EJERCICIO : Pasar un fichero a través de socket

NOTA: A la hora de evaluar el ejercicio se tendrá en cuenta lo siguiente:

- Que el código compile.
- El tratamiento de excepciones
- Los comentarios indicando que se está haciendo en cada momento.
- El control de los socket, flujos y buffers

En este ejercicio se abrirá un servidor y un cliente . El cliente le pedirá un fichero por su nombre y este se lo envía . Cuando el cliente lo recibe, lo guarda en la misma ruta con la extención copia.

Ejemplo: Si el fichero se llama hola.txt el cliente lo guarda como hola.txt_copia.

El cliente le pide un fichero al servidor por medio de una clase “MensajeDameFichero” que contendrá el nombre del fichero que quiere.

El servidor le contestará con uno o más mensajes de la clase “MensajeTomaFichero”. Este mensaje contendrá un array de bytes con el contenido del fichero.

NOTA: Se puede hacer con un solo mensaje en lugar de varios mensajes . Se lee todo del fichero , se mete en el array de bytes del mensaje y se envía. Esto es válido para ficheros pequeños , pero no resulta muy adecuado cuando el fichero es muy grande. En el ejercicio vamos a configurar el tamaño del array de bytes en 1024.

Necesitaríamos dos mensajes:

1. Un mensaje del cliente al servidor “MensajeDameFichero”
2. Un mensaje del servidor al cliente “MensajeTomaFichero”

SOLUCIÓN Clase MensajeDameFichero

```
import java.io.Serializable;


	public class MensajeDameFichero implements Serializable
	{
	    /** path completo del fichero que se pide */
	    public String nombreFichero;
	}
```

SOLUCIÓN Clase MensajeTomaFichero

```
import java.io.Serializable;

	public class MensajeTomaFichero implements Serializable
	{
	    /** Nombre del fichero que se transmite. Por defecto "" */
	    public String nombreFichero="";

	    /** Si este es el último mensaje del fichero en cuestión o hay más después */
	    public boolean ultimoMensaje=true;

	    /** Cuantos bytes son válidos en el array de bytes */
	    public int bytesValidos=0;

	    /** Array con bytes leidos del fichero */
	    public byte[] contenidoFichero = new byte[LONGITUD_MAXIMA];
```

```
        /** Número máximo de bytes que se enviaán en cada mensaje */
        public final static int LONGITUD_MAXIMA=1024;
    }
```

SOLUCIÓN A LA CLASE SERVIDORFICHERO

```
import java.io.FileInputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.ServerSocket;
import java.net.Socket;

/** Clases servidora que envía un fichero al primer cliente que se lo pida.*/

public class ServidorFichero
{
    /** Instancia la clase servidora y la pone a la escucha del puerto 6000  */

       public static void main(String[] args)
    {
        ServidorFichero sf = new ServidorFichero();
        sf.escucha(6000);
       // System.out.println("** servidor inicializado esperando a un cliente **");
    }

    /*Se escucha el puerto indicado en espera de clientes a los que enviar el fichero.
       puerto (El puerto de escucha)
     */
    public void escucha(int puerto)
    {
        try
        {
            // Se abre el socket servidor
            ServerSocket socketServidor = new ServerSocket(puerto);
            System.out.println("** servidor inicializado esperando a un cliente **");

            // Se espera un cliente
            Socket cliente = socketServidor.accept();

            // Llega un cliente.
            System.out.println("Aceptado cliente");

            // Cuando se cierre el socket, esta opción hara que el cierre se
            // retarde automáticamente hasta 10 segundos dando tiempo al cliente
            // a leer los datos.
            cliente.setSoLinger(true, 10);

            // Se lee el mensaje de petición de fichero del cliente.
            ObjectInputStream flujoEntrada = new
ObjectInputStream(cliente.getInputStream());
            Object mensaje = flujoEntrada.readObject();

            // Si el mensaje es de petición de fichero (El operador instanceof nos
permite comprobar si un objeto es de una clase concreta)
            if (mensaje instanceof MensajeDameFichero)
            {
                // Se muestra en pantalla el fichero pedido y se envia
             System.out.println("El servidor muestra el fichero solicitado por el
cliente y lo envia al cliente");
```

```
                System.out.println("Me piden: "+ ((MensajeDameFichero)
mensaje).nombreFichero);
                enviaFichero(((MensajeDameFichero) mensaje).nombreFichero,new
ObjectOutputStream(cliente.getOutputStream()));
            }
            else
            {
                // Si no es el mensaje esperado, se avisa y se sale todo.
                System.err.println ("Mensaje no esperado
"+mensaje.getClass().getName());
            }

            // Cierre de sockets
            cliente.close();
            socketServidor.close();
        } catch (Exception e)
        {
            e.printStackTrace();
        }
    }

    /**
     * Envía el fichero indicado a través del ObjectOutputStream indicado.
     * @param fichero Nombre de fichero
     * @param oos ObjectOutputStream por el que enviar el fichero
     */
    private void enviaFichero(String fichero, ObjectOutputStream oos)
    {
        try
        {
            boolean enviadoUltimo=false;
            // Se abre el fichero.
            FileInputStream fis = new FileInputStream(fichero);

            // Se instancia y rellena un mensaje de envio de fichero
            MensajeTomaFichero mensaje = new MensajeTomaFichero();
            mensaje.nombreFichero = fichero;

            // Se leen los primeros bytes del fichero en un campo del mensaje
            int leidos = fis.read(mensaje.contenidoFichero);

            // Bucle mientras se vayan leyendo datos del fichero
            while (leidos > -1)
            {

                // Se rellena el número de bytes leidos
                mensaje.bytesValidos = leidos;

                // Si no se han leido el máximo de bytes, es porque el fichero
                // se ha acabado y este es el último mensaje
                if (leidos < MensajeTomaFichero.LONGITUD_MAXIMA)
                {
                    mensaje.ultimoMensaje = true;
                    enviadoUltimo=true;
                }
                else
                    mensaje.ultimoMensaje = false;

                // Se envía por el socket
                oos.writeObject(mensaje);

                // Si es el último mensaje, salimos del bucle.
```

```
                if (mensaje.ultimoMensaje)
                    break;

                // Se crea un nuevo mensaje
                mensaje = new MensajeTomaFichero();
                mensaje.nombreFichero = fichero;

                // y se leen sus bytes.
                leidos = fis.read(mensaje.contenidoFichero);
            }

            if (enviadoUltimo==false)
            {
                mensaje.ultimoMensaje=true;
                mensaje.bytesValidos=0;
                oos.writeObject(mensaje);
            }
            // Se cierra el ObjectOutputStream
            oos.close();
        } catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

SOLUCIÓN A LA CLASE CLIENTEFICHERO

