Java Socket Programming

Java Socket programming is used for communication between the applications running on different JRE.

Java Socket programming can be connection-oriented or connection-less.

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

The client in socket programming must know two information:
1 IP Address of Server, and
2 Port number.

Socket Class

A socket is simply an endpoint for communications between the machines. The Socket class can be used to create a socket.

Important methods

public InputStream getInputStream() : returns the InputStream attached with this socket
public OutputStream get OutputStream() : returns the OuputStream attached with this socket
public synchronized void close(): closes this socket

ServerSocket class

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

Important methods

public Socket accept() : returns the socket and establish a connection between server and client
public synchronized void close(): closes the server socket

Example 1: Java socket program in which client sends a text and server receives it

MyServer.java

```
import java.io.*;
import java.net.*;
public class MyServer {
```

```
        public static void main(String[] args){
        try{
        ServerSocket ss=new ServerSocket(6666);
        Socket s=ss.accept();//establishes connection
        DataInputStream dis=new DataInputStream(s.getInputStrea
        m());
        String  str=(String)dis.readUTF();
        System.out.println("message= "+str);
        ss.close();
        }catch(Exception e){System.out.println(e);}
        }
}
```

MyClient.java

```
        import java.io.*;
        import java.net.*;
        public class MyClient {
        public static void main(String[] args) {
        try{
        Socket s=new Socket("localhost",6666);
        DataOutputStream dout=new DataOutputStream(s.getOutput
        Stream());
        dout.writeUTF("Hello Server");
        dout.flush();
        dout.close();
        s.close();
        }catch(Exception e){System.out.println(e);}
        }
}
```

Example 2: Client will write first to the server then server will receive and print the text. Then server will write to the client and client will receive and print the text. The step goes on.

MyServer.java

```
        import java.net.*;
        import java.io.*;
        class MyServer{
        public static void main(String args[])throws Exception{
        ServerSocket ss=new ServerSocket(3333);
```

```
        Socket s=ss.accept();
        DataInputStream din=new DataInputStream(s.getInputStrea
        m());
        DataOutputStream dout=new DataOutputStream(s.getOutput
        Stream());
        BufferedReader br=new BufferedReader(new InputStreamRe
        ader(System.in));

        String str="",str2="";
        while(!str.equals("stop")){
        str=din.readUTF();
        System.out.println("client says: "+str);
        str2=br.readLine();
        dout.writeUTF(str2);
        dout.flush();
        }
        din.close();
        s.close();
        ss.close();
}}
```

MyClient.java

```
        import java.net.*;
        import java.io.*;
        class MyClient{
        public static void main(String args[])throws Exception{
        Socket s=new Socket("localhost",3333);
        DataInputStream din=new DataInputStream(s.getInputStrea
        m());
        DataOutputStream dout=new DataOutputStream(s.getOutput
        Stream());
        BufferedReader br=new BufferedReader(new InputStreamRe
        ader(System.in));

        String str="",str2="";
        while(!str.equals("stop")){
        str=br.readLine();
        dout.writeUTF(str);
        dout.flush();
        str2=din.readUTF();
        System.out.println("Server says: "+str2);
        }
```

```
        dout.close();
        s.close();
}}
```

Java URL

The **Java URL** class represents an URL. URL is an acronym for Uniform Resource Locator. It points to a resource on the World Wide Web. For example:

http://www.manipal.edu/mu/student.html

A URL contains many information:

- **Protocol:** In this case, http is the protocol.
- **Server name or IP Address:** In this case, www.javatpoint.com is the server name.
- **Port Number:** It is an optional attribute. If we write http://www.manipal.edu:80/mu/student.html, 80 is the port number. If port number is not mentioned in the URL, it returns -1.
- **File Name or directory name:** In this case, student.html is the file name.

Important Methods

public string getProtocol(): it returns the host name of the URL
public string getHost(): it returns protocol of the URL
public string getPort(): it returns port number of the URL
public string getFile(): it returns file name of the URL
public URLConnection openConnection(): it returns instance of URL, that is associated with his URL

Example 1: URL Class

```
import java.io.*;
import java.net.*;
public class URLDemo{
public static void main(String[] args){
try{
URL url=new URL("http://www.manipal.edu/mu/student.html
");

System.out.println("Protocol: "+url.getProtocol());
System.out.println("Host Name: "+url.getHost());
System.out.println("Port Number: "+url.getPort());
System.out.println("File Name: "+url.getFile());
```

```
        }catch(Exception e){System.out.println(e);}
        }
}
```

Java URLConnection class

The Java URLConnection class represents a communication link between the URL and the application. This class can be used to read and write data to the specified resource referred by the URL.

