Madan Bhandari Memorial College
Department of Computer Science and Information Technology (B.Sc.CSIT)
Ninayak Nagar, New Baneshwor, Kathmandu

Practical Sheet

Submitted By:- Utsav Acharya Sharma
Program No:- 07
Submitted To Debesh Adhikari
Lab Date:- 2080/08/14
Submission Date:- 2080/08/18
T.U.Roll.No. :- 24179

Title: Java Network Programming. (Socket)

Introduction:

Sockets:

In java a socket is a software endpoint that establishes a communication link between two computers over a network. Sockets provide a mechanism for processes on different devices to communicate, either on the same machine or across a network.

Use of socket programming:

Socket programming is used for communication between two devices over a network. It enables processes on different machines to exchange data. Socket programming is widely used in client-server applications.

TCP	UDP
→ Transmission Control Protocol.	→ User Datagram Protocol.
→ Connection-oriented protocol.	→ Connection less protocol.
→ Provides reliable and ordered delivery of data.	→ Does not guarantee reliable delivery of data.
→ Slower compared to UDP	→ Faster compared to TCP.
→ Offers error checking and correction.	→ Lacks features like error checking and correction.
→ Example: HTTP, FTP, SMTP, etc.	→ Example: DNS, DHCP, etc.

- Which package contains all of the java networking classes and interfaces?

⇒ The java.net package contains all the classes and interfaces for networking operations in Java.

- How Socket and ServerSocket classes work in TCP communications.

Socket Class	ServerSocket Class
• Represents an endpoint for communication. • Used by clients to connect to a server. • Provides input and output streams for data exchange. • Methods like 'getInputStream()' are used for communication.	• Listens for incoming connection requests from clients. • Used by servers to accept client connections. • When a connection is accepted, it returns a Socket object for communication.

Steps for creating clients and creating servers and how they communicate:

Server Side	Client Side
• Create a ServerSocket object and specify a port number. • Use the accept() method to wait for and accept incoming client connections. • Upon accepting a connection, obtain a Socket object for communication. • Create input and output streams to send and receive data. • Implement the desired communication protocol.	• Create a Socket object and specify the server's IP address and port number. • Create input and output streams for data exchange. • Implement the desired communication protocol. • Close the socket when communication is complete.

Unicasting	Multicasting
→ Involves communication between one sender and one receiver. → Uses a standard 'DatagramSocket' and 'DatagramPacket'. → E.g: Sender creates a DatagramSocket and sends data via DatagramPacket to a specific IP address or Port.	→ Involves communication between one sender and multiple receiver. → Uses a MulticastSocket for both sender and receiver. → Eg. Sender creates a MulticastSocket specifies a multicast group address, and sends data to that group using DatagramPacket.

TCP Code:

Server:

```
import java.net.*;
public class Server{
    private Socket socket = null;
    private ServerSocket server = null;
    private DataInputStream inputFromClient = null;
    private DataOutputStream outputToClient = null;
    public Server(int port){
        try{
            server = new ServerSocket(port);
            System.out.println("Server started");
            System.out.println("Waiting for a client...");
            Socket = server.accept();
            System.out.println("Client accepted");
            inputFromClient = new DataInputStream(new BufferedInput
                                                  Stream(socket.getInput
            outputToClient = new DataOutputStream          Stream());
                                (socket.getOutputStream());
            String line = "";
            while (!line.equals("over")){
                try{
                    line = inputFromClient.readUTF();
                    System.out.println(line);
                    outputToClient.writeUTF(line);
                }catch(IOException i){
                    system.out.println(i);
                }
            }
            System.out.println("Closing connection");
            inputFromClient.close();
            outputToClient.close();
            socket.close();
        }catch(IOException i){
            System.out.println(i);
        }
    }public static void main(String args[]){
            Server server = new Server(5000);
}   }
```

Client:

```
import java.io.*;
import java.net.*;
public class Client{
    private Socket socket = null;
    private DataInputStream inputFromServer = null;
    private DataOutputStream outputToServer = null;
    private DataInputStream inputFromUser = null;
    public Client(String address, int port){
        try{
            socket = new Socket(address, port);
            System.out.println("Connected");
            inputFromServer = new DataInputStream(new BufferedInput
                                Stream(socket.getInputStream()));
            outputToServer = new DataOutputStream(socket.getOutputStream
                                ());
            inputFromUser = new DataInputStream(System.in);
            String line = "";
            while(!line.equals("Over")){
                try{
                    line = inputFromUser.readLine();
                    outputToServer.writeUTF(line);
                    String serverResponse = inputFromServer.readUTF();
                    System.out.println("Echo from Server:"+serverResponse);
                }catch(IOException i){
                    System.out.println(i);
                }
            }
            inputFromUser.close();
            inputFromServer.close();
            outputToServer.close();
            socket.close();
        }catch(UnknownHostException u){
            System.out.println(u);
            return;
        }catch(IOException i){
            System.out.println(i);
            return;
        }
    }
    public static void main(String args[]){
        Client client = new Client("127.0.0.1", 5000);
    }
}
```

```
Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

D:\Utsav\Java\Lab - 7 - Network Programming - Sockets\TCP>java Server
Server started
Waiting for a client ...
Client accepted
Hi
Java
Over
Closing connection

D:\Utsav\Java\Lab - 7 - Network Programming - Sockets\TCP>
```

```
Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

D:\Utsav\Java\Lab - 7 - Network Programming - Sockets\TCP>java Client
Connected
Hi
Echo from Server: Hi
Java
Echo from Server: Java
Over
Echo from Server: Over

D:\Utsav\Java\Lab - 7 - Network Programming - Sockets\TCP>
```

