Programación de servicios y procesos

Act3.3. Transferencia de datos por UDP

Francisco José García Cutillas | 2FPGS_DAM

Índice

Ejercicio 1

Transferencia de datos por UDP.

Desarrollar una aplicación en Java que transmite números desde un cliente a un servidor mediante el uso de sockets UDP. Los pasos del proceso son los siguientes:

- **Programa cliente: mediante un bucle genera y envía 10.000 mensaje con el contenido “Mensaje: numero_mensaje” tomando numero_mensaje los valores entre 0 y 9999.**
- **Programa cliente: cuando ha enviado todos los números manda la cadena “FIN”.**
- **Programa servidor: recibe los mensajes y los almacena en un fichero.**
- **Programa servidor: cuando recibe la cadena “FIN” termina la ejecución.**

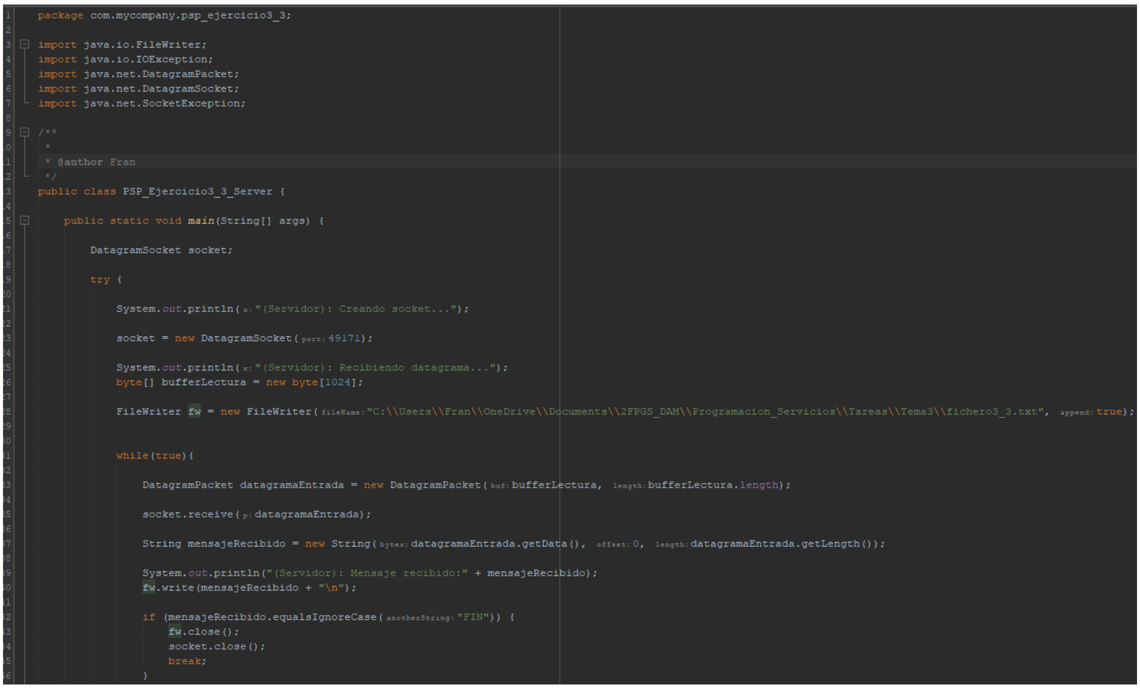
Una vez finalizada la ejecución, comprobar si han llegado todos los datagramas en el mismo orden que se enviaron. Si la ejecución se realiza en un mismo ordenador es probable que lleguen todos los mensajes en el orden correcto. Si la ejecución se realiza en una red de varios ordenadores quizás se produzca alguna pérdida o desorden.

