Name: Shashwat Shah SAP ID:

600042201236

# Experiment No: 6

**Aim**: Write a client server socket program using TCP and using UDP.

## Theory:

Socket programming is a way of programming network communication between applications running on different computers or devices. In socket programming, sockets are endpoints of a two-way communication link, allowing data to be sent and received between applications. UDP and TCP are two common protocols used in socket programming.

### TCP [Transmission Control Protocol]:

TCP is a connection-oriented protocol that establishes a reliable communication channel between the two endpoints before transmitting data. This means that TCP ensures that all data is delivered to the destination in the correct order and without any loss or duplication. However, the reliability of TCP comes at the cost of increased overhead and latency, making it more suitable for applications that require data integrity, such as file transfer or email.

To use socket programming with TCP, an application needs to establish a connection with the remote endpoint by creating a TCP socket and calling its connect() function with the IP address and port number of the remote endpoint. Once the connection is established, the application can send and receive data using the socket's send() and recv() functions. TCP also provides additional features such as flow control, congestion control, and error recovery to ensure reliable data transmission.

## UDP[User Datagram protocol]:

UDP is a connectionless protocol that does not establish a reliable communication channel between the two endpoints. Instead, it simply sends packets of data to the destination, without any guarantee of delivery or order. This makes UDP a fast and lightweight protocol suitable for applications that require low latency and can tolerate data loss or duplication, such as video streaming or online gaming.

To use socket programming with UDP, an application needs to create a UDP socket and bind it to a specific port on the local machine. The application can then send and receive data packets using the socket's sendto() and recvfrom() functions, specifying the IP address and port number of the destination endpoint.

Here as we preform socket programming in java where we make use of package java.net.* which contains various methods and objects for socket programming

# Code[TCP]:

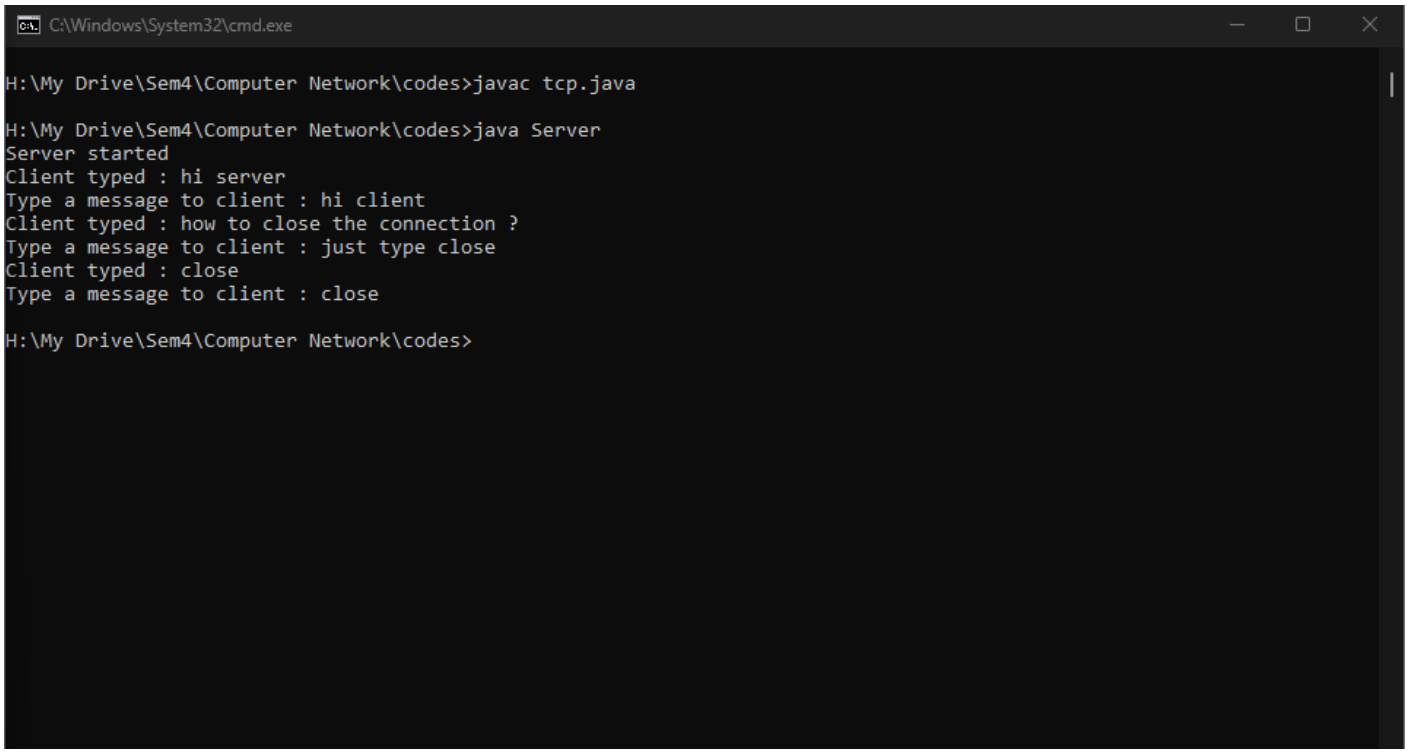
```
import
java.net.*;


class Server {

    public static void main(String args[]) throws Exception

    {

        ServerSocket ss = new

        ServerSocket(8000);Socket s =

        ss.accept();

        DataInputStream din = new DataInputStream(s.getInputStream());

        DataOutputStream dout = new

        DataOutputStream(s.getOutputStream()); BufferedReader br = new
```

```
        {
            str1 =  din.readUTF();
            System.out.println("Client typed : " + str1);
            System.out.print("Type a message to client : ");
            str2 = br.readLine();
            dout.writeUTF(str2);
            dout.flush();
        }
        din.close();
        s.close();
        ss.close();
    }
}
class Client {
    public static void main(String args[]) throws Exception
    {
        Socket s = new Socket("localhost", 8000);
        DataInputStream din = new DataInputStream(s.getInputStream());
        DataOutputStream dout = new DataOutputStream(s.getOutputStream());
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String str1 = "", str2 = "";
        System.out.println("Client started");
        while (!str1.equals("close"))
        {
            System.out.print("Type a message to server : ");
            str1 = br.readLine();
            dout.writeUTF(str1);
            dout.flush();
            str2 = din.readUTF();
```

