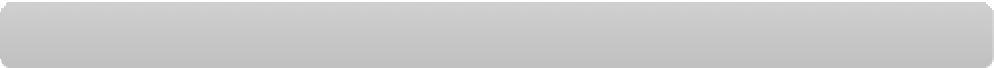




|  |  |
| --- | --- |
| **Student Name** | **Om Atul Mahadeokar** |
| **SRN No** | 202200340 |
| **Roll No** | 17 |
| **Program** | Computer Engg. |
| **Year** | Third Year |
| **Division** | B |
| **Subject** | Computer Network Laboratory (BTECCE21506) |
| **Assignment No** | 1 |



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 EnhancedUDPServer {

    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;

            while (true) {

                // Receive data from the client

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

                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);

                System.out.println("Response sent to client.");

                // Exit if the client message is "bye"

                if (clientMessage.trim().equalsIgnoreCase("bye")) {

                    System.out.println("Server exiting...");

                    break;

                }

            }

            // 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;

import java.util.Scanner;

public class EnhancedUDPClient {

    public static void main(String[] args) {

        try {

            DatagramSocket clientSocket = new DatagramSocket();

            InetAddress serverAddress = InetAddress.getByName("127.0.0.1");

            int serverPort = 65432;

            Scanner scanner = new Scanner(System.in);

            String message;

            byte[] sendBuffer;

            byte[] receiveBuffer = new byte[1024];

            while (true) {

                // Read message from user

                System.out.print("You: ");

                message = scanner.nextLine();

                sendBuffer = message.getBytes();

                // Send message to the server

                DatagramPacket sendPacket = new DatagramPacket(sendBuffer, sendBuffer.length, serverAddress, serverPort);

                clientSocket.send(sendPacket);

                System.out.println("Sent to server: " + message);

                // Exit if the message is "bye"

                if (message.equalsIgnoreCase("bye")) {

                    System.out.println("Ending conversation...");

                    break;

                }

                // Receive response from the server

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

                clientSocket.receive(receivePacket);

                String serverResponse = new String(receivePacket.getData(), 0, receivePacket.getLength());

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

            }

            // Close the socket

            clientSocket.close();

            scanner.close();

        } catch (Exception e) {

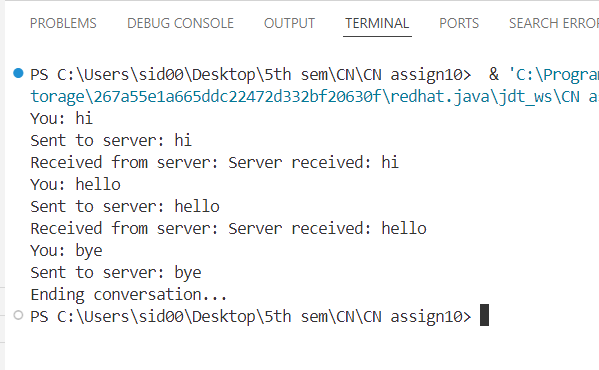
            e.printStackTrace();

        }

    }

}

**Output:**



**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.