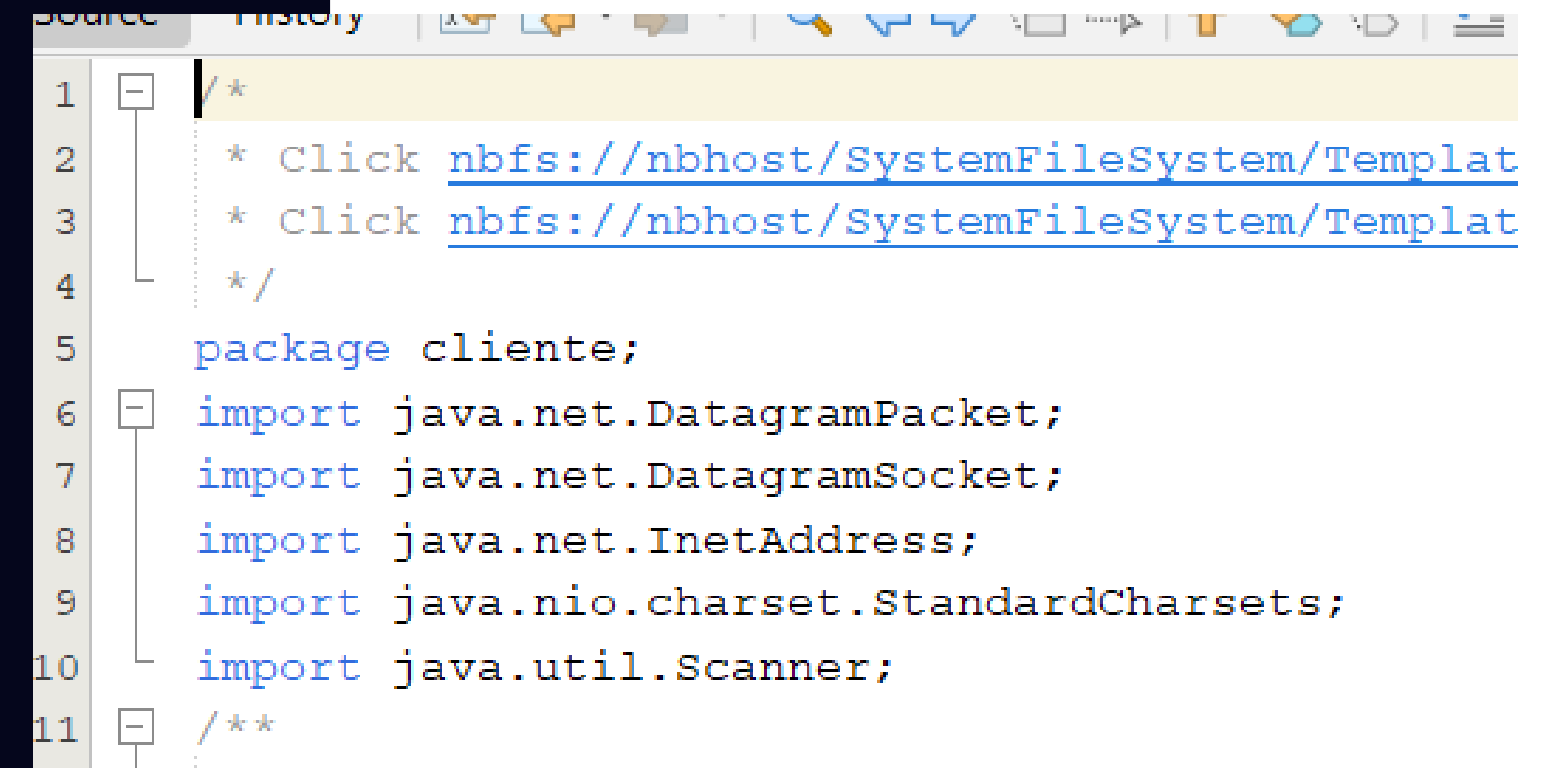


CODIGO SERVIDOR



Paquetes

- DatagramPacket → Representa el paquete de datos que se envía o recibe (mensaje + IP + puerto).
- DatagramSocket → Permite enviar y recibir los paquetes UDP.
- InetAddress → Indica o resuelve la dirección IP del servidor o cliente.
- StandardCharsets → Asegura que los textos se codifiquen correctamente (UTF-8).
- Scanner → Sirve para leer lo que escribe el usuario en la consola.



```
1  /*
2      * Click nbfs://nbhost/SystemFileSystem/Templat
3      * Click nbfs://nbhost/SystemFileSystem/Templat
4      */
5      package cliente;
6      import java.net.DatagramPacket;
7      import java.net.DatagramSocket;
8      import java.net.InetAddress;
9      import java.nio.charset.StandardCharsets;
10     import java.util.Scanner;
11     /**
```

