**Lectura y Escritura de Ficheros en Java**

1. **Lectura de un fichero de texto en java** 
   1. **Carácter a carácter**:

Podemos abrir un fichero de texto para leer usando la clase **[FileReader](http://docs.oracle.com/javase/7/docs/api/java/io/FileReader.html)**. Esta clase tiene métodos que nos permiten leer caracteres. Sin embargo, suele ser habitual querer las líneas completas, bien porque nos interesa la línea completa, bien para poder analizarla luego y extraer campos de ella.

La apertura del fichero y su posterior lectura pueden lanzar excepciones que debemos capturar. Por ello, la apertura del fichero, la lectura, y el cierre, debe meterse en un bloque *try-catch*.

Podemos usar el método read() para leer **carácter a carácter**:

**import** java.io.\*;

**public** **class** LecturaFichero {

**public** **static** **void** main(String[] args) **throws** IOException {

//se puede usar de las dos maneras (sólo una línea de las dos)

String nombreFichero = "C:\\carpeta\\prueba.txt";

String nombreFichero = "C:/carpeta/prueba.txt";

//Declarar una variable FileReader

FileReader fr = **null**;

**try** {

//Abrir el fichero indicado en la variable nombreFichero

fr = **new** FileReader(nombreFichero);

//Leer el primer carácter

//Se debe almacenar en una variable de tipo int

**int** caract = fr.read();

//Se recorre el fichero hasta encontrar el carácter -1

// que marca el final del fichero

**while**(caract != -1) {

//Mostrar en pantalla el carácter leído convertido a char

System.***out***.print((**char**)caract);

caract = fr.read(); //Leer el siguiente carácter

}

fr.close(); //cerramos el fichero

}

**catch** (FileNotFoundException e) {

//Operaciones en caso de no encontrar el fichero

System.***out***.println("Error: Fichero no encontrado");

System.***out***.println(e.getMessage());//Mostrar el error

}

}

}

* 1. **Línea a línea con Buffer**:

FileReader no contiene métodos que nos permitan leer líneas completas, pero sí **BufferedReader**. Podemos construir un BufferedReader a partir del FileReader de la siguiente forma:

File archivo = **new** File ("C:\\archivo.txt");

FileReader fr = **new** FileReader (archivo);

BufferedReader br = **new** BufferedReader(fr);

...

String linea = br.readLine();

El siguiente es un código completo con todo lo mencionado:

**import** java.io.\*;

**class** LeeFichero {

**public** **static** **void** main(String [] arg) **throws** IOException {

String archivo = "C:\\archivo.txt";

FileReader fr = **null**;

BufferedReader br = **null**;

**try** {

// Apertura del fichero y creación de BufferedReader para poder

// hacer una lectura línea a línea con el método readLine()

fr = **new** FileReader (archivo);

br = **new** BufferedReader(fr);

String linea;

// Lectura del fichero

linea=br.readLine();

// Recorremos línea a línea el fichero, hasta que acabe el fichero

**while**(linea!=**null**){

System.out.println(linea);

// Leemos la siguiente línea

linea=br.readLine();

}

// Cierre del fichero

fr.close();

} **catch**(Exception e){

e.printStackTrace();

}

}

}

1. **Escritura de un fichero de texto en java**
   1. **Escritura de fichero carácter a carácter:**

El siguiente código **escribe un fichero de texto** desde cero.

**import** java.io.\*;

**public** **class** EscribeFichero

{

**public** **static** **void** main(String[] args) **throws** IOException

{

//Usamos la clase FileWriter para escribir en un fichero

FileWriter fichero = **null**;

String nombreFichero = "D:\\archivo.txt";

**try** {

fichero = **new** FileWriter(nombreFichero);

String linea = "Esto es una prueba de escritura";

//Escribimos tantos caracteres como haya en linea

**for** (**int** i = 0; i < linea.length(); i++){

//Escribimos el carácter leído en linea

fichero.write(linea.charAt(i));

}

//Cerramos el fichero

fichero.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

Si queremos **añadir al final de un fichero** ya existente, simplemente debemos poner un flag a true como segundo parámetro del constructor de **FileWriter**.

fichero = **new** FileWriter(nombreFichero,**true**);

* 1. **Escritura de fichero línea a línea:**

El siguiente código **escribe un fichero de texto** desde cero. Pone en él 10 líneas.