Se debe entregar:

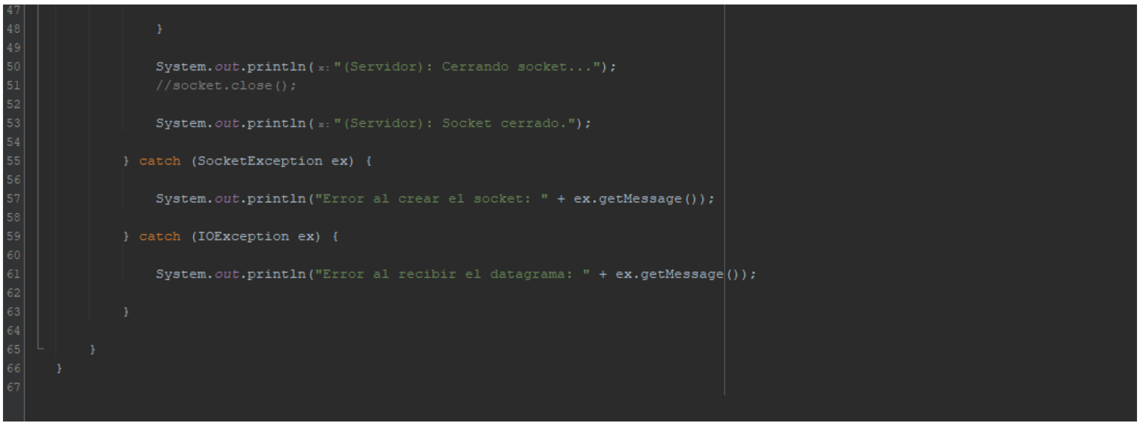
- **Capturas de pantalla tanto del código del servidor y del cliente.**
- **Capturas de las salidas generadas.**

Nota.- Los sockets UDP no están orientados a conexión, por lo que el servidor no se detiene porque el cliente cierre la conexión, ya que esta no existe. La parada del servidor debe ser realizada como consecuencia de una condición cumplida.

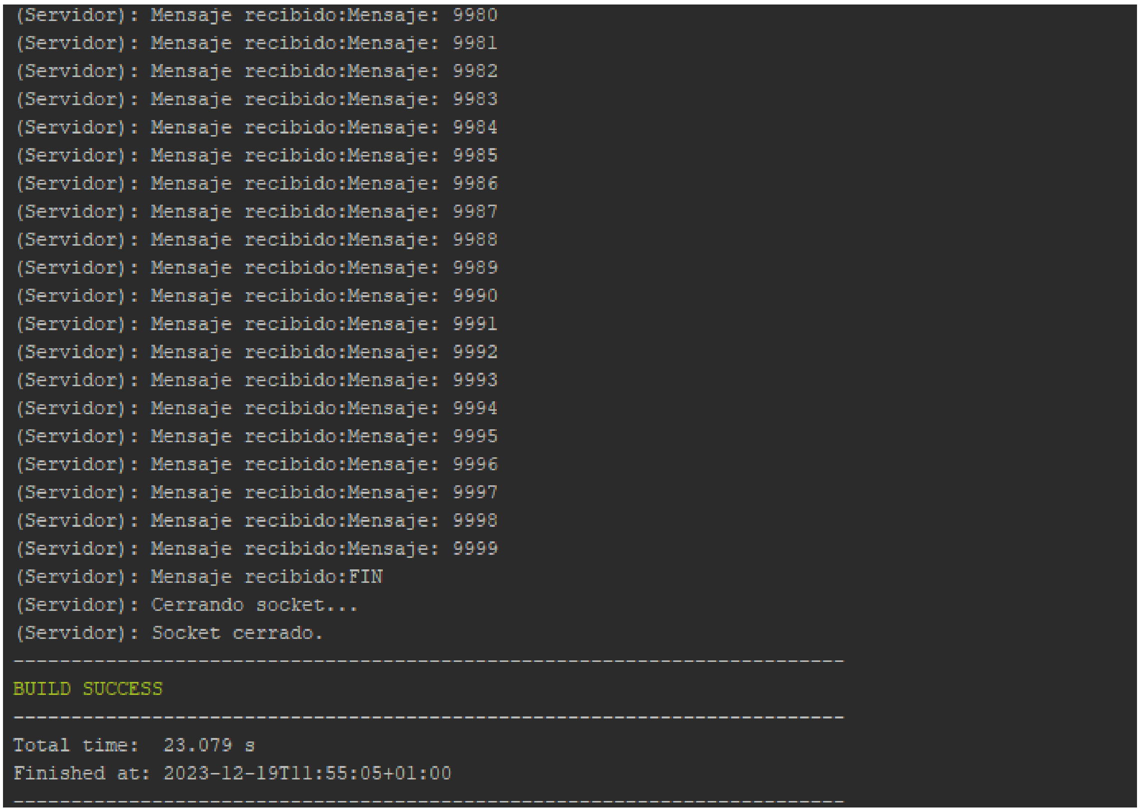
Código del server:



```java
package com.mycompany.psp_ejercicio3_3;

import java.io.FileWriter;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketException;

/**
 *
 * @author Fran
 */
public class PSP_Ejercicio3_3_Server {

    public static void main(String[] args) {

        DatagramSocket socket;

        try {

            System.out.println( x: "(Servidor): Creando socket...");

            socket = new DatagramSocket( port: 49171);

            System.out.println( x: "(Servidor): Recibiendo datagrama...");
            byte[] bufferLectura = new byte[1024];

            FileWriter fw = new FileWriter( fileName: "C:\\Users\\Fran\\OneDrive\\Documents\\2FPGS_DAM\\Programacion_Servicios\\Tareas\\Tema3\\fichero3_3.txt",  append: true);


            while(true){

                DatagramPacket datagramaEntrada = new DatagramPacket( buf: bufferLectura,  length: bufferLectura.length);

                socket.receive( p: datagramaEntrada);

                String mensajeRecibido = new String( bytes: datagramaEntrada.getData(),  offset: 0,  length: datagramaEntrada.getLength());

                System.out.println("(Servidor): Mensaje recibido:" + mensajeRecibido);
                fw.write(mensajeRecibido + "\n");

                if (mensajeRecibido.equalsIgnoreCase( anotherString: "FIN")) {
                    fw.close();
                    socket.close();
                    break;
                }
```