How to get the object of URLConnection class

The openConnection() method of URL class returns the object of URLConnection class. Syntax:

```
public URLConnection openConnection()throws IOException{}
```

Example1: Displaying source code of a webpage by URLConnecton class

```
        import java.io.*;
        import java.net.*;
        public class URLConnectionExample {
        public static void main(String[] args){
        try{
        URL url=new URL("http://www.manipal.edu/mu/student.html
        ");
        URLConnection urlcon=url.openConnection();
        InputStream stream=urlcon.getInputStream();
        int i;
        while((i=stream.read())!=-1){
        System.out.print((char)i);
        }
        }catch(Exception e){System.out.println(e);}
        }
}
```

The URLConnection class provides many methods, we can display all the data of a webpage by using the getInputStream() method. The getInputStream() method returns all the data of the specified URL in the stream that can be read and displayed.

Java HttpURLConnection class

The Java HttpURLConnection class is http specific URLConnection. It works for HTTP protocol only.

By the help of HttpURLConnection class, you can information of any HTTP URL such as header information, status code, response code etc.

The java.net.HttpURLConnection is subclass of URLConnection class.

How to get the object of HttpURLConnection class

The openConnection() method of URL class returns the object of URLConnection class. Syntax:

```
public URLConnection openConnection()throws IOException{}
```

You can typecast it to HttpURLConnection type as given below.

```
URL url=new URL("http://www.manipal.edu/mu/student.html");
HttpURLConnection huc=(HttpURLConnection)url.openConnection();
```

Example 1: Generate the http url information such as date, cookie, content type, expiry date etc.

```
        import java.io.*;
        import java.net.*;
        public class HttpURLConnectionDemo{
        public static void main(String[] args){
        try{
        URL url=new URL("http://www.manipal.edu/mu/student.html
        ");
        HttpURLConnection huc=(HttpURLConnection)url.openConnect
        ion();
        for(int i=1;i<=8;i++){
        System.out.println(huc.getHeaderFieldKey(i)+" = "+huc.getH
        eaderField(i));
        }
        huc.disconnect();
        }catch(Exception e){System.out.println(e);}
    }
}
```

Java InetAddress class

Java InetAddress class represents an IP address. The java.net.InetAddress class provides methods to get the IP of any host name for example www.manipal.edu, www.google.com, www.facebook.com etc.

Important Methods

public static InetAddress getByName(String host):
it returns the instance of InetAddress containing LocalHost IP and name.

public static InetAddress getLocalHost():
it returns the instance of InetAdddress containing local host name and address.

public String getHostName()
it returns the host name of the IP address.

public String getHostAddress()
it returns the IP address in string format.

Example 1: InetAddress class to get IP address of www.manipal.edu website.

```
import java.io.*;
import java.net.*;
public class InetDemo{
public static void main(String[] args){
try{
InetAddress ip=InetAddress.getByName("www.manipal.edu");

System.out.println("Host Name: "+ip.getHostName());
System.out.println("IP Address: "+ip.getHostAddress());
}catch(Exception e){System.out.println(e);}
}
}
```

Java DatagramSocket and DatagramPacket

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

Java DatagramSocket class

Java DatagramSocket class represents a connection-less socket for sending and receiving datagram packets.
A datagram is basically an information but there is no guarantee of its content, arrival or arrival time.

Commonly used Constructors of DatagramSocket class

DatagramSocket() throws SocketEeption: it creates a datagram socket and binds it with the available Port Number on the localhost machine.
DatagramSocket(int port) throws SocketEeption: it creates a datagram socket and binds it with the given Port Number.
DatagramSocket(int port, InetAddress address) throws SocketEeption: it creates a datagram socket and binds it with the specified port number and host address.

Java DatagramPacket class

Java DatagramPacket is a message that can be sent or received. If you send multiple packet, it may arrive in any order. Additionally, packet delivery is not guaranteed.

Commonly used Constructors of DatagramPacket class

DatagramPacket(byte[] barr, int length): it creates a datagram packet. This constructor is used to receive the packets.
DatagramPacket(byte[] barr, int length, InetAddress address, int port): it creates a datagram packet. This constructor is used to send the packets.

Example1: Sending and receiving a packet using UDP

MySender.java

```
import java.net.*;
public class DSender{
public static void main(String[] args) throws Exception {
DatagramSocket ds = new DatagramSocket();
String str = "Welcome java";
InetAddress ip = InetAddress.getByName("127.0.0.1");
DatagramPacket dp = new DatagramPacket(str.getBytes(), st
r.length(), ip, 3000);
ds.send(dp);
ds.close();
}
```

```
}
```

MyReceiver.java

```
import java.net.*;
public class DReceiver{
public static void main(String[] args) throws Exception {
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();
}
}
```