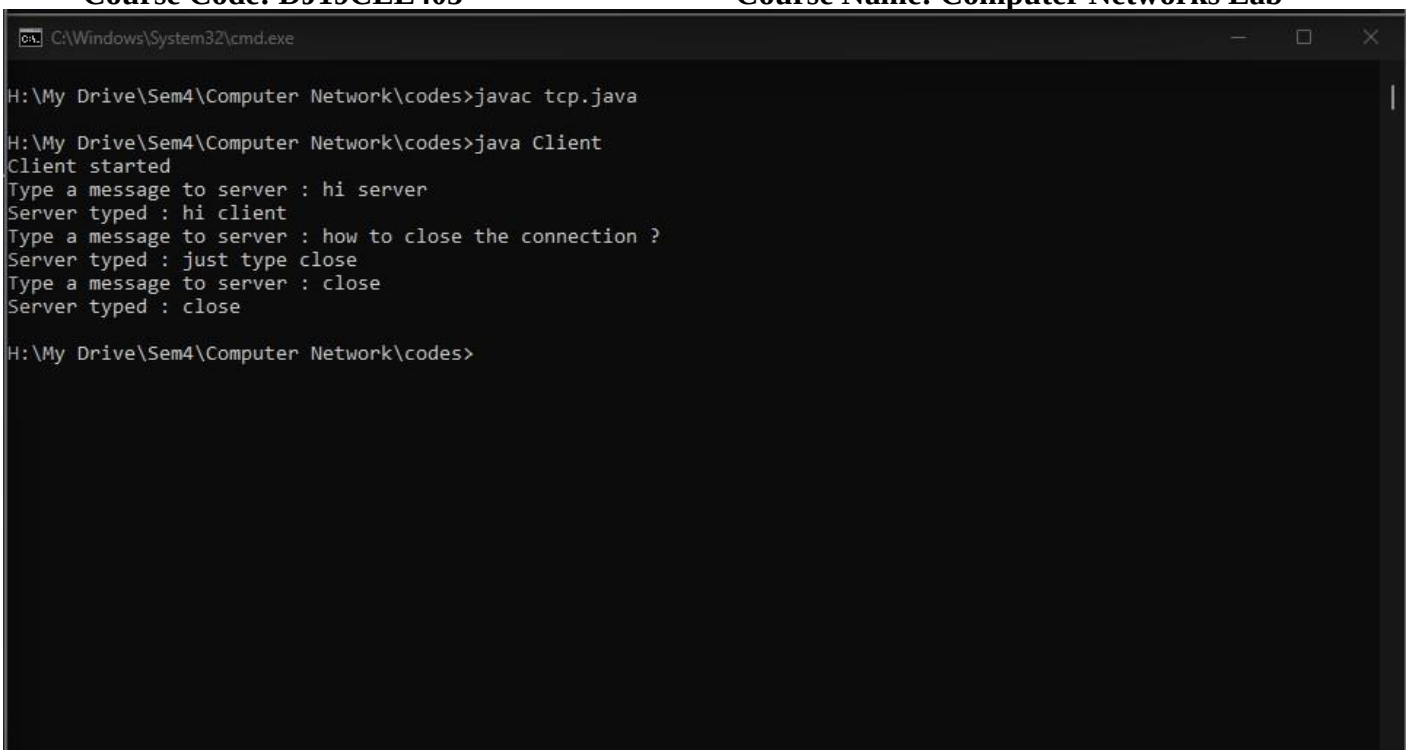
```
            System.out.println("Server typed : " + str2);

        }

        dout.close();

        s.close();

    }
}
```