**Igual que en el ejemplo anterior…**

**try** {

fichero = **new** FileWriter(nombreFichero);

//Escribimos tantas líneas como vueltas da el bucle

**for** (**int** i = 0; i < 10; i++){

//En cada línea escribimos “Linea 0” siendo 0 cada número de línea

fichero.write("Linea " + i + "\r\n");

}

//Cerramos el fichero

fichero.close();

} **catch** (IOException e) {

…

* 1. **Escritura de fichero línea a línea con Buffer:**

Transformamos el mismo código anterior, pero usando un Buffer intermedio:

…

**try** {

fichero = **new** FileWriter(nombreFichero);

BufferedWriter bw = **new** BufferedWriter(fichero);

//Escribimos tantas líneas como vueltas da el bucle

**for** (**int** i = 0; i < 10; i++){

//En cada línea escribimos “Linea 0” siendo 0 cada número de línea

bw.write("Linea " + i );

bw.newLine(); //Hace un salto de línea en el fichero

}

bw.flush(); //Hace la carga del buffer al fichero

//Cerramos el fichero

fichero.close();

} **catch** (IOException e) {

…

1. **Ficheros binarios**

Para ficheros binarios se hace exactamente igual, pero en vez de usar los "**FileReader**" y los "**FileWriter**", se usan los "**FileInputStream**" y los "**FileOutputStream**".

El siguiente ejemplo hace una copia binaria de un fichero:

**import** java.io.\*;

**public** **class** CopiadorMejorado {

**public** **static** **void** main(String[] args) **throws** IOException {

// Creamos todos los flujos de entrada y salida, de ficheros y buffer

String archivoEntrada = "C:\\ Pictures\\original.png";

String archivoSalida = "C:\\Pictures\\copia.png";

**try**{

FileInputStream streamEntrada = **new** FileInputStream(archivoEntrada);

FileOutputStream streamSalida = **new** FileOutputStream(archivoSalida);

BufferedInputStream lectorMejorado = **new** BufferedInputStream(streamEntrada);

BufferedOutputStream escritorMejorado = **new** BufferedOutputStream(streamSalida);

**int** byteLeido;

System.***out***.println("\n\tEl archivo está siendo copiado....");

// leemos cada byte del bufferedImput, y lo escribimos en el bufferedOutput

**while** ((byteLeido = lectorMejorado.read()) != -1) {

escritorMejorado.write(byteLeido);

}

// cerramos los ficheros de entrada y salida

lectorMejorado.close();

escritorMejorado.close();

} **catch** (IOException e) {

e.printStackTrace();

}

System.***out***.println("\tEl archivo ha sido copiado con éxito....\n");

}

}

Los Buffered\*

Si usamos sólo **FileInputStream**, **FileOuputStream**, **FileReader** o **FileWriter**, cada vez que hagamos una lectura o escritura, se hará físicamente en el disco duro. Si escribimos o leemos pocos caracteres cada vez, el proceso se hace costoso y lento, con muchos accesos a disco duro.

Los **BufferedReader**, **BufferedInputStream**, **BufferedWriter** y **BufferedOutputStream** añaden un buffer intermedio. Cuando leamos o escribamos, esta clase controlará los accesos a disco.

* Si vamos escribiendo, se guardarán los datos hasta que tenga bastantes datos como para hacer la escritura eficiente.
* Si queremos leer, la clase leerá muchos datos de golpe, aunque sólo nos dé los que hayamos pedido. En las siguientes lecturas nos dará lo que tiene almacenado, hasta que necesite leer otra vez.

Esta forma de trabajar hace los accesos a disco más eficientes y el programa correrá más rápido. La diferencia se notará más cuanto mayor sea el fichero que queremos leer o escribir.