EJERCICIO 2: 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”

Clase MensajeDameFichero

```
import java.io.Serializable;

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

Clase MensajeTomaFichero

Tendrá los siguientes atributos:

- **byte[] contenidoFichero** (este atributo contendrá el contenido del fichero, por simplificar , lo configuramos a 1024).
- **int bytesValidos** (Como el array tiene un tamaño fijo, hay que indicar cuantos bytes tiene el fichero).
- boolean ultimoMensaje (indica si es el último mensaje).
- String nombreFichero (nombre del fichero enviado)
- int LONGITUD_MAXIMA (Número máximo de bytes que se envia en cada mensaje, lo inicializamos a 1024).

```
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;
}
```

La clase ServidorFichero deberá realizar lo siguiente:

- Abrir el **socket** y establecer comunicación

  Antes de cerrar el socket haremos una llamada a "setSoLinger(true,10); que espera un máximo de 10 segundos a que el cliente termine de leer los datos. Si enviamos un dato y cerramos inmediatamente el socket, es posible que el cliente no tenga tiempo de leerlo y ese dato se pierda

- Esperar el mensaje **MensajeDameFichero** del cliente

  *Para leer el mensaje, creamos un **ObjectInputStream** con el **socket** y leemos de él*

- Comprobamos que es el mensaje de petición de fichero. Si es así, abrimos el fichero y empezamos a enviarlo.

  El operiador "instanceof" nos permite comprobar si un objeto es de una clase concreta.

  Esto lo hacemos con otro método de la clase Servidor al que le pasamos el nombre del fichero y el ObjectOutputStream que creamos para que envíe el fichero.

- Dentro del método, abrimos el fichero y en un bucle vamos leyendo del fichero y enviando distintos mensajes **MensajeTomaFichero**.

```
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

            System.out.println("** servidor inicializado esperando a un cliente **");

            // Se espera un cliente


            // Llega un cliente.


            // Cuando se cierre el socket, esta opción hará 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.


/* Si el mensaje es de petición de fichero (El operador instanceof nos permite
comprobar si un objeto es de una clase concreta)*/

            if ()
            {
                // Se muestra en pantalla el fichero pedido y se envia

            }
            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

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

/*Envía el fichero indicado a través del ObjectOutputStream indicado:
fichero Nombre de fichero
oos ObjectOutputStream por el que enviar el fichero*/
    private void enviaFichero(String fichero, ObjectOutputStream oos)
    {
        try
        {
```

```
            // Se abre el fichero.


            // Se instancia y rellena un mensaje de envio de fichero


            // Se leen los primeros bytes del fichero en un campo del mensaje


            // Bucle mientras se vayan leyendo datos del fichero
            while ()
            {

                // Se rellena el número de bytes leidos


                // Si no se han leido el máximo de bytes, es porque el fichero
                // se ha acabado y este es el último mensaje
                if ()
                {

                }
                else


                // Se envía por el socket


                // Si es el último mensaje, salimos del bucle.
                if (mensaje.ultimoMensaje)
                    break;

                // Se crea un nuevo mensaje


                // y se leen sus bytes.

            }

            if (enviadoUltimo==false)
            {

            }
            // Se cierra el ObjectOutputStream
            oos.close();
        } catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

La clase ServidorFichero deberá realizar lo siguiente:

En la parte del cliente, simplemente se abre el socket, se pide el mensaje y se escribe en un fichero todo lo que venga.

- Se abre el **socket**
- Se crea un **ObjectOutputStream** y se envía un **MensajeDameFichero**

- **Abrimos el fichero en el que vamos a ir guardando lo que nos llegue y creamos un ObjectInputStream del socket para leer los mensajes**
- **En un bucle hasta recibir el mensaje marcado como último, leemos mensajes y los escribimos en fichero**

```
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
{

    public static void main(String[] args)
    {
   // Se crea el objeto Cliente "objClienteFichero" y se le manda pedir el fichero.

    }

    /** 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.


           // Se envía un mensaje de petición de fichero.

           System.out.println("Fichero solicitado por el cliente
"+mensaje.nombreFichero);

System.out.println("*****************************************************");

           // Se abre un fichero para empezar a copiar lo que se reciba.


           // Se crea un ObjectInputStream del socket para leer los mensajes que
contienen el fichero.

           do
           {
               // Se lee el mensaje en una variabla auxiliar
```

```
// 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)
                {


                    // Se escribe en pantalla y en el fichero

                } else
                {
  // Si no es del tipo esperado, se marca error y se termina el bucle

                }
            } while ();

            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

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