

Escribir texto

```
PrintWriter fOut = new PrintWriter(new BufferedWriter(new FileWriter("nombre_fichero.txt")));
```

```
PrintWriter fOut = new PrintWriter(new FileWriter("nombre_fichero.txt"));
```

```
PrintWriter fOut = new PrintWriter("nombre_fichero.txt");
```

```
fOut.println(dato);
```

```
fOut.println(dato);
```

```
fOut.println();
```

```
fOut.format("cadena_de_formato", dato, dato...);
```

```
fOut.format(Locale.ROOT, "cadena_de_formato", dato, dato...);
```

```
fOut.printf("cadena_de_formato", dato, dato...);
```

Si el fichero existe, se machaca, creándose de nuevo.

Puede abrirse para añadir al final mediante el constructor del

FileWriter: `PrintWriter fOut = new PrintWriter(new BufferedWriter(new FileWriter("nombre_fichero.txt")));`

Leer texto (cont)

`FileReader` devuelve un objeto de tipo `InputStream` que representa a un fichero o una carpeta. Se puede especificar un nombre con trayectoria absoluta o relativa sin buffer.

Una vez leída una línea de la entrada se puede tokenizar o realizar cualquier otra acción.

Operaciones con el sistema de ficheros

```
File f = new File("nombre");
```

```
f.delete() boolean
```

```
f.exists() boolean
```

```
f.getAbsolutePath() String
```

```
f.getParent() String
```

```
f.isDirectory() boolean
```

```
f.mkdir() boolean
```

```
f.renameTo(new File("nuevo_nombre")) boolean
```

```
f.list() String[]
```

Escribir binario secuencial

Apertura preparada del

PrintWriter para tokenizar o realizar cualquier acción.

Stream

handler

Un objeto de tipo `File` representa a un fichero o una carpeta. Se puede especificar un nombre con trayectoria absoluta o relativa sin buffer.

Apertura absoluta o relativa sin buffer.

Borrar el fichero. Similar al anterior. True si éxito.

True si el fichero existe. Imprimir un dato.

La trayectoria absoluta de un dato y

El directorio de cambio de nivel superior

True si el nombre representa a un directorio. False si es un fichero u otra cosa.

Crear un directorio con el nombre especificado. True si éxito.

Imprimir un dato. Cambiar nombre. True si éxito.

Si f representa a un directorio, devuelve los nombres de los elementos que contiene.

format

```
er o", true));
```

Leer texto

```
Scanner fIn = new Scanner(new BufferedReader(new FileReader("nomb re_ fic her o")));
```

```
Scanner fIn = new Scanner(new FileReader("nomb re_ fic her o"));
```

```
fIn.nextLine();
```

```
DataOutputStream fOut = new DataOutputStream(new FileOutputStream("nomb re_ fic her o"));
```

```
fOut.writeUTF(fIn.nextLine());
```

```
fOut.writeUTF(dato);
```

Apertura del stream, con buffer de entrada.

Sin buffer

Lee una línea de la entrada.



By **victorjfg**
cheatography.com/victorjfg/

Published 28th May, 2017.
Last updated 28th May, 2017.
Page 1 of 4.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
<http://crosswordcheats.com>

Escribir binario secuencial (cont)

`fOut.writeUTF(data)` Escribe una cadena en la codificación UTF de Unicode.

Dispone de métodos para grabar todos los tipos primitivos, cada uno con una longitud fija y cadenas de longitud variable. Para las cadenas graba primero su longitud en dos bytes, y a continuación el contenido de la cadena.

Dispone de un método para escribir un número arbitrario de bytes.

Leer binario secuencial

```
DataInputStream fin = new DataInputStream(new FileInputStream("archivo.dat", true));
ObjectInputStream ois = new ObjectInputStream(fin);
```

```
int n = fin.readInt();
```

```
double d = fin.readDouble();
```

```
String s = fin.readUTF();
```

Dispone de métodos para leer todos los tipos primitivos.

Dispone de un método para leer un número arbitrario de bytes.

Escribir / leer binario aleatorio

Escribir / leer binario aleatorio (cont)

`f.read(buff, offset, length)` Lee una cierta cantidad de bytes suficiente para llenar el array buff

RandomAccessFile permite grabar y leer del fichero. También permite saltar con `seek(..)` y establecer la posición por la que el S.O. lee o escribe.

Dispone de los mismos métodos que el `DataOutputStream` y el `DataInputStream`.

Leer serialización nativa

```
ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("archivo.dat", true));
ObjectInputStream ois = new ObjectInputStream(new FileInputStream("archivo.dat", true));
```

```
ois.readObject()
```

Object

entero

Lee un

double

Leer

una

cadena

`ObjectInputStream` dispone de los mismos métodos que `DataInputStream`.

Escribir Serialización nativa

```
Random AccessFile f = new Random Access File ("nombre", "rw");
```

```
f.writeInt(n)
```

```
f.writeDouble(d)
```

```
f.write(buff)
```

```
f.seek(pos)
```

```
int n = f.readInt()
```

```
double d = f.readDouble()
```

ObjectOutputStream oos = new ObjectOutputStream("nombre", "rw");

oos.writeObject(o)

Abre un
archivo de
acceso
aleatorio
que
permite
leer y
escribir
simult-
áne-
amente.
"rw"=
read/write
Escribe
un entero
Escribe
un double
Escribe
un array
de bytes.

Establece

Los objetos serializables deben implementar la interface java.io.Serializable, que no conlleva obligaciones.

Puede guardarse más de un objeto en un stream, que deben recuperarse luego en el mismo orden.

ObjectOutputStream también tiene los mismos métodos que DataOutputStream.

byte
especi-
ficado,
para que
la
siguiente
operación
de lectura
o
escritura
se
produzca
a partir
de ese
byte.

leer un
entero

Leer un
double

