

**EXPERIMENT NO. 8**

**SEMESTER: V**

**DATE OF PERFORMANCE: 25<sup>th</sup> September 2024**

**SUBJECT: CN LAB**

**DATE OF SUBMISSION: 05<sup>th</sup> October 2024**

**NAME OF THE STUDENT: Dwayne George Nixon ROLL NO: 21**

<b>AIM</b>	Socket Programming using UDP
<b>LEARNING OBJECTIVE</b>	Students will be able to develop a chat application using UDP protocols.
<b>LEARNING OUTCOME</b>	The students will be able to write client server chat application program using TCP.
<b>COURSE OUTCOME</b>	CSL502.4: Illustrate socket programming for TCP/UDP connections for demonstrating networking concepts
<b>PROGRAM OUTCOME</b>	PO1,PO2,PO3,PO4,PO5,PO9,PO10,PSO1,PSO2,PSO3
<b>BLOOM'S TAXONOMY LEVEL</b>	Analyze
<b>THEORY</b>	<p><b>Java Socket Programming</b></p> <ul style="list-style-type: none"><li>• Java Socket programming is used for communication between the applications running on different JRE.</li><li>• Java Socket programming can be connection-oriented or connection-less.</li><li>• Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming.</li></ul> <p>The client in socket programming must know two information: a. IP Address of Server, and b. Port number.</p> <p>Here, we are going to make one-way client and server communication. In this application, client sends a message to the server, server reads the message and prints it. Here, two classes are being used: Socket and ServerSocket. The Socket class is used to communicate client and server. Through this class, we can read and write message. The ServerSocket class is used at server-side. The accept() method of ServerSocket class blocks the console until the client is connected. After the successful connection of client, it returns the instance of Socket at server-side.</p> <p><b>#Socket class</b> A socket is simply an endpoint for communications between the machines. The Socket class can be used to create a socket.</p> <p><b>#ServerSocket class</b> The ServerSocket class can be used to create a server socket. This object is used to establish communication with the clients.</p>

### Creating Server:

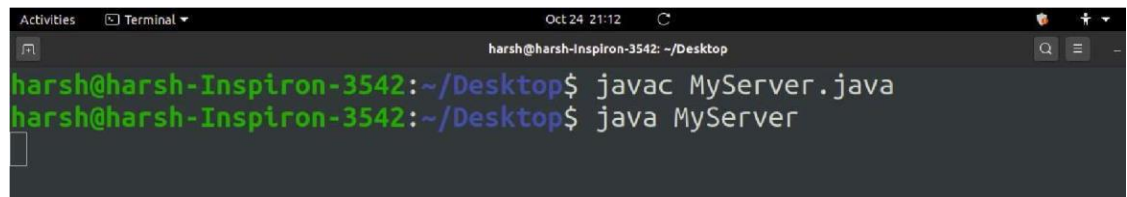
To create the server application, we need to create the instance of ServerSocket class. Here, we are using 6666 port number for the communication between the client and server. You may also choose any other port number. The accept() method waits for the client. If the client connects with the given port number, it returns an instance of Socket. `ServerSocket ss=new ServerSocket(6666);`  
`Socket s=ss.accept();` //establishes connection and waits for the client

### Creating Client:

To create the client application, we need to create the instance of Socket class. Here, we need to pass the IP address or hostname of the Server and a port number. Here, we are using "localhost" because our server is running on the same system.  
`Socket s=new Socket("localhost",6666);`

### Output:

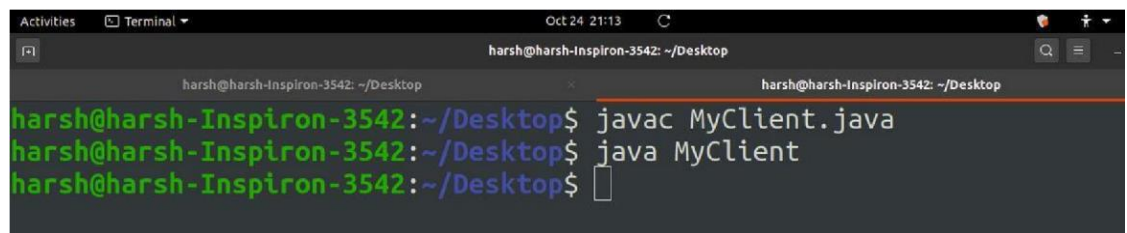
To execute this program, open two command prompts and execute each program at each command prompt as displayed in the below figures. First run MyServer.java file in terminal/cmd,

