10.C. Ficheros.

1. Trabajando con ficheros.

1.3. Ficheros binarios y ficheros de texto (II).

Los ficheros binarios almacenan la información en bytes, codificada en binario, pudiendo ser de cualquier tipo: fotografías, números, letras, archivos ejecutables, etc.

Los archivos binarios guardan una representación de los datos en el fichero. O sea que, cuando se guarda texto no se guarda el texto en sí, sino que se guarda su representación en código UTF-8.

Para lleer datos de un fichero binario, Java proporciona la clase FillelinputStream. Dicha clase trabaja con bytes que se leen desde el flujo asociado a un fichero. Aquí puedes ver un ejemplo comentado

<mark>asociado a un fichero.</mark> Aquí puedes ver un ejemplo comentado.
package fileinputconbuffer;
import inva in Buffored Input Chronnel
<pre>import java.io.BufferedInputStream;</pre>
import java.io.File;
import java.io.FileInputStream;
<pre>import java.io.IOException;</pre>
nublic class learCanBuffer A
public class leerConBuffer {
<pre>public static void main(String[] args) {</pre>
int tama ;
try{
// Creamos un nuevo objeto File, que es la ruta hasta el fichero desde
<pre>File f = new File("C:\\apuntes\\test.bin");</pre>
// Construimos un flujo de tipo FileInputStream (un flujo de entrada desde
// fichero) sobre el objeto File. Estamos conectando nuestra aplicación
// a un extremo del flujo, por donde van a salir los datos, y "pidiendo"
// al Sistema Operativo que conecte el otro extremo al fichero que indica
// la ruta establecida por el objeto File f que habíamos creado antes. De
<pre>FileInputStream flujoEntrada = new FileInputStream(f);</pre>
<pre>BufferedInputStream fEntradaConBuffer = new BufferedInputStream(flujoEntrada);</pre>
// Escribimos el tamaño del fichero en bytes.
<pre>tama = fEntradaConBuffer.available();</pre>
<pre>System.out.println("Bytes disponibles: " + tama);</pre>
// Indicamos que vamos a intentar leer 50 bytes del fichero.
<pre>System.out.println("Leyendo 50 bytes");</pre>

