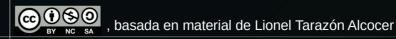
Escritura y lectura de ficheros en Java



package java.io

Librería con las clases necesarias para manejar ficheros.

- Clase File
- Clase FileWriter (escritura de archivos)

Usaremos la clase Scanner de java.util para leer.

Tipos de lectura/escritura

Java permite leer/escribir caracteres de un archivo, pero también permite leer/escribir un byte.

Solo nos vamos a centrar en la lectura de ficheros "orientados a carácter".

Para más información sobre ficheros "orientados a byte", ver las clases FileInputStream, FileOutputStream y otras en la documentación sobre java.io.

Ejemplo archivo de texto

```
Terminal - cmorenoruiz@madrid-B85M-DS3H-A: ~
 Archivo Editar Ver Terminal Pestañas
                                        Avuda
cmorenoruiz@madrid-B85M-DS3H-A:~$ touch hola.txt
cmorenoruiz@madrid-B85M-DS3H-A:~$ cat hola.txt
cmorenoruiz@madrid-B85M-DS3H-A:~$ echo cualquier cosa >> hola.txt
cmorenoruiz@madrid-B85M-DS3H-A:~$ cat hola.txt
cualquier cosa
cmorenoruiz@madrid-B85M-DS3H-A:~$ xxd hola.txt
                                                   cualquier cosa.
00000000: 6375 616c 7175 6965 7220 636f 7361 0a
cmorenoruiz@madrid-B85M-DS3H-A:~$
```

<u>Leyendo con la clase Scanner</u>

```
import java.io.File;
import java.util.Scanner;
File archivo = new File("Unidad11/Documento.txt");
Scanner lectorArchivo = new Scanner(archivo);
```

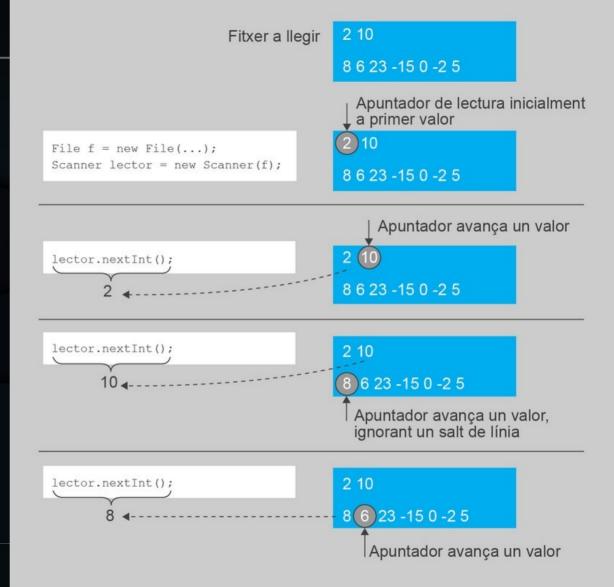
Leyendo con la clase Scanner

El objeto Scanner gestiona internamente un **apuntador**, que indica sobre qué valor actuarán las operaciones de lectura.

Inicialmente el apuntador se encuentra en el primer valor dentro del archivo. Cada vez que se hace una lectura, el apuntador avanza automáticamente hasta el siguiente valor dentro del archivo y **no hay ninguna manera de hacerlo retroceder.**

El apuntador seguirá avanzando hasta que queramos, o hasta llegar al final del fichero.

Leyendo con la clase Scanner



<u>Importante sobre el Scanner</u>

Se lanzará una excepción de tipo java.io.FileNotFoundException si el fichero no existe y pude lanzar otras excepciones.

Recordar la diferencia entre los métodos next() y nextLine() y utilizar hasNext()

Siempre hay que cerrar los archivos con **close()** cuando se ha terminado de leer o escribir en ellos.

Clase FileWriter

import java.io.FileWriter

Representa el acceso a un archivo, que estará disponible, de manera exclusiva, para escribir en él.

public FileWriter(File file)

public FileWriter(File file, boolean append)

Métodos de la clase FileWriter

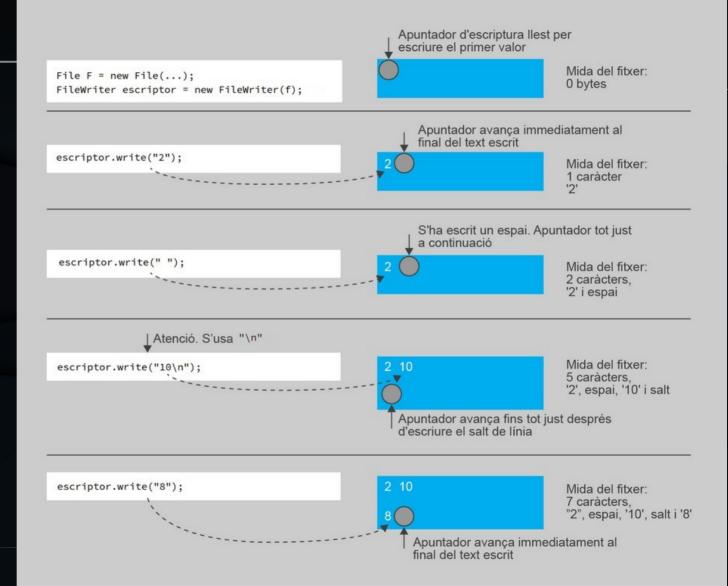
void write(String str)

Escribirá la cadena str en el fichero. Si se desea agregar un final de línea se puede agregar "\n".

void close()

Al terminar la escritura también es imprescindible invocar este método para cerrarlo y asegurar la correcta escritura de datos.

Escribiendo con la clase FileWriter



Importante sobre FileWriter

Tanto el constructor de FileWriter como el método write() pueden lanzar una excepción IOException si se produce algún error inesperado.

Ojo al escribir números: fichero.write("" + valor)