A terminal window titled 'harsh@harsh-Inspiron-3542: ~/Desktop' showing the execution of a Java program. The prompt is 'harsh@harsh-Inspiron-3542:~/Desktop\$'. The first command is 'javac MyServer.java' and the second is 'java MyServer'. The cursor is on the line following the second command.

```
harsh@harsh-Inspiron-3542:~/Desktop$ javac MyServer.java
harsh@harsh-Inspiron-3542:~/Desktop$ java MyServer
```

Running MyServer.java

Then in new terminal/cmd run MyClient.java file,

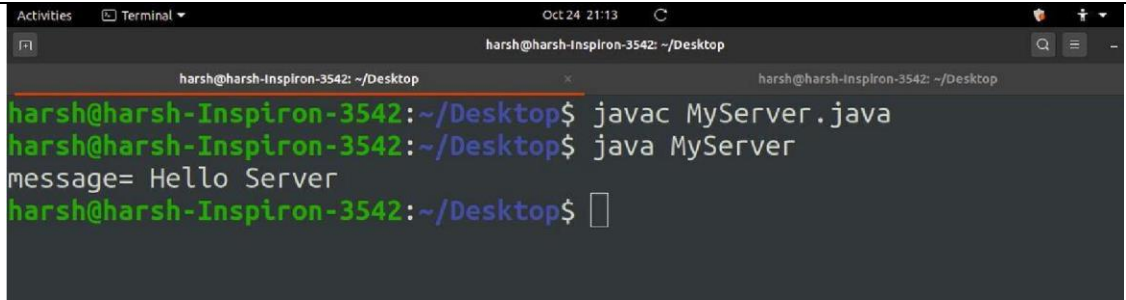
A terminal window titled 'harsh@harsh-Inspiron-3542: ~/Desktop' showing the execution of a Java program. The prompt is 'harsh@harsh-Inspiron-3542:~/Desktop\$'. The first command is 'javac MyClient.java' and the second is 'java MyClient'. The cursor is on the line following the second command.

```
harsh@harsh-Inspiron-3542:~/Desktop$ javac MyClient.java
harsh@harsh-Inspiron-3542:~/Desktop$ java MyClient
```

Running MyClient.java

As soon as you run MyClient program a message is sent to server and displayed in MyServer Terminal/CMD as shown below,

**Don Bosco Institute of Technology, Kurla**  
**Academic Year 2024-25**



```
harsh@harsh-Inspiron-3542: ~/Desktop$ javac MyServer.java
harsh@harsh-Inspiron-3542: ~/Desktop$ java MyServer
message= Hello Server
harsh@harsh-Inspiron-3542: ~/Desktop$
```

Message displayed in MyServer after running MyClient

**LAB EXERCISE**

Write a program to build chat application using SOCKET programming (UDP).

**Client Side Program: udpBaseClient\_2.java**

```
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);

        // Step 1: Create the socket object for carrying the data.
        DatagramSocket ds = new DatagramSocket();
        InetAddress ip = InetAddress.getLocalHost();
        byte buf[] = null;

        // Loop while user has not entered "bye"
        while (true) {
            String inp = sc.nextLine();

            // Convert the String input into the byte array.
            buf = inp.getBytes();

            // Step 2: Create the DatagramPacket for sending the data.
            DatagramPacket DpSend = new DatagramPacket(buf, buf.length, ip, 1234);

            // Step 3: Invoke the send call to actually send the data.
            ds.send(DpSend);

            // Break the loop if user enters "bye"
            if (inp.equals("bye")) {
                break;
            }
        }
    }
}
```

```
    }  
    }  
    sc.close();  
    ds.close();  
    }  
}
```

