Computer Networks
Experiment-6

Name: Ayush Jain
SAP ID: 60004200132
Batch: B2
Computer Engineering

Aim: To implement socket communication in java

Theory:

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server.

Java Socket programming is used for communication between the applications running on different JRE. Java Socket programming can be connection-oriented or connectionless. Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connectionless 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.

ServerSocket Class

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

User Datagram Protocol(UDP):

DatagramSockets are Java's mechanism for network communication via UDP instead of TCP. Java provides DatagramSocket to communicate over UDP instead of TCP. It is also built on top of IP. DatagramSockets can be used to both send and receive packets over the Internet.

One of the examples where UDP is preferred over TCP is the live coverage of TV channels. In this aspect, we want to transmit as many frames to live audience as possible not worrying about the loss of one or two frames. TCP being a reliable protocol add its own overhead while transmission. Another example where UDP is preferred is online multiplayer gaming. In games like counter-strike or call of duty, it is not necessary to relay all the information but the most important ones. It should also be noted that most of the applications in real life uses careful blend of both UDP and TCP; transmitting the critical data over TCP and rest of the data via UDP.

Code:

(Server Side):

```
import java.net.*; import

java.io.*;



public class Server

{

            //initialize socket and input stream

private Socket     socket = null; private
```

```
ServerSocket server = null; private
DataInputStream in         = null;

        // constructor with port
        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");


                        in = new DataInputStream(
new BufferedInputStream(socket.getInputStream()));

                        String line = "";
        while (!line.equals("Over"))
```

```
                    {
                              try
                              {
                                        line = in.readUTF();
                                        System.out.println(line);

                              }
                              catch(IOException i)
                              {
                                        System.out.println(i);
                              }
                    }
                    System.out.println("Closing connection");


                    socket.close();
in.close();
          }
          catch(IOException i)
          {
                    System.out.println(i);
          }
}
```

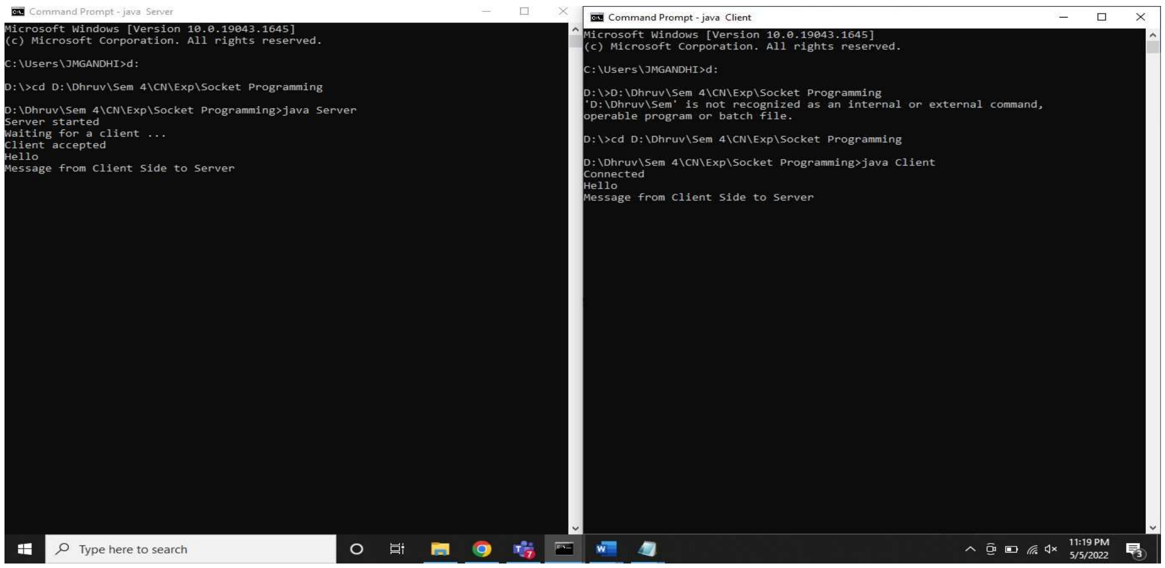
```
            }
            public static void main(String args[])

            {

                        Server server = new Server(5000);

            }

}
```

