerApp.java

```
import java.io.*;
import java.net.*;
public class ServerApp{
      public static void main(String[] args)throws Exception{
            ServerSocket ss = new ServerSocket(4444);
            Socket s = ss.accept();
            InputStream is = s.getInputStream();
            BufferedReader br1 = new BufferedReader(new
InputStreamReader(is));
            String data1 = br1.readLine();
            System.out.println(data1);

            OutputStream os = s.getOutputStream();
            PrintStream ps = new PrintStream(os);
            BufferedReader br2 = new BufferedReader(new
InputStreamReader(System.in));
            String data2 = br2.readLine();
            ps.println(data2);


      }
}
```



```
Client Console
D:\java6\networking\app01>javac ClientApp.java

D:\java6\networking\app01>java ClientApp
Hello    ---> enterbutton

Hai
```

```
Server Console
D:\java6\networking\app01>javac ServerApp.java
D:\java6\networking\app01>java ServerApp

Hello

Hai    ---> Enter
```

EX:
ClientApp.java

```
import java.io.*;
import java.net.*;
class ClientApp{
      public static void main(String[] args)throws Exception{
            Socket s = new Socket("localhost", 4444);
            OutputStream os = s.getOutputStream();
            PrintStream ps = new PrintStream(os);
            BufferedReader br1 = new BufferedReader(new
InputStreamReader(System.in));


            InputStream is = s.getInputStream();
            BufferedReader br2 = new BufferedReader(new
InputStreamReader(is));


            while(true){
                  String data1 = br1.readLine();
                  ps.println(data1);

                  String data2 = br2.readLine();
                  System.out.println(data2);

                  if(data1.equalsIgnoreCase("bye") &&
data2.equalsIgnoreCase("bye")){
```

```
                        System.exit(0);
                  }
            }


      }
}


ServerApp.java
import java.io.*;
import java.net.*;
public class ServerApp{
      public static void main(String[] args)throws Exception{
            ServerSocket ss = new ServerSocket(4444);
            Socket s = ss.accept();
            InputStream is = s.getInputStream();
            BufferedReader br1 = new BufferedReader(new
InputStreamReader(is));


            OutputStream os = s.getOutputStream();
            PrintStream ps = new PrintStream(os);
            BufferedReader br2 = new BufferedReader(new
InputStreamReader(System.in));


            while(true){
                  String data1 = br1.readLine();
                  System.out.println(data1);

                  String data2 = br2.readLine();
                  ps.println(data2);

                  if(data1.equalsIgnoreCase("bye") &&
data2.equalsIgnoreCase("bye")){
                        System.exit(0);
                  }
            }


      }
}
```