Student Name	**Pratik Manoj Dharam**
SRN No	31232438
Roll No	20
Program	Computer Engg.
Year	Third Year
Division	H
Subject	Computer Network Laboratory (BTECCE22506)
Assignment No	10

Assignment Number - 10

Title : Socket Programming for UDP Client, UDP Server.

Problem Statement : Implement a simple **UDP Client-Server** communication using **Socket Programming** in **Java**. The client sends a message to the server, and the server responds with the message prefixed by "Server received:". Both client and server should run on the same machine, using UDP as the communication protocol.

Theory :

- **Server**: The server should listen on a specific port for incoming UDP datagrams. When a client sends a message, the server should receive it, process the message by adding a prefix "Server received: ", and send the response back to the client.
- **Client**: The client should send a message to the server on a specified IP address and port using UDP and print the server's response.

Source code:

UDP Server code:

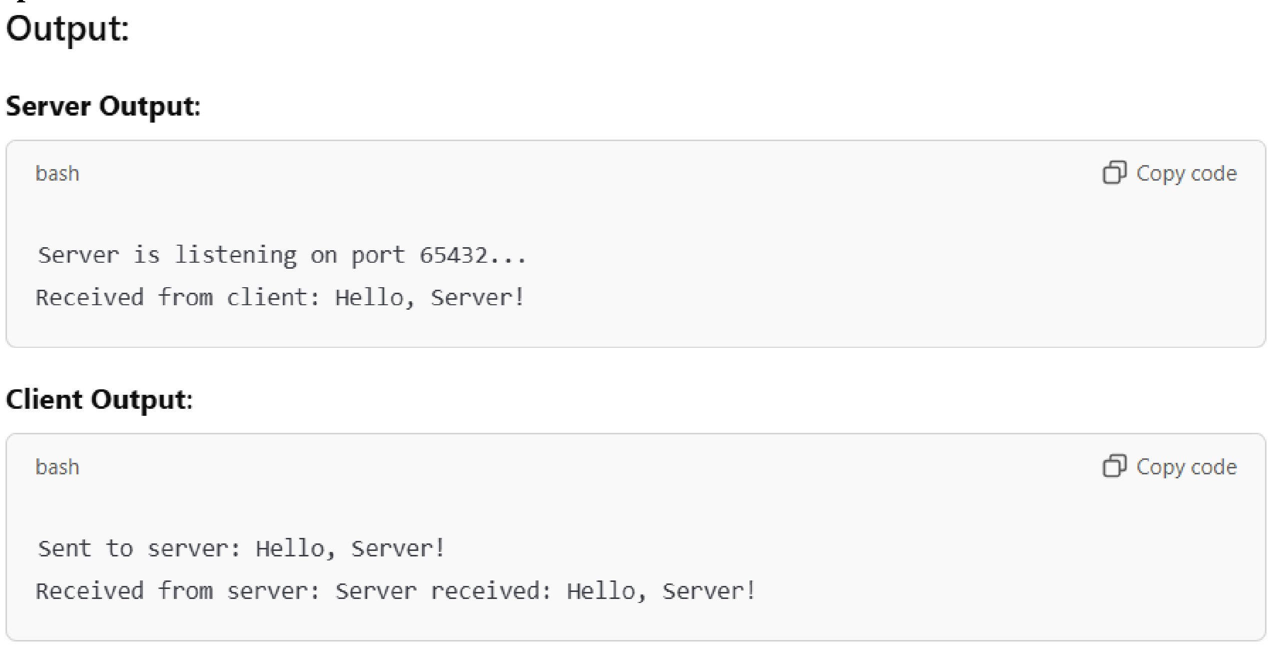
```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class UDPServer {
    public static void main(String[] args) {
        try {
            // Create a DatagramSocket to listen on port 65432
            DatagramSocket serverSocket = new DatagramSocket(65432);
            System.out.println("Server is listening on port 65432...");

            byte[] receiveBuffer = new byte[1024];
            byte[] sendBuffer;

            // Create a DatagramPacket to receive incoming data
            DatagramPacket receivePacket = new DatagramPacket(receiveBuffer, receiveBuffer.length);

            // Receive data from the client
            serverSocket.receive(receivePacket);
            String clientMessage = new String(receivePacket.getData(), 0, receivePacket.getLength());
            System.out.println("Received from client: " + clientMessage);

            // Get client's address and port
            InetAddress clientAddress = receivePacket.getAddress();
            int clientPort = receivePacket.getPort();

            // Prepare response
            String response = "Server received: " + clientMessage;
            sendBuffer = response.getBytes();
            // Send the response back to the client
            DatagramPacket sendPacket = new DatagramPacket(sendBuffer, sendBuffer.length,
clientAddress, clientPort);
            serverSocket.send(sendPacket);

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

TCP Client code:

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

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

            // Define server address and port
            InetAddress serverAddress = InetAddress.getByName("127.0.0.1");
            int serverPort = 65432;

            // Message to be sent to the server
            String message = "Hello, Server!";
            byte[] sendBuffer = message.getBytes();

            // Create a DatagramPacket for sending the message
            DatagramPacket sendPacket = new DatagramPacket(sendBuffer, sendBuffer.length,
serverAddress, serverPort);

            // Send the message to the server
            clientSocket.send(sendPacket);
            System.out.println("Sent to server: " + message);

            // Prepare a buffer to receive the server's response
            byte[] receiveBuffer = new byte[1024];
            DatagramPacket receivePacket = new DatagramPacket(receiveBuffer,
receiveBuffer.length);

            // Receive the response from the server
            clientSocket.receive(receivePacket);
            String serverResponse = new String(receivePacket.getData(), 0,
receivePacket.getLength());
```

```
            System.out.println("Received from server: " + serverResponse);

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

Output:



Output:

Server Output:

bash Copy code

```
Server is listening on port 65432...
Received from client: Hello, Server!
```

Client Output:

bash Copy code

```
Sent to server: Hello, Server!
Received from server: Server received: Hello, Server!
```

Conclusion:

This UDP Client-Server communication using Socket Programming in Java demonstrates how the server listens for incoming UDP datagrams and responds to the client's message. The client sends a message to the server and waits for a response. Unlike TCP, UDP is connectionless, so there is no need to establish a persistent connection between client and server, making it lightweight and faster, but with no guarantee of message delivery.

This implementation can be expanded to handle more advanced scenarios like handling larger data or multiple clients.