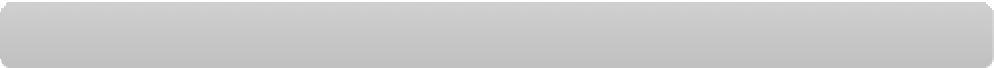




|  |  |
| --- | --- |
| **Student Name** | **Om Atul Mahadeokar** |
| **SRN No** | 202200340 |
| **Roll No** | 17 |
| **Program** | Computer Engg. |
| **Year** | Third Year |
| **Division** | B |
| **Subject** | Computer Network Laboratory (BTECCE21506) |
| **Assignment No** | 1 |



Assignment Number - 09

**Title :** Socket Programming for TCP Client and TCP Server.

**Problem Statement** : Implement a simple **TCP 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 TCP as the communication protocol.

# Theory :

**** Server: The server should listen on a specific port for incoming client connections. When a client connects, the server should accept the connection, receive a message, and respond by sending back the message prefixed with "Server received: ". The server should handle one connection at a time.

 Client: The client should connect to the server on the specified IP address and port, send a message, and print the server's response.

Requirements:

* TCP connection.
* The server should be able to accept a connection, receive a message from the client, and send a response back.
* The client should be able to send a message to the server and receive the server's response.

# Source Code:

# TCP Server code:

import java.io.\*;

import java.net.\*;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

public class EnhancedTCPServer {

    public static void main(String[] args) {

        try {

            ServerSocket serverSocket = new ServerSocket(65432);

            System.out.println("Server listening on port 65432...");

            Socket clientSocket = serverSocket.accept();

            System.out.println("Connected to client: " + clientSocket.getInetAddress());

            BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

            PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);

            // Send greeting message to the client

            String greeting = "Welcome to the Server! Type 'bye' to end the conversation.";

            out.println(greeting);

            String clientMessage;

            DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");

            // Continuously listen for messages until client says 'bye'

            while ((clientMessage = in.readLine()) != null) {

                System.out.println("[" + dtf.format(LocalDateTime.now()) + "] Client: " + clientMessage);

                // End conversation if client sends "bye"

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

                    out.println("Goodbye! Connection will be closed.");

                    break;

                }

                // Send response back to the client

                String response = "Server received: " + clientMessage;

                out.println("[" + dtf.format(LocalDateTime.now()) + "] " + response);

            }

            clientSocket.close();

            serverSocket.close();

            System.out.println("Connection closed.");

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}

TCP Client code:

import java.io.\*;

import java.net.\*;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

import java.util.Scanner;

public class EnhancedTCPClient {

    public static void main(String[] args) {

        try {

            Socket socket = new Socket("127.0.0.1", 65432);

            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

            // Read the server's greeting message

            System.out.println(in.readLine());

            Scanner scanner = new Scanner(System.in);

            DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");

            // Allow multiple messages to be sent

            String message;

            while (true) {

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

                message = scanner.nextLine();

                out.println(message);

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

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

                    break;

                }

                // Receive response from the server

                String serverResponse = in.readLine();

                System.out.println("[" + dtf.format(LocalDateTime.now()) + "] " + serverResponse);

            }

            socket.close();

            scanner.close();

        } catch (IOException e) {

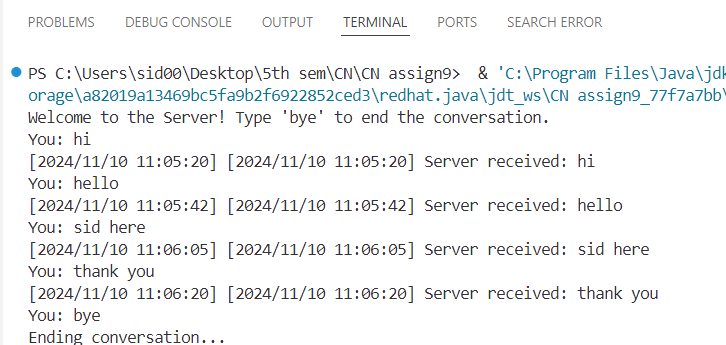
            e.printStackTrace();

        }

    }

}

**Output:**



**Conclusion:**

This **TCP Client-Server** communication using **Socket Programming in Java** demonstrates how the server listens for incoming connections and responds to client messages. The client connects to the server, sends a message, and receives a response. This implementation can be expanded for more advanced scenarios such as multi-threaded servers to handle multi