// que leamos del flujo (realmente del fichero)*/ Byte bytearray(! = new byte[50]) // El método read() de la clase FileInputStream recibe como parámetro un // array de byte, y lo llena leyendo bytes desde el flujo. // Devuelve un número entero, que es el número de bytes que realmente se // han leído desde el flujo. Si el fichero tiene menos de 50 bytes, no // podrá leer los 50 bytes, y escribirá un mensaje indicándolo. ## (IfintradaconBuffer.read(bytearray) != 50) System.out.println("No se pudieron leer 50 bytes"); // Usamos un constructor adecuado de la clase String, que crea un nuevo // String a partir de los bytes leídos desde el flujo, que se almacenaron // en el array bytearray, y escribimos ese String. System.out.println(new String(bytearray, 0, 50)); // Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leído por la aplicación. */ ## IntradaConBuffer.close(!) // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e) System.err.println("No se encuentra el fichero");		
// que leamos del flujo (realmente del fichero)*/ Byte bytearray(! = new byte[50]) // El método read() de la clase FileInputStream recibe como parámetro un // array de byte, y lo llena leyendo bytes desde el flujo. // Devuelve un número entero, que es el número de bytes que realmente se // han leído desde el flujo. Si el fichero tiene menos de 50 bytes, no // podrá leer los 50 bytes, y escribirá un mensaje indicándolo. ## (IfintradaconBuffer.read(bytearray) != 50) System.out.println("No se pudieron leer 50 bytes"); // Usamos un constructor adecuado de la clase String, que crea un nuevo // String a partir de los bytes leídos desde el flujo, que se almacenaron // en el array bytearray, y escribimos ese String. System.out.println(new String(bytearray, 0, 50)); // Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leído por la aplicación. */ ## IntradaConBuffer.close(!) // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e) System.err.println("No se encuentra el fichero");	// Creamos un array de 50 bytes para llenarlo con los 50 bytes	
<pre>// El método read() de la clase FileInputStream recibe como parámetro un // array de byte, y lo llena leyendo bytes desde el flujo. // bevuelve un número entero, que es el número de bytes que realmente se // han leido desde el flujo. Si el fichero tiene menos de 50 bytes, no // podrá leer los 50 bytes, y escribirá un mensaje indicándolo. df IfEntradaConBuffer.read(bytearray) != 50) System.out.println("No se pudieron leer 50 bytes")} // Usamos un constructor adecuado de la clase String, que crea un nuevo // String a partir de los bytes leidos desde el flujo, que se almacenaron // en el array bytearray, y escribinos ese String. System.out.println(new String(bytearray, 0, 501)); // Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leido por la aplicación. */ fEntradaConBuffer.close()) // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException el) System.err.println("No se encuentra el fichero"))</pre>		
// El método read() de la clase FileInputStream recibe como parámetro un // array de byte, y lo llena leyendo bytes desde el flujo. // bevuelve un número entero, que es el número de bytes que realmente se // han leido desde el flujo. Si el fichero tiene menos de 50 bytes, no // podrá leer los 50 bytes, y escribirá un mensaje indicándolo. df (fEntradaConBuffer.read(bytearray) != 50) System.out.println("No se pudieron leer 50 bytes"); // Usamos un constructor adecuado de la clase String, que crea un nuevo // String a partir de los bytes leidos desde el flujo, que se almacenaron // en el array bytearray, y escribimos ese String. System.out.println(new String(bytearray, 0, 50)); // Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leido por la aplicación. */ fEntradaConBuffer.close(); // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException el) System.err.println("No se encuentra el fichero");	// que leamos del flujo (realmente del fichero)*/	
// El método read() de la clase FileInputStream recibe como parámetro un // array de byte, y lo llena leyendo bytes desde el flujo. // bevuelve un número entero, que es el número de bytes que realmente se // han leido desde el flujo. Si el fichero tiene menos de 50 bytes, no // podrá leer los 50 bytes, y escribirá un mensaje indicándolo. df (fEntradaConBuffer.read(bytearray) != 50) System.out.println("No se pudieron leer 50 bytes"); // Usamos un constructor adecuado de la clase String, que crea un nuevo // String a partir de los bytes leidos desde el flujo, que se almacenaron // en el array bytearray, y escribimos ese String. System.out.println(new String(bytearray, 0, 50)); // Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leido por la aplicación. */ fEntradaConBuffer.close(); // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException el) System.err.println("No se encuentra el fichero");	huta hutaansay [] _ ray huta[50] a	