**Server Side Program: udpBaseServer\_2.java**

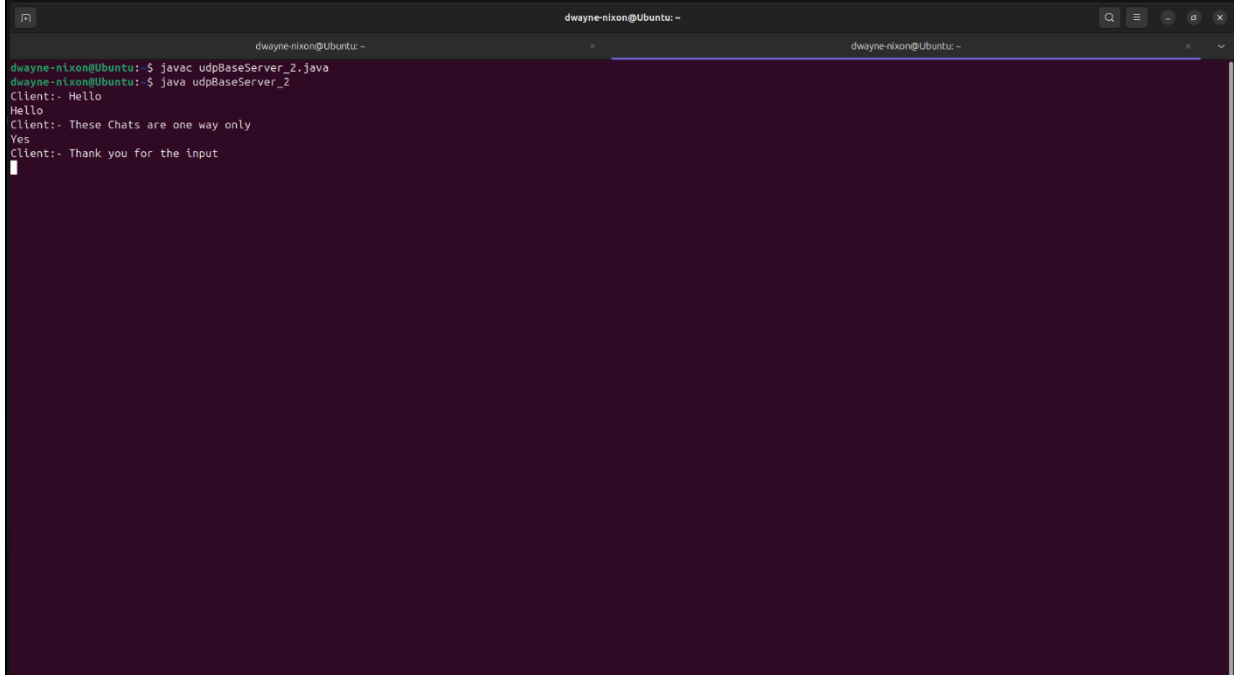
```
import java.io.IOException;  
import java.net.DatagramPacket;  
import java.net.DatagramSocket;  
import java.net.InetAddress;  
import java.net.SocketException;  
  
public class udpBaseServer_2 {  
    public static void main(String[] args) throws IOException {  
        // Step 1: Create a socket to listen at port 1234  
        DatagramSocket ds = new DatagramSocket(1234);  
        byte[] receive = new byte[65535];  
  
        DatagramPacket DpReceive = null;  
  
        while (true) {  
            // Step 2: Create a DatagramPacket to receive the data.  
            DpReceive = new DatagramPacket(receive, receive.length);  
  
            // Step 3: Receive the data in byte buffer.  
            ds.receive(DpReceive);  
  
            System.out.println("Client:- " + data(receive));  
  
            // Exit the server if the client sends "bye"  
            if (data(receive).toString().trim().equals("bye")) {  
                System.out.println("Client sent bye... EXITING");  
                break;  
            }  
  
            // Clear the buffer after every message.  
            receive = new byte[65535];  
        }  
        ds.close();  
    }  
  
    // A utility method to convert the byte array data into a string representation.  
    public static StringBuilder data(byte[] a) {  
        if (a == null) return null;  
        StringBuilder ret = new StringBuilder();  
        int i = 0;
```