UDP Unicasting Code:

Sender:

```
import java.net.*;
public class Sender{
    public static void main(String [] args) throws Exception{
        DatagramSocket ds = new DatagramSocket();
        String str1 = "Welcome java";
        String str2 = "Bye Bye Java";
        InetAddress ip = InetAddress.getByName("127.0.0.1");
        DatagramPacket dp1 = new DatagramPacket(str1.getBytes(),
                                str1.length(), ip, 3000);
        DatagramPacket dp2 = new DatagramPacket(str2.getBytes(),
                                str2.length(), ip, 3000);
        ds.send(dp1);
        ds.send(dp2);
        ds.close();
    }
}
```

Receiver:

```
import java.net.*;
public class Receiver{
    public static void main(String[] args) throws Exception{
        DatagramSocket ds = new DatagramSocket(3000);
        byte[] buf1 = new byte[1024];
        byte[] buf2 = new byte[1024];
        DatagramPacket dp1 = new DatagramPacket(buf1, 1024);
        ds.receive(dp1);
        String str1 = new String(dp1.getData(), 0, dp1.getLength());
        DatagramPacket dp2 = new DatagramPacket(buf2, 1024);
        ds.receive(dp2);
        String str2 = new String(dp2.getData(), 0, dp2.getLength());
        System.out.println(str1);
        System.out.println(str2);
        ds.close();
    }
}
```

```
Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

D:\Utsav\Java\Lab - 7 - Network Programming - Sockets\UDP\Uni Casting>java Sender

D:\Utsav\Java\Lab - 7 - Network Programming - Sockets\UDP\Uni Casting>
```

```
Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

D:\Utsav\Java\Lab - 7 - Network Programming - Sockets\UDP\Uni Casting>java Receiver
Welcome java
Bye Bye Java

D:\Utsav\Java\Lab - 7 - Network Programming - Sockets\UDP\Uni Casting>
```

UDP Multicasting Code:

Sender:

```
import java.io.IOException;
import java.net.InetAddress;
import java.net.MulticastSocket;
import java.net.DatagramPacket;
public class MulticastingSender{
    public static void main(String[] args) throws InterruptedExeption,
                                                   IOExcreption{
        String group = "226.4.5.6";
        MulticastSocket ms = new MulticastSocket();
        String message1 = "Hello using Multicast";
        String message2 = "Bye bye using Multicast";
        DatagramPacket dp1 = new DatagramPacket(message1.getBytes(),
                             message1.length(), InetAddress.getByName
                             (group), 5000);
        ms.send(dp1);
        DatagramPacket dp2 = new DatagramPacket(message2.get
                             Bytes(), message2.length(), InetAddress.
                             getByName(group), 5000);
        ms.send(dp2);
        ms.close();
    }
}
```

Receiver:

```
import java.io.IOException;
import java.net.InetAddress;
import java.net.MulticastSocket;
import java.net.DatagramPacket;
public class MulticastingReceiver{
    public static void main(String[] args) throws InterruptedExeption,
                                                   IOExcreption{
        String group = "226.4.5.6";
        MulticastSocket ms = new MulticastSocket(5000);
        ms.joinGroup(InetAddress.getByName(group));
        byte[] buf = new byte[1024];
```

```
        DatagramPacket dp = new DatagramPacket(buf, 1024);
        ms.receive(dp);
        String str1 = new String(dp.getData(), 0, dp.getLength());
        System.out.println(str1);
        ms.receive(dp);
        String str2 = new String(dp.getData(), 0, dp.getLength());
        System.out.println(str2);
        ms.leaveGroup(InetAddress.getByName(group));
        ms.close();
    }
}
```

```
Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

D:\Utsav\Java\Lab - 7 - Network Programming - Sockets\UDP\Multi Casting>java MulticastingSender

D:\Utsav\Java\Lab - 7 - Network Programming - Sockets\UDP\Multi Casting>
```

```
Administrator: Command Pro

Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

D:\Utsav\Java\Lab - 7 - Network Programming - Sockets\UDP\Multi Casting>java MulticastingReceiver (226.4.5.6)
Hello using Multicast
Bye bye using Multicast

D:\Utsav\Java\Lab - 7 - Network Programming - Sockets\UDP\Multi Casting>

Administrator: Command Pro

Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

D:\Utsav\Java\Lab - 7 - Network Programming - Sockets\UDP\Multi Casting>java MulticastingReceiver (226.4.5.6)
Hello using Multicast
Bye bye using Multicast

D:\Utsav\Java\Lab - 7 - Network Programming - Sockets\UDP\Multi Casting>S
```

