THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Name: Manasi Bhagwat
Roll No: 50

Date: 12/04/2021

Lab Assignment No: 9

Aim: Socket Programming with C/Java – UDP Client, UDP Server.

Lab Outcome Attained: To implement client-server socket programs.

Theory:

UDP is a connection-less protocol that, unlike TCP, does not require any handshaking prior to sending or receiving data, which simplifies its implementation.

- No initial handshaking between the two processes, and therefore no need for a welcoming socket.
- No streams are attached to the sockets,
- The sending hosts creates "packets" by attaching the IP destination address and port number to each batch of bytes it sends.
- The server must extract IP address, port of sender from received datagram.
- The receiving process must unravel to received packet to obtain the packet's information bytes.

Program-

UDP – SERVER

Some packets are imported.

The java.io package is imported for providing a set of input and output streams used to read and write data to files or other input and output sources.

```
import java.io.*
```

The java.net package is imported for network related programming.

```
import java.net.*;
class udpserver1
{
public static void main(String args[]) throws Exception
{
```

This command creates a datagram socket at port 9876.

```
DatagramSocket serverSocket = new DatagramSocket(9876);
byte[] receiveData = new byte[1024];
byte[] sendData = new byte[1024];
int count;
while(true)
{
count=0;
```

This command creates Create space for received datagram.

DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);

This command receives the datagram from client.

serverSocket.receive(receivePacket);

The received datagram is converted to a string.

String sentence = new String(receivePacket.getData());

System.out.println("RECEIVED: " + sentence.trim());

The following commands get the IP address and Port number of client.

InetAddress IPAddress = receivePacket.getAddress();

int port = receivePacket.getPort();

The following commands count the number of vowels in input string.

```
String str= sentence.toLowerCase();

for (int i = 0; i < str.length(); i++)

{

    if(str.charAt(i) == 'a' || str.charAt(i) == 'e' || str.charAt(i) == 'i' ||
    str.charAt(i) == 'o' || str.charAt(i) == 'u')

{count++;  }

}

String s=String.valueOf(count);
```

The output string is converted to byte.

```
sendData = s.getBytes();
```

This command creates datagram to send the output to client.

```
DatagramPacket sendPacket =
new DatagramPacket(sendData, sendData.length, IPAddress, port);
```

This command writes out the datagram to the socket.

```
serverSocket.send(sendPacket);
}}}
```

UDP – CLIENT

The java.io package is imported for providing a set of input and output streams used to read and write data to files or other input and output sources.

```
import java.io.*
```

The java.net package is imported for network related programming.

```
import java.net.*;
class udpclient1
{
public static void main(String args[]) throws Exception
{
```

This command creates an input stream attached to socket.

```
BufferedReader inFromUser =
new BufferedReader(new InputStreamReader(System.in));
```

This command creates client socket.

```
DatagramSocket clientSocket = new DatagramSocket();
```

This command gets the IP address of the localhost.

```
InetAddress IPAddress = InetAddress.getByName("localhost");
byte[] sendData = new byte[1024];
byte[] receiveData = new byte[1024];
```

This command reads in line from socket.

```
String sentence = inFromUser.readLine();

sendData = sentence.getBytes();
```

This command creates a datagram to send to the server with input data, IP address and port number.

```
DatagramPacket sendPacket =

new DatagramPacket(sendData, sendData.length, IPAddress, 9876);
```

This command sends datagram to server.

```
clientSocket.send(sendPacket);
```

This command creates Create space for received datagram.

```
DatagramPacket receivePacket =

new DatagramPacket(receiveData, receiveData.length);
```

This command receives the datagram from server.

```
clientSocket.receive(receivePacket);
```

The received datagram is converted to a string.

```
String vcount = new String(receivePacket.getData());
```

The output is printed onto the screen.

