DEFINICIÓN

Es una secuencia de bytes en un dispositivo de almacenamiento tales como disco duro, CD, DVD, memoria USB, etc. Se puede leer y/o escribir. Se identifica mediante un nombre conocido como pathname.

La lectura y escritura de ficheros es muy importante porque permite a nuestros programas dialogar e intercambiar información con el usuario. La entrada/salida se realiza mediante el empleo de flujo de datos. Un flujo es una secuencia ordenada de datos que se transmite desde una fuente hasta un destino. En Java se denomina Stream. El origen o el destino pueden ser un fichero, un String, un dispositivo (teclado, altavoz, pantalla…), etc.

FUJO DE DATOS

La entrada/salida se organiza generalmente mediante objetos llamados Streams. Un stream es la generalización de un fichero, es decir: su secuencia es ordenada y su origen o destino pueden ser un fichero, un String, otro dispositivo, etc. Para poder usar un stream primero hay que abrirlo. Esto se hace en el momento de su creación y se cierra cuando se deja de utilizar. Las clases relacionadas con streams se encuentran definidas en el paquete java.io, abreviatura de Input/Output.

CLASIFICACIÓN DE LOS STREAMS

Por el tipo de datos que transportan:

* Binarios: de bytes.
* Caracteres: de texto.

Por el sentido del flujo de datos:

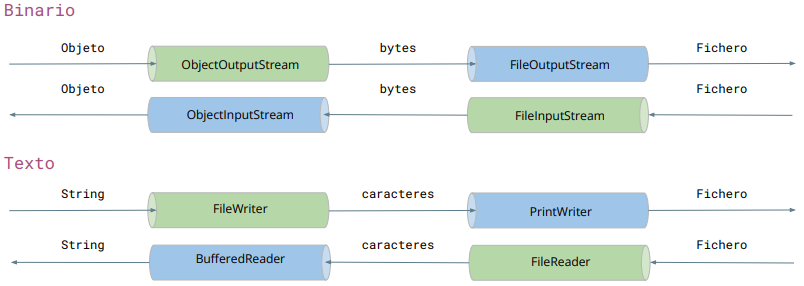
* Entrada: los datos fluyen desde el fichero hacia el programa.
* Salida: los datos fluyen desde el programa hacia el fichero.

Según su cercanía al dispositivo:

* Iniciadores: directamente vuelcan o recogen los datos del dispositivo.
* Filtros: se sitúan entre un stream iniciador y el programa.

USO DE LOS STREAMS

Para la leer y escribir de fichero se suele usar un stream iniciador y un filtro. Esto nos permite hablar “la misma lengua” que el fichero y comunicarnos con él correctamente.



JERARQUÍA DE CLASES STREAMS BINARIOS

OutputStream: escritura de ficheros binarios.

* FileOutputStream: escribe bytes en un fichero.
* ObjectOutputStream: convierte objetos y variables en arrays de bytes que pueden ser escritos en un OutputStream.

InputStream: lectura de ficheros binarios.

* FileInputstream: lee bytes de un fichero.
* ObjectInputStream: convierte en objetos y variables los arrays de bytes leídos de un InputStream.

JERARQUÍA DE CLASES DE STREAMS DE CARACTERES

Writer: escritura de ficheros de texto.

* FileWriter: escribe texto en un fichero.
* PrintWriter: permite convertir a texto variables y objetos para escribirlos en un Writer.

Reader: lectura de ficheros de texto.

* FileReader: lee texto de un fichero.
* BufferedReader: lee texto (línea a línea) de un Reader.

OBJETOS STREAM PREDEFINIDOS

* System.out: salida estándar por consola. Es un objeto de la clase PrintStream (subclase de OutputStream)
* System.err: salida de error por consola. También es un objeto de la clase PrintStream.
* System.in: entrada estándar por teclado. Es un objeto de la clase InputStream.

FICHEROS BINARIOS

MISCELÁNEA DE CLASES

FileOutputStream: permite escribir bytes hacia un fichero binario (ejemplo página 222).

BufferedOutputStream: permite escribir información a otro OutputStream utilizando un buffer interno que mejora el rendimiento (ejemplo página 222).

DataOutputStream: define algunos métodos que permiten escribir datos de tipo primitivo desde un OutputStream (ejemplo página 222).