(Client Side):

```
import java.net.*;
import java.io.*;

public class Client
{

            private Socket socket                  = null;
private BufferedReader input = null;
            private DataOutputStream out      = null;

            public Client(String address, int port)
            {

                        try
                        {
                                    socket = new Socket(address, port);
                                    System.out.println("Connected");



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

                                    out = new DataOutputStream(socket.getOutputStream());
                        }
                        catch(UnknownHostException u)
                        {
                                    System.out.println(u);
                        }
                        catch(IOException i)
```

```
                {
                        System.out.println(i);
                }


                String line = "";


                while (!line.equals("Over"))
                {
        try
                        {
                                line = input.readLine();
out.writeUTF(line);
                        }
                        catch(IOException i)
                        {
                                System.out.println(i);
                        }
                }

                try
                {
                        input.close();
out.close();                            socket.close();
                }
                catch(IOException i)
                {
                        System.out.println(i);
                }
        }

        public static void main(String args[])
        {
                Client client = new Client("127.0.0.1", 5000);
        }
}
```

Output:



```
Microsoft Windows [Version 10.0.19043.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\JMGANDHI>d:

D:\>cd D:\Dhruv\Sem 4\CN\Exp\Socket Programming

D:\Dhruv\Sem 4\CN\Exp\Socket Programming>java Server
Server started
Waiting for a client ...
Client accepted
Hello
Message from Client Side to Server
```

```
Microsoft Windows [Version 10.0.19043.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\JMGANDHI>d:

D:\>D:\Dhruv\Sem 4\CN\Exp\Socket Programming
'D:\Dhruv\Sem' is not recognized as an internal or external command,
operable program or batch file.

D:\>cd D:\Dhruv\Sem 4\CN\Exp\Socket Programming

D:\Dhruv\Sem 4\CN\Exp\Socket Programming>java Client
Connected
Hello
Message from Client Side to Server
```

Code (UDP):

(Server Side):

```
// UDP Server side import
java.io.IOException; import
java.net.DatagramPacket; import
java.net.DatagramSocket; import
java.net.InetAddress; import
java.net.SocketException;


public class udpBaseServer_2
{
        public static void main(String[] args) throws IOException
        {
```