Servidor

- Puerto: 9876
- Función: Escucha mensajes de clientes y responde con un eco (“ECO: mensaje”).
- Flujo del programa:
 1. Crea un socket UDP que escucha en el puerto 9876.
 2. Espera recibir un paquete del cliente.
 3. Lee el mensaje recibido.
 4. Muestra el mensaje en consola.
 5. Envía una respuesta al mismo cliente.
 6. Repite el proceso indefinidamente.

```
1 public static void main(String[] args) {  
2     final int PORT = 9876;  
3     final int BUFFER_SIZE = 1024;  
4  
5     try (DatagramSocket serverSocket = new DatagramSocket(port: PORT)) {  
6         System.out.println("Servidor UDP iniciado en el puerto " + PORT + ". Esperando paquetes...");  
7  
8         byte[] receiveBuffer = new byte[BUFFER_SIZE];  
9  
10        while (true) {  
11            DatagramPacket receivePacket = new DatagramPacket(buf: receiveBuffer, length: receiveBuffer.length);  
12            serverSocket.receive(p: receivePacket);  
13  
14            String message = new String(bytes: receivePacket.getData(), offset: 0, length: receivePacket.getLength(), charset: StandardCharsets.UTF_8);  
15            InetAddress clientAddress = receivePacket.getAddress();  
16            int clientPort = receivePacket.getPort();  
17  
18            System.out.printf(format: "Recibido de %s:%d -> %s\n", args: clientAddress.getHostAddress(), args: clientPort, args: message);  
19  
20            String response = "ECO: " + message;  
21            byte[] sendBuffer = response.getBytes(charset: StandardCharsets.UTF_8);  
22            DatagramPacket sendPacket = new DatagramPacket(buf: sendBuffer, length: sendBuffer.length, address: clientAddress, port: clientPort);  
23            serverSocket.send(p: sendPacket);  
24        }  
25    }  
26 }
```

Cliete

- Se conecta al servidor UDP en el puerto 9876.
- Permite al usuario escribir mensajes por consola.
- Envía cada mensaje al servidor y espera la respuesta (eco).
- Si el usuario escribe “salir”, el programa termina.
- Si no recibe respuesta en 5 segundos, muestra un mensaje de tiempo agotado.
-

```
public static void main(String[] args) {  
    // TODO code application logic here  
    final String SERVER_HOST = "localhost";  
    final int SERVER_PORT = 9876;  
    final int BUFFER_SIZE = 1024;  
  
    try (DatagramSocket socket = new DatagramSocket();  
        Scanner scanner = new Scanner(System.in, charset: StandardCharsets.UTF_8)) {  
  
        InetAddress serverAddress = InetAddress.getByName(host: SERVER_HOST);  
        System.out.println(x: "Cliente UDP. Escribe mensajes para enviar al servidor (escribe 'salir' para terminar).");  
  
        while (true) {  
            System.out.print(s: "> ");  
            String line = scanner.nextLine();  
            if (line == null) break;  
            if ("salir".equalsIgnoreCase(anotherString: line.trim())) {  
                System.out.println(x: "Saliendo...");  
                break;  
            }  
  
            byte[] sendBuffer = line.getBytes(charset: StandardCharsets.UTF_8);  
            DatagramPacket sendPacket = new DatagramPacket(buf: sendBuffer, length: sendBuffer.length, address: serverAddress, port: SERVER_PORT);  
            socket.send(p: sendPacket);  
        }  
    }  
}
```

Conclusion

El programa UDP demuestra cómo establecer una comunicación sencilla entre un cliente y un servidor mediante el envío y recepción de paquetes de datos.

A través del uso de DatagramSocket y DatagramPacket, se logra un intercambio rápido de mensajes sin necesidad de una conexión permanente.

Aunque UDP no garantiza la entrega de los datos, su velocidad y simplicidad lo hacen ideal para aplicaciones donde la rapidez es más importante que la fiabilidad.