Date: 16 / 8 / 24

Lab Practical #07:

Study Client-Server Socket programming - TCP & UDP

Practical Assignment #07:

1. **Write a C/Java code for TCP Server-Client Socket Programming.**
2. **Write a C/Java code for UDP Server-Client Socket Programming.**

1. For TCP Server-Client:

- **TCP Server Program:**

```java
// A Java program for a Server
import java.net.*;
import java.io.*;

public class TCP_Server
{
    //initialize socket and input stream
    private Socket       socket = null;
    private ServerSocket server = null;
    private DataInputStream in   = null;

    // constructor with port
    public TCP_Server(int port)
    {
        // starts server and waits for a connection
        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");

            // takes input from the client socket
            in = new DataInputStream(
                new BufferedInputStream(socket.getInputStream()));

            String line = "";

            // reads message from client until "Over" is sent
            while (!line.equals("Over"))
```

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

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

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

    public static void main(String args[])
    {
        TCP_Server server = new TCP_Server(5000);
    }
}
```

- **TCP Client Program:**

```
// A Java program for a Client
import java.io.*;
import java.net.*;

public class TCP_Client {
    // initialize socket and input output streams
    private Socket socket = null;
    private DataInputStream input = null;
    private DataOutputStream out = null;

    // constructor to put ip address and port
```

Date: 16 / 8 / 24

```
    public TCP_Client(String address, int port)
    {
        // establish a connection
        try {
            socket = new Socket(address, port);
            System.out.println("Connected");

            // takes input from terminal
            input = new DataInputStream(System.in);

            // sends output to the socket
            out = new DataOutputStream(
                socket.getOutputStream());
        }
        catch (UnknownHostException u) {
            System.out.println(u);
            return;
        }
        catch (IOException i) {
            System.out.println(i);
            return;
        }

        // string to read message from input
        String line = "";

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

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

Date: 16 / 8 / 24

```
    }

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

2. For UDP Server-Client:

- **UDP Server Program:**

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;

public class UDP_Server {
    public static void main(String[] args) {
        try {
            // Create a DatagramSocket that listens on port 9876
            DatagramSocket serverSocket = new DatagramSocket(9876);
            System.out.println("Server started and listening on port 9876...");

            // Buffer to hold incoming datagrams
            byte[] receiveData = new byte[1024];

            while (true) {
                // Create a packet to receive data
                DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);

                // Receive the data from the client
                serverSocket.receive(receivePacket);

                // Extract the message and the client address
                String clientMessage = new String(receivePacket.getData(), 0,
receivePacket.getLength());
                System.out.println("Received from client: " + clientMessage);

                // Prepare a response
                String responseMessage = "Server received: " + clientMessage;
                byte[] responseData = responseMessage.getBytes();

                // Send the response back to the client
                DatagramPacket sendPacket = new DatagramPacket(
                        responseData, responseData.length,
```

Date: 16 / 8 / 24

```
                        receivePacket.getAddress(), receivePacket.getPort()
                );
                serverSocket.send(sendPacket);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

- **UDP Client Program:**

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class UDP_Client {
    public static void main(String[] args) {
        try {
            // Create a DatagramSocket
            DatagramSocket clientSocket = new DatagramSocket();

            // Get the IP address of the server
            InetAddress IPAddress = InetAddress.getByName("localhost");

            // Prepare the message to send
            String message = "Hello, Server!";
            byte[] sendData = message.getBytes();

            // Create a packet to send the data to the server
            DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, IPAddress, 9876);

            // Send the packet to the server
            clientSocket.send(sendPacket);

            // Buffer to hold the incoming response
            byte[] receiveData = new byte[1024];

            // Create a packet to receive the response
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
```

Date: 16 / 8 / 24

```
            // Receive the response from the server
            clientSocket.receive(receivePacket);

            // Extract and display the response
            String responseMessage = new String(receivePacket.getData(), 0,
receivePacket.getLength());
            System.out.println("FROM SERVER: " + responseMessage);

            // Close the socket
            clientSocket.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```