**Socket Programming in Java**

**Networking** is a concept of connecting two or more computing devices together so that we can share resources like printer, scanner, memory.

In Networking application mainly two programs are running one is **Client program** and another is **Server program**. In Core java **Client program** can be design using **Socket** class and **Server program** can be design using **ServerSocket** class.

Both Socket and ServerSocket classes are predefined in **java.net** package

**Advantage of Network Programming**

The main advantage of network Programming is sharing of data and resources, some more advantages are;

* Sharing resources like printer, Scanner.
* Centralize software management, Software install on only one system and used in multiple system.
* Sharing of data due to this reduce redundancy of application.
* Burden on the developer can be reduced.
* Wastage of memory can be reduced because no need to install same application on every system.
* Time consuming process to develop application is reduced.

**Terms used in Socket Programming**

**Port number:**It is unique identification value represents residing position of a server in the computer. It is four digit +ve number.

**Port Name:**It is a valid user defined name to know about client system, the default port name for any local computer is **localhost.**. Port name should be the some value which is given at Server programming.

**Socket class**

Socket class are used for design a client program, it have some constructor and methods which are used in designing client program.

**Constructor:**Socket class is having a constructor through this Client program can request to server to get connection.

**Syntax to call Socket() Constructor**

Socket s=**new** Socket("localhost", 8080);

// localhost -- port name and 8080 -- port number

**Note:**If given port name is invalid then UnknownHostException will be raised.

**Method of Socket class**

* public InputStream getInputStream()
* public OutputStream getOutputStream()
* public synchronized void close()

**getInputStream()**

This method take the permission to write the data from client program to server program and server program to client program which returns OutputStream class object.

**Syntax**

Socket s=**new** Socket("localhost", 8080);

OutputStream os=**new** s.getOutputStream();

DataOutputStream dos=**new** DataOutputStream(os);

**getOutputStream()**

This method is used to take the permission to read data from client system by the server or from the server system by the client which returns InputStream class object.

**Syntax**

Socket s=**new** Socket("localhost", 8080);

InputStream **is**=**new** s.getInputStream();

DataInputStream dis=**new** DataInputStream(**is**);

**close()**

This method is used to request for closing or terminating an object of socket class or it is used to close client request.

**Syntax**

Socket s=**new** Socket("localhost", 8080);

s.close();

**ServerSocket class**

The ServerSocket class can be used to create a server socket. ServerSocket object is used to establish the communication with clients.

ServerSocket class are used for design a server program, it have some constructor and methods which are used in designing server program.

**Constructor:**ServerSocket class contain a constructor used to create a separate port number to run the server program.

**Syntax to call ServerSocket() Constructor**

ServerSocket ss=**new** ServerSocket(8080);

// 8080 -- port number

**Method of ServerSocket class**

* public Socket accept()
* public InputStream getInputStream()
* public OutputStream getOutputStream()
* public synchronized void close()

**accept():**Used to accept the client request it returns class reference.

**Syntax**

Socket s=**new** Socket("localhost", 8080);

ServerSocket ss=**new** ServerSocket(8080);

Socket s=ss.accept();

**Rules to design server program**

* Every server program should run accepted port number (any 4 digit +ve numeric value) It can set by relating an object for server socket class (In any used defined java program).

**Syntax**

ServerSocket ss=**new** ServerSocket(8080);

* Accept client request.
* Read input data from client using InputStream class.
* Perform valid business logic operation
* Send response (writing output data) back to client using OutputStream class.
* close or terminate client request.

**Rules to design client program**

* Obtain connection to the server from the client program (any user defined class) by passing port number and port name in the socket class.

**Syntax**

Socket s=**new** Socket(8080, "localhost");

// 8080 is port number and localhost is port name

* Send request (writing input data) to the server using OutputStream class.
* Read output data from the server using InputStream class.
* Display output data

**Note:**close the connection is optional.

**Server Code**

// Saved by Server.java

**import** java.net.\*;

**import** java.io.\*;

**class** Server

{

**public** **static** **void** main(String[] args)

{

**try**

{

**int** pno=Integer.parseInt(args[0]);

ServerSocket ss=**new** ServerSocket(pno);

System.**out**.println("server is ready to accept clint request");

Socket s1=ss.accept();

InputStream **is**=s1.getInputStream();

DataInputStream dis=**new** DataInputStream(**is**);

**int** n=dis.readInt();

System.**out**.println("Value from client : "+n);

**int** res=n\*n;

OutputStream os=s1.getOutputStream();

DataOutputStream dos=**new** DataOutputStream(os);

dos.writeInt(res);

s1.close();

}

**catch** (Exception e)

{

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

}

}

}

**Client Code**

// Saved by Client.java

**import** java.net.\*;

**import** java.io.\*;

**import** java.util.\*;

**class** Client

{

**public** **static** **void** main(String[] args)

{

**try**

{

String pname=args[0];

**int** pno=Integer.parseInt(args[1]);

Socket s=**new** Socket(pname,pno);

System.**out**.println("clint obtailed connection from server");

System.**out**.println("Enter a number ");

Scanner sn=**new** Scanner(System.**in**);

**int** data=sn.nextInt();

OutputStream os=s.getOutputStream();

DataOutputStream dos=**new** DataOutputStream(os);

dos.writeInt(data);

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

DataInputStream dis=**new** DataInputStream(**is**);

**int** res=dis.readInt();

System.**out**.println("Result from server : "+res);

}

**catch** (Exception e)

{

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

}

}

}