# Output :

## Server side:




## *Client side:*




# Code[UDP]:

## *Server side:*

```
import
java.io
.*;
import
java.ne
t.*;


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

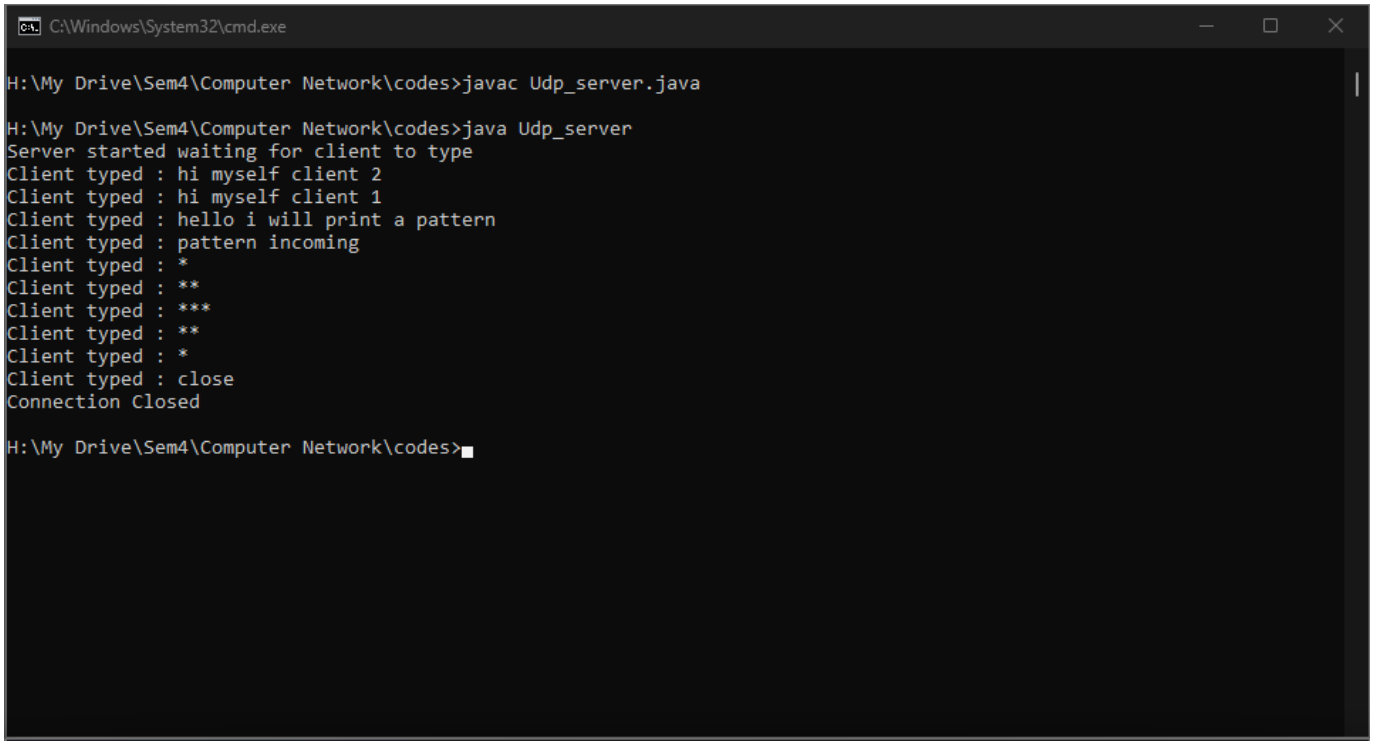
```
        while (true)
        {
            DpReceive = new DatagramPacket(receive, receive.length);
            dsock.receive(DpReceive);
            System.out.println("Client typed : " + data(receive));
            if (data(receive).toString().equals("close"))
            {
                System.out.println("Connection Closed ");
                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:

