The Java socket programming is used for communication between two applications running on same JRE or Different JRE. The Socket programming provides facility to share the data between two applications. We can develop the distributed application on the basis of SOCKET using “socket Programming” Technology. The JAVA has provided the complete predefined library in the form of “java.net “ package . This package is useful to prepare the Distributive Applications Using Socket Technology. The java.net package supports two protocols,

1.TCP/IP:- It provides **reliable** communication between sender application and receiver application. It is connection oriented protocol. Before starting communication , The connection is established between sender and receiver.

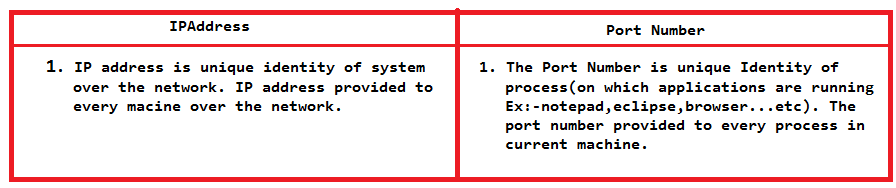
The Socket and ServerSocket classes are used for connection oriented socket programming.

2. UDP:- It provides Unreliable communication between sender application and receiver application. IT is connection less protocol.

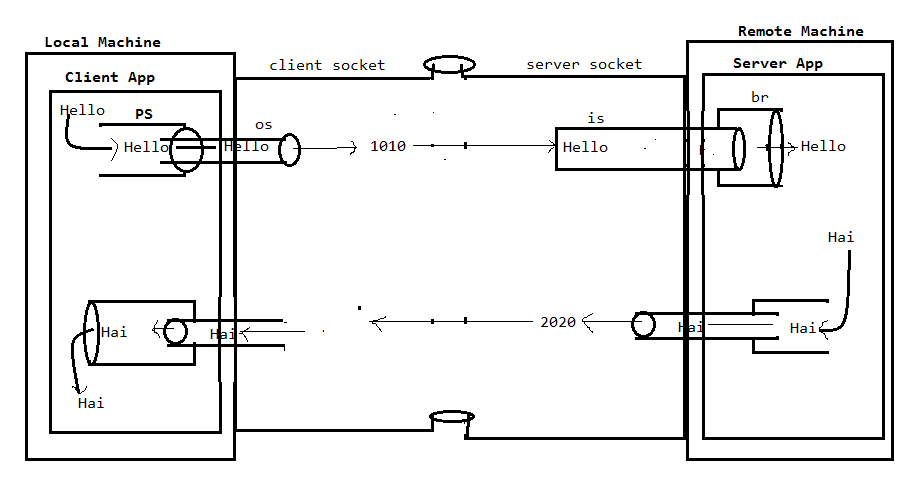
DatagramSocket and DatagramPacket classes are used for connection-less socket programming.

**1.Socket:-** The socket is an endpoint between The socket is created on the basis of System IP address and Port Number.

* 1. **Difference Between IPAddress and Port Number:**

****

**2.socket programming Architecture:**

****

**3. Steps to Prepare Client Application:**

3.1. Create Socket at client machine.

a. To create Socket class object, we have to use following constructor from java.net.Socket class.

Syntax:

Public Socket(String Ipaddress, int portno);

3.2. Get Output Stream from socket:

a. To get OutputStream attached with Socket, we have to use following method from java.net .Socket class.

Syntax:

Public OutputStream getOutputStream();

3.3. Create Print Stream and connect with OutputStream.

Syntax:

New PrintStream(OutputstreamObject);

3.4 . Send the data to print Stream

printStream object. Println(“data”);

Note:- With above steps , data will be sent to server, where server will send response data to client.

3.5. Get Input stream from socket:

To get Inputstream attached with socket, we have to use the following method . Syntax:

Public InputStream getInputStream();

3.6. Create BufferedReader with InputStream:

BufferedReader br=new BufferedReader(new InputStreamReader(InputStream Object));

3.7. Read the data from BufferedReader using readLine().

3.8. Close this socket.

Public Synchronized void close();

**4.Steps To Prepare Server Application:**

Create a socket at sever machine, we have to use following constructor from java.rmi.ServerSocket.

ServerSocket ss=new ServerSocket(int portnumber);

4.1. Get InputStream attached wtih socket:

InputStream is1=ss.getInputStream();

4.2. Create BufferedReader with InputStream:

BufferedReader br=new BufferedReader(new InputStreamReader(InputStream Object));

4.3. Read the data from BufferedReader using readLine().

4.4. GetOutputStream attached with Server socket:

To get OutputStream attached with Socket, we have to use following method from java.net .Socket class.

Syntax:

Public OutputStream getOutputStream();

4.5. Create Print Stream with outputStream

PrintStream ps=new PrintStream(OutStream object);

4.6. Send the data to printStream.

4.7. close the server socket.

Public synchronized void close();

**Chating Application**:

ServerApplication:

import java.io.BufferedReader;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.io.OutputStream;

import java.io.PrintStream;

import java.net.ServerSocket;

import java.net.Socket;

public class Server {

public static void main(String[] args) {

try {

ServerSocket ss=new ServerSocket(7777);

Socket s=ss.accept();

InputStream is=s.getInputStream();

BufferedReader b1=new BufferedReader(new InputStreamReader(System.in));

BufferedReader b2=new BufferedReader(new InputStreamReader(is));

OutputStream os=s.getOutputStream();

PrintStream ps=new PrintStream(os);

while(true)

{

String data2=b2.readLine();

System.out.println(data2);

String data1=b1.readLine();

ps.println(data1);

if(data1.equals("bye")&&data2.equals("bye"))

{

s.close();

System.exit(0);

}

}

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

}

}

ClientApplication:

import java.io.BufferedReader;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.io.OutputStream;

import java.io.PrintStream;

import java.net.Socket;

public class Client {

public static void main(String[] args) {

try {

Socket s=new Socket("localhost",7777);

OutputStream os=s.getOutputStream();

PrintStream ps=new PrintStream(os);

BufferedReader b1=new BufferedReader(new InputStreamReader(System.in));

InputStream is=s.getInputStream();

BufferedReader b2=new BufferedReader(new InputStreamReader(is));

while(true)

{

String data1=b1.readLine();

ps.println(data1);

String data2=b2.readLine();

System.out.println(data2);

if(data1.equals("bye")&&data2.equals("bye"))

{

s.close();

System.exit(0);

}

}

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

}

}

Output:-