MultiSock Code:

```
import java.io.IOExcephion;
import java.net.InetAddress;
import java.net.MulticastSocket;
import java.net.NetworkInterface;
import java.util.Enumeration;
public class MultiSock{
    public static void main(String[] args) throws IOExcephion
    {
        MulticastSocket ms = new MulticastSocket();
        InetAddress ip = InetAddress.getByName("224.168.1.124");
        ms.setTTL((byte) 25);
        ms.setTimeToLive(20);
        System.out.println("Time to Live : " + ms.getTimeToLive());
        NetworkInterface nif = NetworkInterface.getByIndex(2);
        Enumeration<InetAddress> enu = nif.getInetAddresses();
        InetAddress intadd = enu.nextElement();
        ms.setInterface(intadd);
        System.out.println("Interface : " + ms.getInterface());
        ms.setNetworkInterface(nif);
        System.out.println("Network Interface : " + ms.getNetworkI
                                                    Nterface());
        ms.setLoopbackMode(true);
        System.out.println("Loopback mode: " + ms.getLoopbackMode());
    }
}
```

```
Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

D:\Utsav\Java\Lab - 7 - Network Programming - Sockets\UDP\Multi Casting\For Self Study>java MultiSock
TTL : 20
Time to Live : 20
Interface : /0:0:0:0:0:0:0:1
Network Interface : name:loopback_0 (Software Loopback Interface 1)
Loopback mode : true
```

URL Class Code:

```
import java.net.*;
public class URLDemo{
    public static void main(String[] args){
        try{
            URL url = new URL("https://www.google.com/search
                                ?q=javatpoint&oq=javatpoint&
                                sourceid=chrome&ie=UTF-8");
            System.out.println("Protocal: "+url.getProtocal());
            System.out.println("Host Name: "+url.getHost());
            System.out.println("Port Number: "+url.getPort());
            System.out.println("Default Port Number: "+url.getDefault
                                                    Port());
            system.out.println("Query String: "+ url.getQuery());
            System.out.println("Path: "+ url.getPath());
            System.out.println("File: "+url.getFile());

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

```
Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

D:\Utsav\Java\lab 7\URL>java URLDemo
Protocol: https
Host Name: www.google.com
Port Number: -1
Default Port Number: 443
Query String: q=javatpoint&oq=javatpoint&sourceid=chrome&ie=UTF-8
Path: /search
File: /search?q=javatpoint&oq=javatpoint&sourceid=chrome&ie=UTF-8
```

HttpURL Connection Code:

```
import java.io.IOException;
import java.net.HttpURLConnection;
import java.net.URL;
public class HttpURLConnectionDemo{
    public static void main(String[] args){
        try{
            URL url = new URL("http://www.javatpoint.com/java-tutorial");
            HttpURLConnection huc = (HttpURLConnection) url.openConnection();
            System.out.println("Status Code = "+huc.getResponseCode());
            System.out.println("Message = "+huc.getResponseMessage());
            for(int i=1; ;i++){
                String key = huc.getHeaderFieldKey(i);
                String value = huc.getHeaderField(i);
                if(key==null && value==null){
                    break;
                }
                System.out.println(key+" = "+value);
            }
            huc.disconnect();
        }catch (IOException e){
            System.out.println("Error: "+e.getMessage());
        }
    }
}
```

```
D:\Utsav\Java\lab 7\URL>java HttpURLConnectionDemo
Status Code = 301
Message = Moved Permanently
Date = Sat, 02 Dec 2023 09:25:34 GMT
Transfer-Encoding = chunked
Connection = keep-alive
Cache-Control = max-age=3600
Expires = Sat, 02 Dec 2023 10:25:34 GMT
Location = https://www.javatpoint.com/java-tutorial
Report-To = {"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v3?s=hH4MtvhD44uDr%2FcFAuAypNbJJ%2BHHLq8yIJGin
k3sJYVgP3ytxDcMWEJ%2FIkNy3yEFBF5B%2FusaU7%2B0Dm3gZpJnia4dn9f9PCo5R4igR0rHjZqShxgwDzsahxv48fpOhRdKeWnbqmm%3D"}],"group":"
cf-nel","max_age":604800}
NEL = {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
Server = cloudflare
CF-RAY = 82f285f87cba249f-KTM
alt-svc = h3=":443"; ma=86400
```