**Don Bosco Institute of Technology, Kurla**  
**Academic Year 2024-25**

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

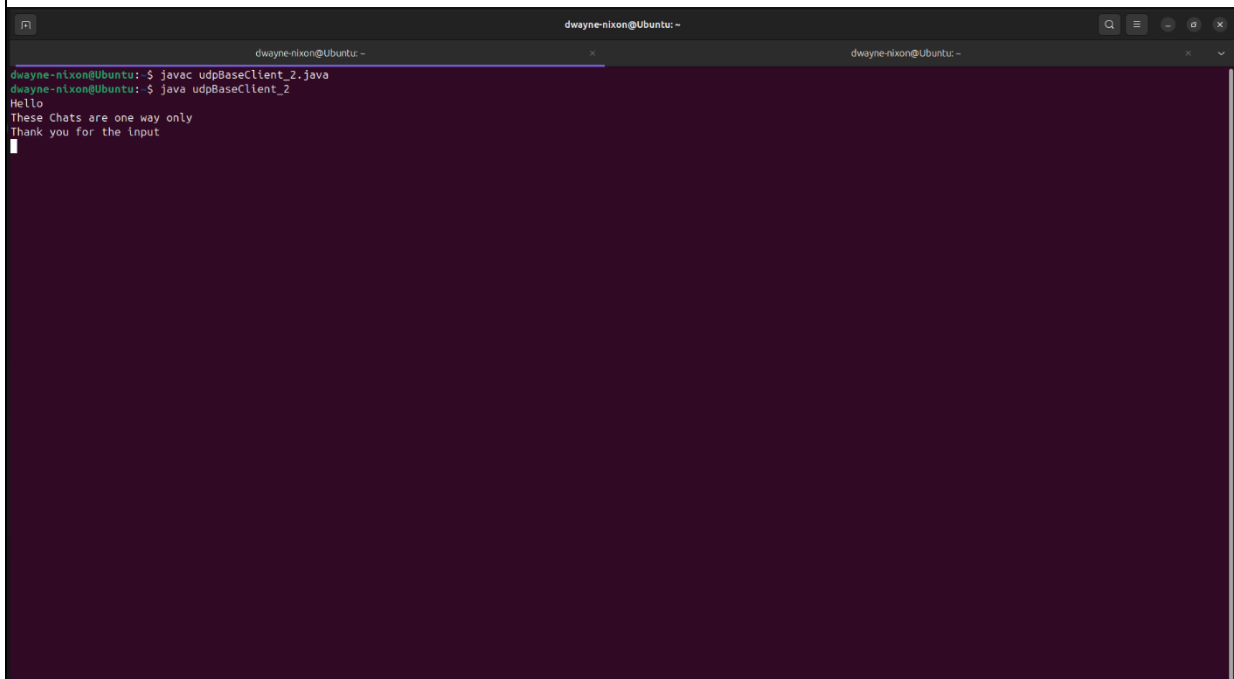
**Output:**

**Server:**



```
dwane-nixon@Ubuntu: ~  
$ javac udpBaseServer_2.java  
dwane-nixon@Ubuntu: ~$ java udpBaseServer_2  
Client:- Hello  
Hello  
Client:- These Chats are one way only  
Yes  
Client:- Thank you for the input
```

**Client:**



```
dwane-nixon@Ubuntu: ~  
$ javac udpBaseClient_2.java  
dwane-nixon@Ubuntu: ~$ java udpBaseClient_2  
Hello  
These Chats are one way only  
Thank you for the input
```

**Don Bosco Institute of Technology, Kurla**  
**Academic Year 2024-25**

<b>REFERENCES</b>	<ul style="list-style-type: none"><li>• B.A. Forouzan, “Data Communications and Networking”, TMH, Fourth Edition.</li><li>• <a href="https://www.tutorialspoint.com/unix_sockets/what_is_socket.htm">https://www.tutorialspoint.com/unix_sockets/what_is_socket.htm</a></li><li>• <a href="https://www.geeksforgeeks.org/working-udp-datagramsockets-java/">https://www.geeksforgeeks.org/working-udp-datagramsockets-java/</a> <a href="https://www.youtube.com/watch?v=UaM1JmQliTs">https://www.youtube.com/watch?v=UaM1JmQliTs</a></li></ul>
-------------------	---