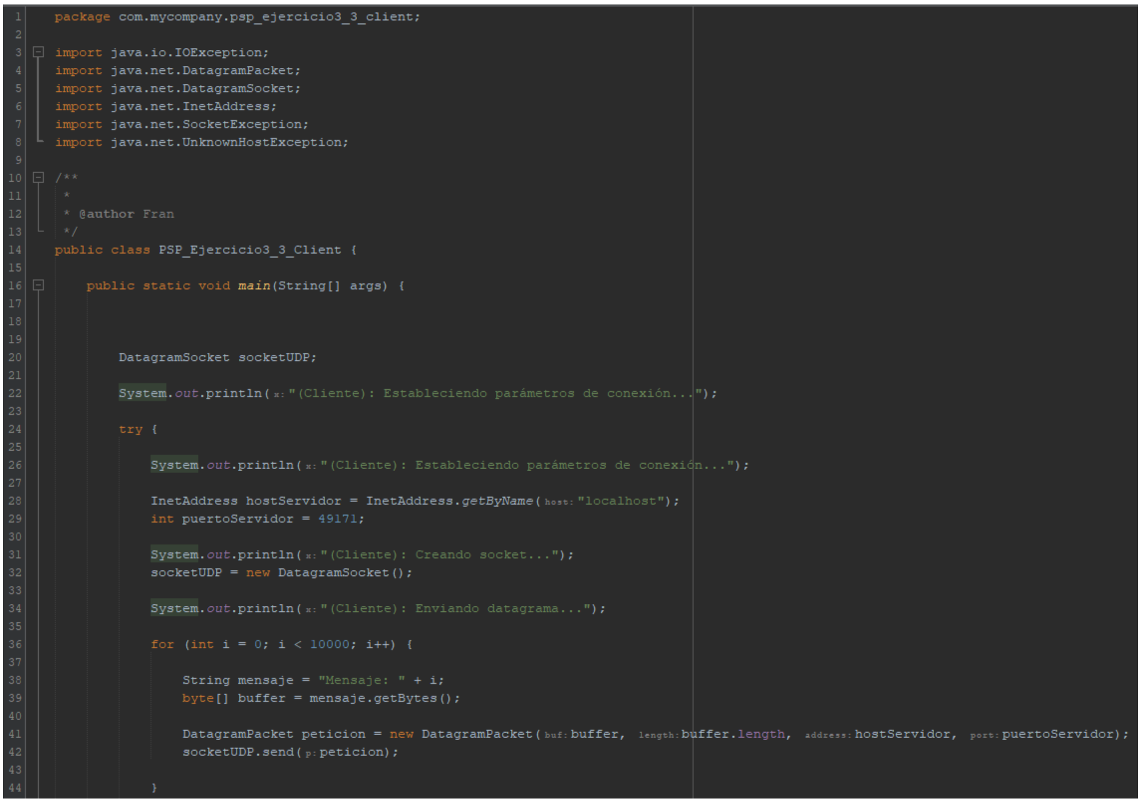


```java

                }

                System.out.println( x: "(Servidor): Cerrando socket...");
                //socket.close();

                System.out.println( x: "(Servidor): Socket cerrado.");

            } catch (SocketException ex) {

                System.out.println("Error al crear el socket: " + ex.getMessage());

            } catch (IOException ex) {

                System.out.println("Error al recibir el datagrama: " + ex.getMessage());

            }

        }
    }

