Cheatography

Ficheros en java Cheat Sheet

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

Escribir texto

```
PrintW riter fOut = new PrintW rit er(new Buffer edW rit er(new FileWr ite r("n omb re_ fic her o")));
```

```
PrintW riter fOut = new PrintW rit er((new FileWr ite r("n omb re_ fic her o"));
```

```
PrintW riter fOut = new PrintW rit er( " nom bre _fi che ro");
```

```
fOut.p rin t(d ato);
```

```
fOut.p rin tln (dato);
```

```
fOut.p rin tln();
```

```
fOut.f orm at( " cad ena _de _fo rma to", dato, dato...);
```

```
fOut.f orm at( Loc ale.ROOT, " cad ena _de _fo rma to", dato, dato...);
```

```
fOut.p rin tf( " cad ena _de _fo rma to", dato, dato...);
```

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

Puede abrirse para añadir al final mediante el constructor del FileWriter: `PrintW riter fOut = new PrintW rit er(new Buffer edW rit er(new FileWr ite r("n omb re_ fic h`

Leer texto (cont)

`f.readL ine()`	Devuelve si hay una línea 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("n omb re");`	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.getA bso lut ePath()` 👉 String	La trayectoria absoluta
`f.getP arent()` 👉 String	El directorio de nivel superior
`f.isDi rec tory()` 👉 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.rena meT o(new File("n uev o_n omb re"))` 👉 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

Apertura preparada del PrintW-riter como Stream handler utilizando un buffer de salida para acelerar las operac-iones.

Apertura sin buffer

Similar al anterior

Imprimir un dato.

Imprimir un dato y cambio de línea

Imprimir un cambio de línea en el fichero

Imprimir un dato con formato

Imprimir un dato con formato en cultura nelista

printf es equiva-lente a format

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

```
DataOu tpu tStream fOut = new DataOu tpu tSt rea m(ne
 omb re));
```

Leer texto

```
Scanner fIn = new Scanne r(new Buffer edR ead er(new FileRe ade r("n omb re_ fic -
her o"))):
```

```
fOut.w rit eIn t(dato)
```

Apertura del stream, con buffer de entrada.

```
fOut.w rit eDo ubl e(dato)
```

```
Scanner fIn = new Scanne r(new FileRe ade r("n omb re_ fic her o")):
```

Sin buffer

```
fIn.ne xtL ine();
```

Lee una línea de la entrada.

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

Published 28th May, 2017.
Last updated 28th May, 2017.

Cheatography

Ficheros en java Cheat Sheet

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

Escribir binario secuencial (cont)

`fOut.w rit eUT F(da to)`	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

```
DataIn put Stream fIn = new DataIn put Str eam(new FileInp utS tre am("fich ero mbr e"));
```

Abrir stream handler

```
int n = fIn.re adI nt();
```

Leer entero

```
double d = fIn.re adD oub le();
```

Lee un double

```
String s = fIn.re adU TF();
```

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

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

```
Object Inp utS tre am ois = new Object Inp utS tre am(new FileInp utS tre am("fich ero mbr e"));
```

Abrir stream handler

```
ois.re adO bject() // Object
```

Leer Object

ObjectInputStream dispone de los mismos métodos que DataInput-Stream.

Escribir Serialización nativa

`Random Acc essFile f = new Random Acc ess Fil e("n omb re", " rw");`	Abre un fichero de acceso aleatorio que permite leer y escribir simult-áne-amente. "rw"= read/write
`f.writ eInt(n)`	Escribe un entero
`f.writ eDo uble(d)`	Escribe un double
`f.writ e(buff)`	Escribe un array de bytes.
`f.seek (pos)`	Establece el despla-zamiento del puntero del fichero al byte especi-ficado, para que la siguiente operación de lectura o escritura se produzca a partir de ese byte.
`int n = f.read Int()`	leer un entero
`double d = f.read Dou ble()`	Leer un double

`Object Out put Stream oos = new Object Out put Str ea n omb re();`

`oos.wr ite Obj ect(o)`

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.