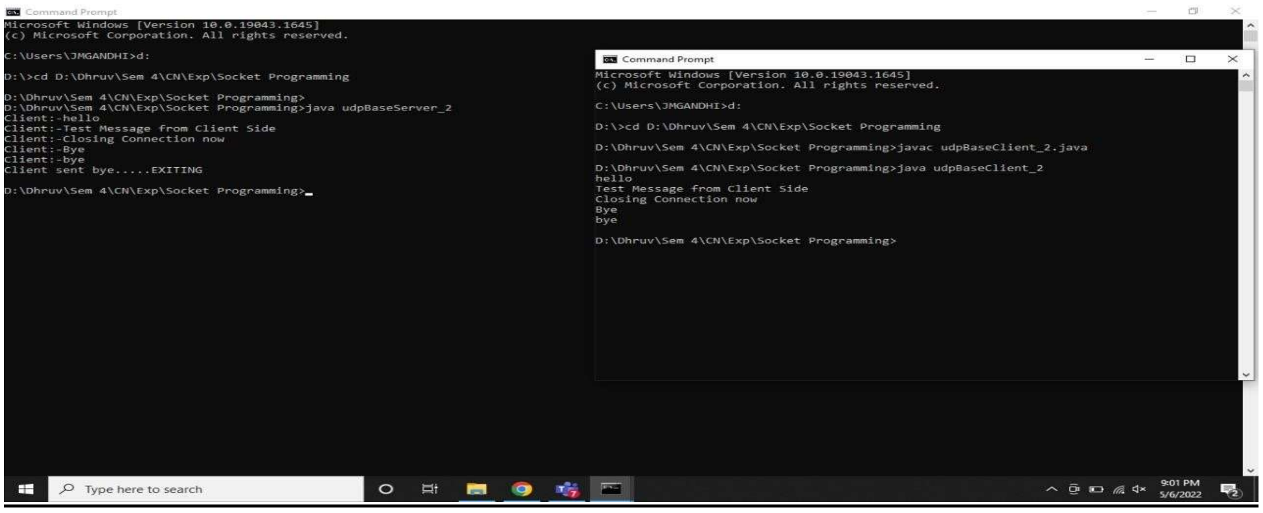
```
                    // Step 1 : Create a socket to listen at port 1234
DatagramSocket ds = new DatagramSocket(1234);                    byte[]
receive = new byte[65535];

                    DatagramPacket DpReceive = null;
while (true)
                    {

                              DpReceive = new DatagramPacket(receive, receive.length);

                              ds.receive(DpReceive);

                              System.out.println("Client:-" + data(receive));

                               // Exit the server if the client sends "bye"
                               if (data(receive).toString().equals("bye"))
                                                       {
                                        System.out.println("Client sent bye.....EXITING");
                                        break;
                              }

                              receive = new byte[65535];
                    }
          }

          public static StringBuilder data(byte[] a)
          {
                    if (a == null)
                              return null;
                    StringBuilder ret = new StringBuilder();
```

```
                int i = 0;
                while (a[i] != 0)
                {
                        ret.append((char) a[i]);
                        i++;
                }
                return ret;
        }
}
```

(Client Side):

```
// UDP Client side import
java.io.IOException; import
java.net.DatagramPacket; import
java.net.DatagramSocket; import
java.net.InetAddress; import
java.util.Scanner;


public class udpBaseClient_2
{
        public static void main(String args[]) throws IOException
        {
                Scanner sc = new Scanner(System.in);


                DatagramSocket ds = new DatagramSocket();


                InetAddress ip = InetAddress.getLocalHost();
byte buf[] = null;


                // loop while user not enters "bye"
                while (true)
                {
                        String inp = sc.nextLine();

                        buf = inp.getBytes();

                        DatagramPacket DpSend =
                                new DatagramPacket(buf, buf.length, ip, 1234);

                        ds.send(DpSend);

                        // break the loop if user enters "bye"

                if (inp.equals("bye"))

                                break;
                }
        }
}
```

<u>**Output:**</u>



```
Microsoft Windows [Version 10.0.19043.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\JMGANDHI>d:

D:\>cd D:\Dhruv\Sem 4\CN\Exp\Socket Programming

D:\Dhruv\Sem 4\CN\Exp\Socket Programming>
D:\Dhruv\Sem 4\CN\Exp\Socket Programming>java udpBaseServer_2
Client:-hello
Client:-Test Message from Client Side
Client:-Closing Connection now
Client:-Bye
Client:-bye
Client sent bye.....EXITING

D:\Dhruv\Sem 4\CN\Exp\Socket Programming>_
```

```
Microsoft Windows [Version 10.0.19043.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\JMGANDHI>d:

D:\>cd D:\Dhruv\Sem 4\CN\Exp\Socket Programming

D:\Dhruv\Sem 4\CN\Exp\Socket Programming>javac udpBaseClient_2.java

D:\Dhruv\Sem 4\CN\Exp\Socket Programming>java udpBaseClient_2
hello
Test Message from Client Side
Closing Connection now
Bye
bye

D:\Dhruv\Sem 4\CN\Exp\Socket Programming>
```

<u>**Conclusion:**</u> We have Successfully implemented Socket programming using TCP and UDP protocols in JAVA.