// array de byte, y lo llena leyendo bytes desde el flujo. // Devuelve un número entero, que es el número de bytes que realmente se // han leído desde el flujo. Si el fichero tiene menos de 50 bytes, no // podrá leer los 50 bytes, y escribirá un mensaje indicándolo. df (fintradaConBuffer.read(bytearray) != 50) System.out.println("No se pudieron leer 50 bytes"); // Usamos un constructor adecuado de la clase String, que crea un nuevo // String a partir de los bytes leidos desde el flujo, que se almacenaron // en el array bytearray, y escribimos ese String. System.out.println(new String(bytearray, 0, 50)); // Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leido por la aplicación. */ tentradaConBuffer.close(); // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e)á System.err.println("No se encuentra el fichero");	byte bytearray[] = new byte[50];	
// array de byte, y lo llena leyendo bytes desde el flujo. // Devuelve un número entero, que es el número de bytes que realmente se // han leído desde el flujo. Si el fichero tiene menos de 50 bytes, no // podrá leer los 50 bytes, y escribirá un mensaje indicándolo. df (fintradaConBuffer.read(bytearray) != 50) System.out.println("No se pudieron leer 50 bytes"); // Usamos un constructor adecuado de la clase String, que crea un nuevo // String a partir de los bytes leidos desde el flujo, que se almacenaron // en el array bytearray, y escribimos ese String. System.out.println(new String(bytearray, 0, 50)); // Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leido por la aplicación. */ tentradaConBuffer.close(); // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e)á System.err.println("No se encuentra el fichero");		
// array de byte, y lo llena leyendo bytes desde el flujo. // Devuelve un número entero, que es el número de bytes que realmente se // han leído desde el flujo. Si el fichero tiene menos de 50 bytes, no // podrá leer los 50 bytes, y escribirá un mensaje indicándolo. df (fintradaConBuffer.read(bytearray) != 50) System.out.println("No se pudieron leer 50 bytes"); // Usamos un constructor adecuado de la clase String, que crea un nuevo // String a partir de los bytes leidos desde el flujo, que se almacenaron // en el array bytearray, y escribimos ese String. System.out.println(new String(bytearray, 0, 50)); // Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leido por la aplicación. */ tentradaConBuffer.close(); // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e)á System.err.println("No se encuentra el fichero");		
// Devuelve un número entero, que es el número de bytes que realmente se // han leído desde el flujo. Si el fichero tiene menos de 50 bytes, no // podrá leer los 50 bytes, y escribirá un mensaje indicándolo. (f (fentradaConBuffer.read(bytearray) != 50) System.out.println("No se pudieron leer 50 bytes"); // Usamos un constructor adecuado de la clase String, que crea un nuevo // String a partir de los bytes leídos desde el flujo, que se almacenaron // en el array bytearray, y escribimos ese String. System.out.println(new String(bytearray, 0, 50)); // Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leído por la aplicación. */ (EntradaConBuffer.close()) // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e) System.err.println("No se encuentra el fichero");	// El método read() de la clase FileInputStream recibe como parámetro un	
// Devuelve un número entero, que es el número de bytes que realmente se // han leído desde el flujo. Si el fichero tiene menos de 50 bytes, no // podrá leer los 50 bytes, y escribirá un mensaje indicándolo. (f (fentradaConBuffer.read(bytearray) != 50) System.out.println("No se pudieron leer 50 bytes"); // Usamos un constructor adecuado de la clase String, que crea un nuevo // String a partir de los bytes leídos desde el flujo, que se almacenaron // en el array bytearray, y escribimos ese String. System.out.println(new String(bytearray, 0, 50)); // Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leído por la aplicación. */ (EntradaConBuffer.close()) // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e) System.err.println("No se encuentra el fichero");	// array de byte, y lo llena leyendo bytes desde el flujo.	
// han leido desde el flujo. Si el fichero tiene menos de 50 bytes, no // podrá leer los 50 bytes, y escribirá un mensaje indicándolo. df (fEntradaConBuffer.read(bytearray) != 50) System.out.println("No se pudieron leer 50 bytes"); // Usamos un constructor adecuado de la clase String, que crea un nuevo // String a partir de los bytes leidos desde el flujo, que se almacenaron // en el array bytearray, y escribimos ese String. System.out.println(new String(bytearray, 0, 50)); // Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leido por la aplicación. */ fEntradaConBuffer.close()) // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e)4 System.err.println("No se encuentra el fichero");		