```

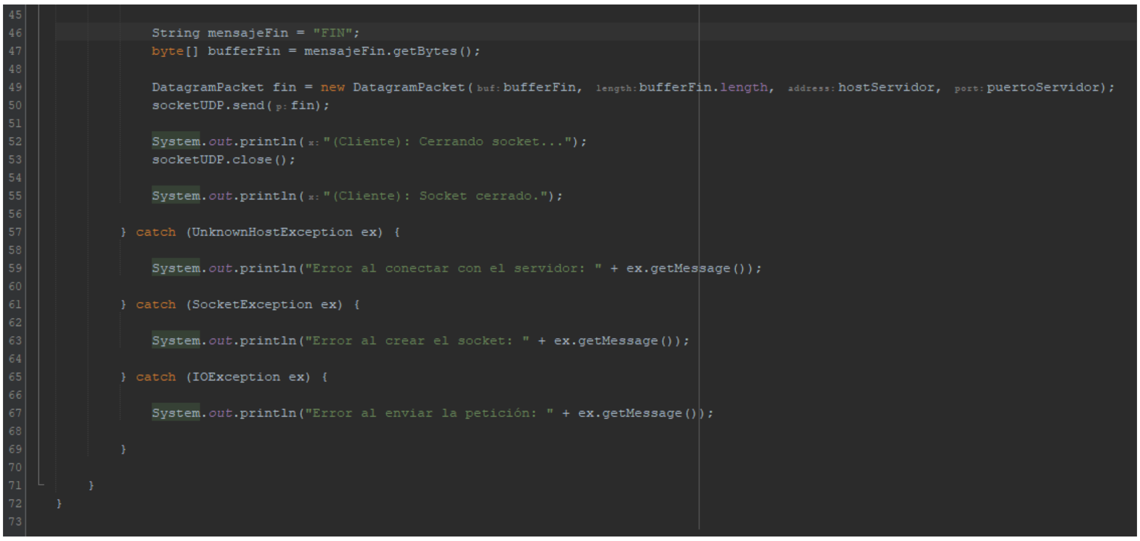


```
(Servidor): Mensaje recibido:Mensaje: 9980
(Servidor): Mensaje recibido:Mensaje: 9981
(Servidor): Mensaje recibido:Mensaje: 9982
(Servidor): Mensaje recibido:Mensaje: 9983
(Servidor): Mensaje recibido:Mensaje: 9984
(Servidor): Mensaje recibido:Mensaje: 9985
(Servidor): Mensaje recibido:Mensaje: 9986
(Servidor): Mensaje recibido:Mensaje: 9987
(Servidor): Mensaje recibido:Mensaje: 9988
(Servidor): Mensaje recibido:Mensaje: 9989
(Servidor): Mensaje recibido:Mensaje: 9990
(Servidor): Mensaje recibido:Mensaje: 9991
(Servidor): Mensaje recibido:Mensaje: 9992
(Servidor): Mensaje recibido:Mensaje: 9993
(Servidor): Mensaje recibido:Mensaje: 9994
(Servidor): Mensaje recibido:Mensaje: 9995
(Servidor): Mensaje recibido:Mensaje: 9996
(Servidor): Mensaje recibido:Mensaje: 9997
(Servidor): Mensaje recibido:Mensaje: 9998
(Servidor): Mensaje recibido:Mensaje: 9999
(Servidor): Mensaje recibido:FIN
(Servidor): Cerrando socket...
(Servidor): Socket cerrado.
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  23.079 s
Finished at: 2023-12-19T11:55:05+01:00
------------------------------------------------------------------------
```