```
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;

/**
 * Pide un fichero al ServidorFichero, lo escribe en pantalla cuando lo recibe y
 * lo guarda en disco añadiendo "_copia" al final del nombre del fichero.

 */
public class ClienteFichero
{

    /**
     * Main del Cliente
     *
     * @param args
     *            de la línea de comandos. Se ignora.
     */
    public static void main(String[] args)
    {
        // Se crea el objeto Cliente "objClienteFichero" y se le manda pedir el
fichero.
        ClienteFichero objClienteFichero = new ClienteFichero();
        objClienteFichero.pide("N:/hola.txt", "localhost", 6000);
    }

    /** Establece comunicación con el servidor en el puerto indicado. Pide el fichero.
Cuando llega, lo escribe en pantalla y en disco duro.
     *
     * Parámetros:  path ( path completo del fichero que se quiere)
                          servidor ( host donde está el servidor)
```

```
                        puerto (Puerto de conexión)
     */
    public void pide(String path, String servidor, int puerto)
    {
       int numeroBytes=0;
        try
        {
            // Se abre el socket.
            Socket socketCliente = new Socket(servidor, puerto);

            // Se envía un mensaje de petición de fichero.
            ObjectOutputStream flujoSalida = new
ObjectOutputStream(socketCliente.getOutputStream());
            MensajeDameFichero mensaje = new MensajeDameFichero();
            mensaje.nombreFichero = path;
            System.out.println("Fichero solicitado por el cliente
"+mensaje.nombreFichero);

System.out.println("*****************************************************");
            flujoSalida.writeObject(mensaje);

            // Se abre un fichero para empezar a copiar lo que se reciba.
            FileOutputStream ficheroCopia = new FileOutputStream(mensaje.nombreFichero
+ "_copia");

            // Se crea un ObjectInputStream del socket para leer los mensajes que
contienen el fichero.
            ObjectInputStream flujoEntrada = new
ObjectInputStream(socketCliente.getInputStream());
            MensajeTomaFichero mensajeRecibido;
            Object mensajeAux;
            do
            {
                // Se lee el mensaje en una variabla auxiliar
                mensajeAux = flujoEntrada.readObject();

                // Si es del tipo esperado, se trata: El operador instanceof nos
permite comprobar si un objeto es de una clase concreta
                if (mensajeAux instanceof MensajeTomaFichero)
                {

                    mensajeRecibido = (MensajeTomaFichero)mensajeAux;
                    // Se escribe en pantalla y en el fichero
                    System.out.print(new String(mensajeRecibido.contenidoFichero, 0,
mensajeRecibido.bytesValidos));
                    ficheroCopia.write(mensajeRecibido.contenidoFichero, 0,
mensajeRecibido.bytesValidos);
                    numeroBytes=numeroBytes+mensajeRecibido.bytesValidos;
                } else
                {
                    // Si no es del tipo esperado, se marca error y se termina
                    // el bucle
                    System.err.println("Mensaje no esperado " +
mensajeAux.getClass().getName());
                    break;
                }
            } while (!mensajeRecibido.ultimoMensaje);
            System.out.println();
            System.out.println();
            System.out.println("Fichero copiado en "+mensaje.nombreFichero + "_copia"+"
Se han copiado un total de "+ numeroBytes + " Bytes");
            // Se cierra socket y fichero
```

```
            ficheroCopia.close();
            flujoEntrada.close();
            socketCliente.close();

        } catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```