// podrá leer los 50 bytes, y escribirá un mensaje indicándolo. (f (fEntradaConBuffer.read(bytearray) != 50) System.out.println("No se pudieron leer 50 bytes")) // Usamos un constructor adecuado de la clase String, que crea un nuevo // String a partir de los bytes leidos desde el flujo, que se almacenaron // en el array bytearray, y escribimos ese String. System.out.println(new String(bytearray, 0, 50))) // Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leido por la aplicación. */ tentradaConBuffer.close()) // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e) { System.err.println("No se encuentra el fichero");	// Devuelve un número entero, que es el número de bytes que realmente se	
// podrá leer los 50 bytes, y escribirá un mensaje indicándolo. (f (fEntradaConBuffer.read(bytearray) != 50) System.out.println("No se pudieron leer 50 bytes")) // Usamos un constructor adecuado de la clase String, que crea un nuevo // String a partir de los bytes leidos desde el flujo, que se almacenaron // en el array bytearray, y escribimos ese String. System.out.println(new String(bytearray, 0, 50))) // Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leido por la aplicación. */ tentradaConBuffer.close()) // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e) { System.err.println("No se encuentra el fichero");	// ban leído desde el fluio. Si el fichero tiene menos de 50 hytes, no	
<pre>System.out.println("No se pudieron leer 50 bytes"); // Usamos un constructor adecuado de la clase String, que crea un nuevo // String a partir de los bytes leidos desde el flujo, que se almacenaron // en el array bytearray, y escribimos ese String. System.out.println(new String(bytearray, 0, 50)); // Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leido por la aplicación. */ **EntradaConBuffer.close(); // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e)* System.err.println("No se encuentra el fichero");</pre>		
System.out.println("No se pudieron leer 50 bytes"); // Usamos un constructor adecuado de la clase String, que crea un nuevo // String a partir de los bytes leídos desde el flujo, que se almacenaron // en el array bytearray, y escribimos ese String. System.out.println(new String(bytearray, 0, 50))) // Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leído por la aplicación. */ fEntradaConBuffer.close(); // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e). System.err.println("No se encuentra el fichero");	// podrá leer los 50 bytes, y escribirá un mensaje indicándolo.	
System.out.println("No se pudieron leer 50 bytes"); // Usamos un constructor adecuado de la clase String, que crea un nuevo // String a partir de los bytes leídos desde el flujo, que se almacenaron // en el array bytearray, y escribimos ese String. System.out.println(new String(bytearray, 0, 50))) // Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leído por la aplicación. */ fEntradaConBuffer.close(); // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e). System.err.println("No se encuentra el fichero");	if (fEntradaConBuffer read(hytearray) != 50)	
// Usamos un constructor adecuado de la clase String, que crea un nuevo // String a partir de los bytes leidos desde el flujo, que se almacenaron // en el array bytearray, y escribimos ese String. System.out.println(new String(bytearray, 0, 50)); // Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leido por la aplicación. */ fEntradaConBuffer.close(); // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (10Exception el) System.err.println("No se encuentra el fichero");	2. (Filteraddoonburrerredd by tearray) .— 30)	
<pre>// String a partir de los bytes leidos desde el flujo, que se almacenaron // en el array bytearray, y escribimos ese String. System.out.println(new String(bytearray, 0, 50)); // Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leido por la aplicación. */ (EntradaConBuffer.close(); // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e){ System.err.println("No se encuentra el fichero");</pre>	<pre>System.out.println("No se pudieron leer 50 bytes");</pre>	
<pre>// String a partir de los bytes leidos desde el flujo, que se almacenaron // en el array bytearray, y escribimos ese String. System.out.println(new String(bytearray, 0, 50)); // Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leido por la aplicación. */ (EntradaConBuffer.close(); // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e){ System.err.println("No se encuentra el fichero");</pre>		
<pre>// String a partir de los bytes leidos desde el flujo, que se almacenaron // en el array bytearray, y escribimos ese String. System.out.println(new String(bytearray, 0, 50)); // Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leido por la aplicación. */ (EntradaConBuffer.close(); // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e){ System.err.println("No se encuentra el fichero");</pre>		
<pre>// en el array bytearray, y escribimos ese String. System.out.println(new String(bytearray, 0, 50)); // Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leído por la aplicación. */ fEntradaConBuffer.close(); // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e); System.err.println("No se encuentra el fichero");</pre>	// Usamos un constructor adecuado de la clase String, que crea un nuevo	