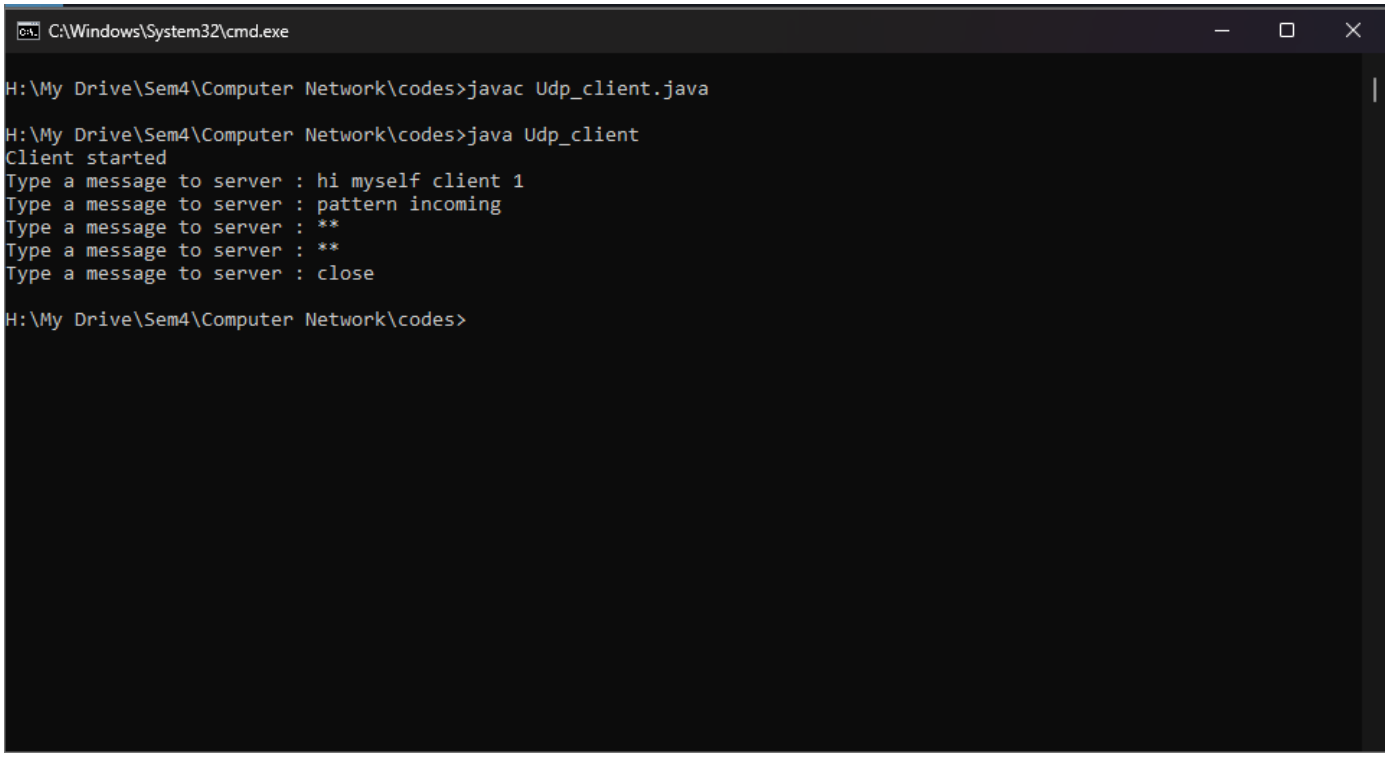
```
import java.io.*;
import java.net.*;
import java.util.*;

public class Udp_client
{
    public static void main(String args[]) throws IOException
    {
        Scanner sc = new Scanner(System.in);
        DatagramSocket dsock = new DatagramSocket();
        InetAddress ip = InetAddress.getLocalHost();
        byte byt[] = null;
        System.out.println("Client started");
        while (true)
        {
            System.out.print("Type a message to server : ");
            String inp = sc.nextLine();
            byt = inp.getBytes();
            DatagramPacket DpSend = new DatagramPacket(byt, byt.length, ip, 8000);
            dsock.send(DpSend);
            if (inp.equals("close"))
                break;
        }
    }
}
```

# Output :

## *Server side:*



## *Client side:*




# Conclusion:

Hence socket programming is performed using both UDP and TCP protocols in java.