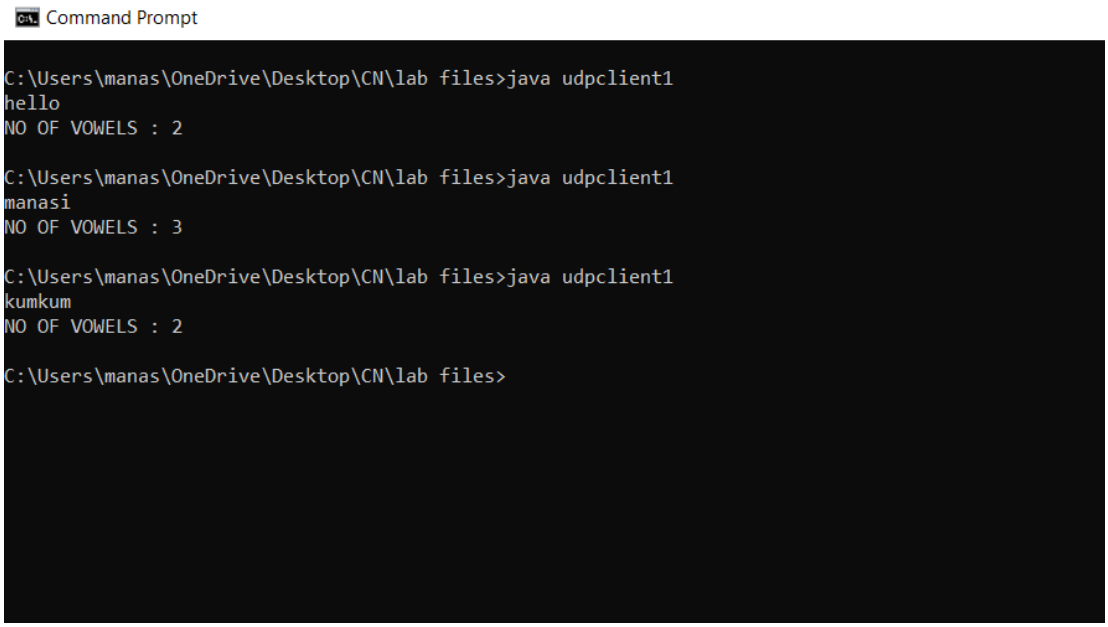
```
System.out.println("NO OF VOWELS : " + vcount.trim());
```

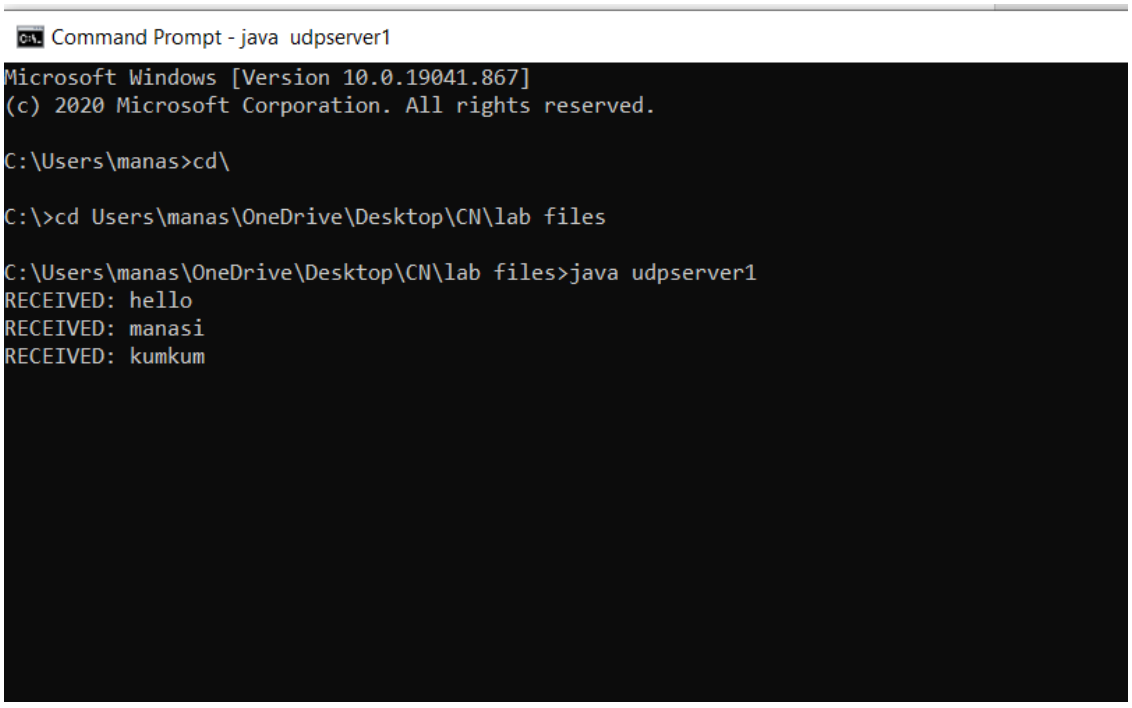
This command is used to close the connection with the server.

```
clientSocket.close();

}}
```

OUTPUT -





CONCLUSION

We have successfully understood and implemented Socket Programming with Java – UDP Client, UDP Server.