<pre>// en el array bytearray, y escribimos ese String. System.out.println(new String(bytearray, 0, 50)); // Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leído por la aplicación. */ fEntradaConBuffer.close(); // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e); System.err.println("No se encuentra el fichero");</pre>	// Chaire a sentia de las hutes lafdes desde al fluis que se almessare	
System.out.println(new String(bytearray, 0, 50)); // Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leído por la aplicación. */ fEntradaConBuffer.close(); // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e){ System.err.println("No se encuentra el fichero");	// String a partir de los bytes terdos desde et riujo, que se atmacenaron	
<pre>// Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leído por la aplicación. */ fEntradaConBuffer.close(); // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e){ System.err.println("No se encuentra el fichero");</pre>	// en el array bytearray, y escribimos ese String.	
<pre>// Finalmente cerramos el flujo.Es importante cerrar los flujos // para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leído por la aplicación. */ fEntradaConBuffer.close(); // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e){ System.err.println("No se encuentra el fichero");</pre>	System out println/new String/bytearray 0 50\\.	
<pre>// para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leído por la aplicación. */ fEntradaConBuffer.close(); // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e){ System.err.println("No se encuentra el fichero");</pre>	ejotemiottipi intiitiinin oti inglajitati tajį tį otiij	
<pre>// para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leído por la aplicación. */ fEntradaConBuffer.close(); // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e){ System.err.println("No se encuentra el fichero");</pre>		
<pre>// para liberar ese recurso. Al cerrar el flujo, se comprueba que no // haya quedado ningún dato en el flujo sin que se haya leído por la aplicación. */ fEntradaConBuffer.close(); // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e){ System.err.println("No se encuentra el fichero");</pre>	// Finalmente corrames el fluie Es importante corrar les fluies	
<pre>// haya quedado ningún dato en el flujo sin que se haya leído por la aplicación. */ fEntradaConBuffer.close(); // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e){</pre>	// I Ina chieffe Cerranius et regiones importante cerrar los regios	
<pre>fEntradaConBuffer.close(); // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e){ System.err.println("No se encuentra el fichero");</pre>	// para liberar ese recurso. Al cerrar el flujo, se comprueba que no	
<pre>fEntradaConBuffer.close(); // Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e){ System.err.println("No se encuentra el fichero");</pre>	// hava quodado ningún dato en el flujo sin que se hava leíde per la anlicación */	
<pre>// Capturamos la excepción de Entrada/Salida. El error que puede // producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e){ System.err.println("No se encuentra el fichero");</pre>	// maya quedado mingum dato em et reajo sim que se maya teruo por la apricación. */	
<pre>// producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e){ System.err.println("No se encuentra el fichero"); }</pre>	<pre>fEntradaConBuffer.close();</pre>	
<pre>// producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e){ System.err.println("No se encuentra el fichero"); }</pre>		
<pre>// producirse en este caso es que el fichero no esté accesible, y // es el mensaje que enviamos en tal caso. catch (IOException e){ System.err.println("No se encuentra el fichero"); }</pre>		
<pre>// es el mensaje que enviamos en tal caso. catch (IOException e){ System.err.println("No se encuentra el fichero"); </pre>	// Capturamos la excepción de Entrada/Salida. El error que puede	
<pre>// es el mensaje que enviamos en tal caso. catch (IOException e){ System.err.println("No se encuentra el fichero"); </pre>	// producirco en este care es que el fichero po está casacible ::	
catch (IOException e){ System.err.println("No se encuentra el fichero");	// productive en este caso es que et lichero no este accesiote, y	
System.err.println("No se encuentra el fichero");	// es el mensaje que enviamos en tal caso.	
System.err.println("No se encuentra el fichero");	1tk / TOP	
	reaccii (10Except10ii e/i	
	<pre>System.err.println("No se encuentra el fichero");</pre>	

Para escribiir datos a un fiichero binario, la clase nos permite usar un fichero para escritura de bytes en él, es la clase FileOutputStream. La filosofía es la misma que para la lectura de datos, pero ahora el flujo es en dirección contraria, desde la aplicación que hace de fuente de datos hasta el fichero, que los consume.

Autoevaluación

Señala si es verdadera o falsa la siguiente afirmación:

Para leer datos desde un fichero codificados en binario empleamos la clase FileOutputStream. ¿Verdadero o falso?

O Verdadero.



EducaMadrid - Vicepresidencia, Consejería de Educación y Universidades - Ayuda



