Cheatography

# Ficheros en java Cheat Sheet

by victorjfg via cheatography.com/35914/cs/11925/

## Escribir texto

| | |
|---|---|
| `PrintWriter fOut = new PrintWriter(new BufferedWriter(new FileWriter("nombre_fichero")));` | Apertura del PrintWriter como Stream handler utilizando un buffer de salida para acelerar las operaciones. |
| `PrintWriter fOut = new PrintWriter((new FileWriter("nombre_fichero"));` | Apertura sin buffer |
| `PrintWriter fOut = new PrintWriter("nombre_fichero");` | Similar al anterior |
| `fOut.print(dato);` | Imprimir un dato. |
| `fOut.println(dato);` | Imprimir un dato y cambio de línea |
| `fOut.println();` | Imprimir un cambio de línea en el fichero |
| `fOut.format("cadena_de_formato", dato, dato...);` | Imprimir un dato con formato |
| `fOut.format(Locale.ROOT, "cadena_de_formato", dato, dato...);` | Imprimir un dato con formato en cultura neutra |
| `fOut.printf("cadena_de_formato", dato, dato...);` | printf es equivalente a format |

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", true)));`

## Leer texto

| | |
|---|---|
| `Scanner fIn = new Scanner(new BufferedReader(new FileReader("nombre_fichero")));` | Apertura del stream, con buffer de entrada. |
| `Scanner fIn = new Scanner(new FileReader("nombre_fichero"));` | Sin buffer |
| `fIn.nextLine();` | Lee una línea de la entrada. |

## Leer texto (cont)

| | |
|---|---|
| `fIn.hasNextLine()` | Devuelve `true`si hay una línea preparada en la entrada. |

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");` | Un objeto de tipo File representa a un fichero o una carpeta. Se puede especificar un nombre con trayectoria absoluta o relativa. |
| `f.delete()` 👉 boolean | Borrar el fichero. True si éxito. |
| `f.exists()` 👉 boolean | True si el fichero existe. |
| `f.getAbsolutePath()` 👉 String | La trayectoria absoluta |
| `f.getParent()` 👉 String | El directorio de nivel superior |
| `f.isDirectory()` 👉 boolean | True si el nombre representa a un directorio. False si es un fichero u otra cosa. |
| `f.mkdir()` 👉 boolean | Crea un directorio con el nombre especificado. True si éxito |
| `f.renameTo(new File("nuevo_nombre"))` 👉 boolean | Cambiar nombre. True si éxito |
| `f.list()` 👉 String[] | Si f representa a un directorio, lista los nombres de los elementos que contiene. |

## Escribir binario secuencial

| | |
|---|---|
| `DataOutputStream fOut = new DataOutputStream(new FileWriter(nombre));` | Abrir stream handler |
| `fOut.writeInt(dato)` | Escribe un entero |
| `fOut.writeDouble(dato)` | Escribe un double |

# Ficheros en java Cheat Sheet

by victorjfg via cheatography.com/35914/cs/11925/

## Escribir binario secuencial (cont)

| | |
|---|---|
| `fOut.writeUTF(dato)` | 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("nombre"));` | Abrir stream handler |
| `int n = fIn.readInt();` | Lee un entero |
| `double d = fIn.readDouble();` | Lee un double |
| `String s = fIn.readUTF();` | Leer una cadena |

Dispone de métodos para leer todos los tipos primitivos.
Dispone de un método para leer un número arbitrario de bytes.

## Escribir / leer binario aleatorio

| | |
|---|---|
| `RandomAccessFile f = new RandomAccessFile ("nombre", "rw");` | Abre un fichero de acceso aleatorio que permite leer y escribir simultáneamente. "rw"= read/write |
| `f.writeInt(n)` | Escribe un entero |
| `f.writeDouble(d)` | Escribe un double |
| `f.write(buff)` | Escribe un array de bytes. |
| `f.seek(pos)` | Establece el desplazamiento del puntero del fichero al byte especificado, para que la siguiente operación de lectura o escritura se produzca a partir de ese byte. |
| `int n = f.readInt()` | leer un entero |
| `double d = f.readDouble()` | Leer un double |

## Escribir / leer binario aleatorio (cont)

| | |
|---|---|
| `f.read(buff)` | 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

| | |
|---|---|
| `ObjectInputStream ois = new ObjectInputStream(new FileInputStream("nombre"));` | Abrir stream handler |
| `ois.readObject()` 👉 Object | Lee un objeto y lo devuelve con una referencia a Object. Para engancharlo con una referencia más específica se puede hacer un typecast |

ObjectInputStream dispone de los mismos métodos que DataInputStream.

## Escribir Serialización nativa

| | |
|---|---|
| `ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("nombre"));` | Apertura del Stream handler |
| `oos.writeObject(o)` | Escribe el objeto apuntado por la referencia. Si el objeto contiene referencias a otros objetos, se serializan todos si es posible |

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.