ITCSIU24092 - Phạm Hoàng Tuấn Tú

"Explain and run the UDPclient - UDPserver, TCPclient-TCPserver programs, capture the outputs."

- **UDPClient–UDPServer**: These programs demonstrate a client-server model using UDP, a connectionless, lightweight protocol. UDP doesn't guarantee delivery, order, or error checking, making it faster but less reliable than TCP. The server listens for incoming UDP datagrams (packets), and the client sends messages to the server.
- **TCPClient–TCPServer**: These programs demonstrate a client-server model using TCP, a connection-oriented protocol. TCP ensures reliable communication by establishing a connection (via a three-way handshake), guaranteeing delivery, order, and error correction. The server listens for incoming TCP connections, and the client connects to the server to exchange data.

Here is the code:

1. **UDPServer.java**

```
package socket_programing;

import java.net.*;

public class UDPServer {
    public static void main(String[] args) {
        // Define the port number the server will listen on
        int port = 10000;

        try {
            // Create a DatagramSocket to listen for incoming UDP packets
            DatagramSocket serverSocket = new DatagramSocket(port);
            System.out.println("UDP Server is running on port " + port + "...");

            // Buffer to store incoming data
            byte[] receiveData = new byte[1024];

            while (true) {
                // Create a DatagramPacket to receive the message from the client
                DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
                serverSocket.receive(receivePacket); // Wait for a packet

                // Convert the received data to a string
                String message = new String(receivePacket.getData(), 0,
receivePacket.getLength());
                System.out.println("Received from client: " + message);

                // Process the message: Convert to uppercase
                String response = message.toUpperCase();

                // Prepare to send the response back to the client
                byte[] sendData = response.getBytes();

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

```
                // Create a DatagramPacket to send the response
                DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, clientAddress, clientPort);
                serverSocket.send(sendPacket); // Send the response
                System.out.println("Sent to client: " + response);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

2. UDPClient.java

```
package socket_programing;

import java.net.*;

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

            // Define the server address and port
            InetAddress serverAddress = InetAddress.getByName("localhost");
            int serverPort = 10000;

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

            // Create a DatagramPacket to send the message
            DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, serverAddress, serverPort);
            clientSocket.send(sendPacket); // Send the message
            System.out.println("Sent to server: " + message);

            // Buffer to store the server's response
            byte[] receiveData = new byte[1024];
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
            clientSocket.receive(receivePacket); // Wait for the response

            // Convert the response to a string and print it
            String response = new String(receivePacket.getData(), 0,
receivePacket.getLength());
            System.out.println("Received from server: " + response);

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

3. TCPServer.java

```
package socket_programing;

import java.net.*;
import java.io.*;

public class TCPServer {
    public static void main(String[] args) {
        // Define the port number the server will listen on
        int port = 10000;

        try {
            // Create a ServerSocket to listen for incoming connections
            ServerSocket serverSocket = new ServerSocket(port);
            System.out.println("TCP Server is running on port " + port + "...");

            while (true) {
                // Accept a connection from a client
                Socket clientSocket = serverSocket.accept();
                System.out.println("Connected to client: " +
clientSocket.getInetAddress().getHostAddress());

                // Create input and output streams to communicate with the client
                BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                PrintWriter out = new PrintWriter(clientSocket.getOutputStream(),
true);

                // Read the message from the client
                String message = in.readLine();
                System.out.println("Received from client: " + message);

                // Process the message: Convert to uppercase
                String response = message.toUpperCase();

                // Send the response back to the client
                out.println(response);
                System.out.println("Sent to client: " + response);

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

4. TCPClient.java

```
package socket_programing;

import java.net.*;
import java.io.*;

public class TCPClient {
    public static void main(String[] args) {
```

```
        try {
            // Define the server address and port
            String serverAddress = "localhost";
            int serverPort = 10000;

            // Create a Socket to connect to the server
            Socket socket = new Socket(serverAddress, serverPort);
            System.out.println("Connected to server: " + serverAddress);

            // Create input and output streams to communicate with the server
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
            BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));

            // Send a message to the server
            String message = "Hello, TCP Server!";
            out.println(message);
            System.out.println("Sent to server: " + message);

            // Receive the response from the server
            String response = in.readLine();
            System.out.println("Received from server: " + response);

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

Here is the output of UDP:

```
UDP Server is running on port 10000...
Received from client: Hello, UDP Server!
Sent to client: HELLO, UDP SERVER!
```

```
Sent to server: Hello, UDP Server!
Received from server: HELLO, UDP SERVER!
```

Here is the output of TCP:

```
TCP Server is running on port 10000...
Connected to client: 127.0.0.1
Received from client: Hello, TCP Server!
Sent to client: HELLO, TCP SERVER!
```

```
Connected to server: localhost
Sent to server: Hello, TCP Server!
Received from server: HELLO, TCP SERVER!
```