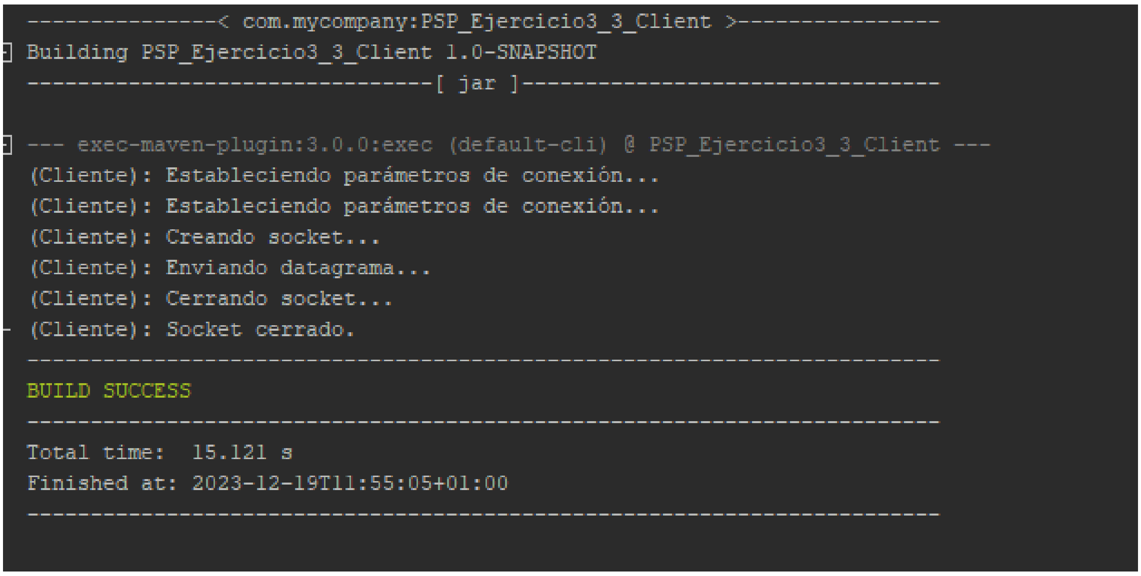
El fichero resultado se encuentra adjunto en el directorio junto al PDF de la tarea



```java
package com.mycompany.psp_ejercicio3_3_client;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.net.UnknownHostException;

/**
 *
 * @author Fran
 */
public class PSP_Ejercicio3_3_Client {

    public static void main(String[] args) {



        DatagramSocket socketUDP;

        System.out.println( x: "(Cliente): Estableciendo parámetros de conexión...");

        try {

            System.out.println( x: "(Cliente): Estableciendo parámetros de conexión...");

            InetAddress hostServidor = InetAddress.getByName( host: "localhost");
            int puertoServidor = 49171;

            System.out.println( x: "(Cliente): Creando socket...");
            socketUDP = new DatagramSocket();

            System.out.println( x: "(Cliente): Enviando datagrama...");

            for (int i = 0; i < 10000; i++) {

                String mensaje = "Mensaje: " + i;
                byte[] buffer = mensaje.getBytes();

                DatagramPacket peticion = new DatagramPacket( buf: buffer,  length: buffer.length,  address: hostServidor,  port: puertoServidor);
                socketUDP.send( p: peticion);

            }
```



```java

            String mensajeFin = "FIN";
            byte[] bufferFin = mensajeFin.getBytes();

            DatagramPacket fin = new DatagramPacket( buf: bufferFin,  length: bufferFin.length,  address: hostServidor,  port: puertoServidor);
            socketUDP.send( p: fin);

            System.out.println( x: "(Cliente): Cerrando socket...");
            socketUDP.close();

            System.out.println( x: "(Cliente): Socket cerrado.");

        } catch (UnknownHostException ex) {

            System.out.println("Error al conectar con el servidor: " + ex.getMessage());

        } catch (SocketException ex) {

            System.out.println("Error al crear el socket: " + ex.getMessage());

        } catch (IOException ex) {

            System.out.println("Error al enviar la petición: " + ex.getMessage());

        }

    }
}

```



```
---------------< com.mycompany:PSP_Ejercicio3_3_Client >----------------
Building PSP_Ejercicio3_3_Client 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ PSP_Ejercicio3_3_Client ---
(Cliente): Estableciendo parámetros de conexión...
(Cliente): Estableciendo parámetros de conexión...
(Cliente): Creando socket...
(Cliente): Enviando datagrama...
(Cliente): Cerrando socket...
(Cliente): Socket cerrado.
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  15.121 s
Finished at: 2023-12-19T11:55:05+01:00
------------------------------------------------------------------------
```