ByteArrayOutputStream: permite usar un array de bytes como un OutputStream (ejemplo página 223).

PrintStream: permite escribir los datos de un stream en otro automáticamente y sin necesidad de invocar al método flush() (ejemplo página 223).

FileInputStream: permite leer bytes desde un fichero binario (ejemplo página 223).

BufferedInputStream: permite leer información de otro InputStream utilizando un buffer que mejora el rendimiento (ejemplo página 224).

DataInputStream: define métodos que permiten leer un tipo de datos primitivo desde un InputStream (ejemplo página 224).

ByteArrayInputStream: permite usar un array de bytes como InputStream (ejemplo página 224).

SequenceInputStream: flujo de entrada que puede combinar varios InputStream en uno (ejemplo página 225).

FICHEROS DE TEXTO

MISCELÁNEA DE CLASES

FileWriter: permite escribir caracteres hacia un fichero de texto (ejemplo página 226).

BufferedWriter: permite escribir información de otro Writer utilizando un buffer interno que mejora el rendimiento (ejemplo en página 226).

PrintWriter: es la implementación de la clase Writer. Permite escribir una cadena de caracteres en cualquier OutputStream (ejemplo en página 227).

StringWriter: flujo de caracteres que recoge su salida de un buffer y que puede utilizarse para construir un String (ejemplo en página 227).

FileReader: flujo de caracteres para la lectura de ficheros de texto (ejemplo página 228).

BufferedReader: permite leer información de otro Reader utilizando un buffer interno que mejora el rendimiento (ejemplo página 228).

LineNumberReader: tipo especial de BufferedReader que lleva un conteo del número de línea que se está leyendo en cada momento (ejemplo página 228).

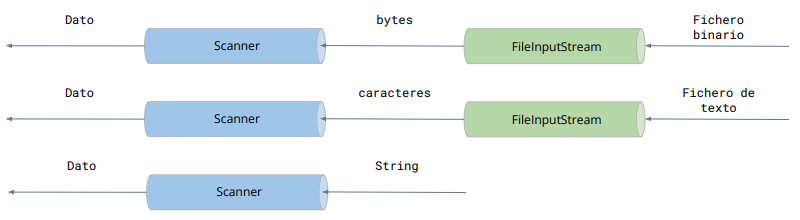
CharArrayReader: recoge de un buffer los datos provenientes de un array de caracteres (ejemplo página 228).

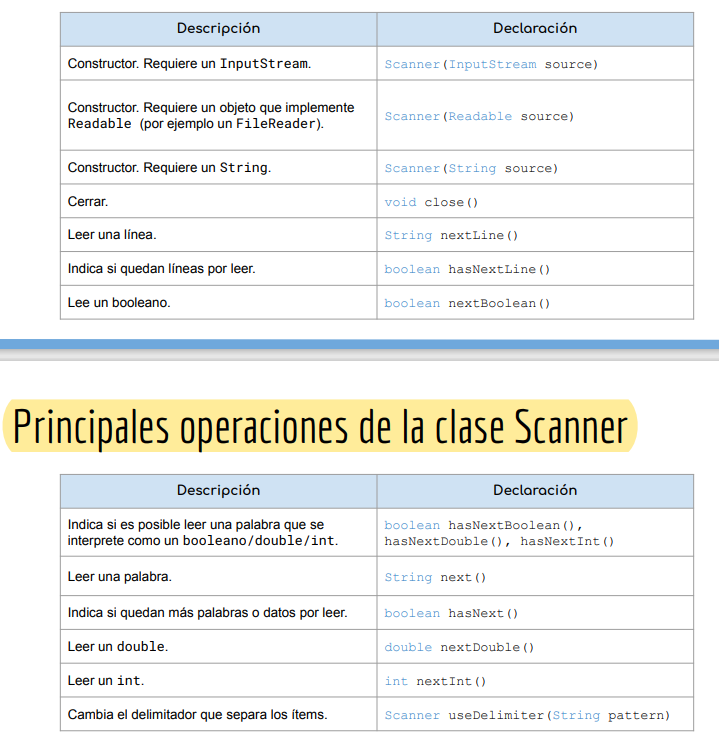
StringReader: recoge de un buffer los datos provenientes de un String (ejemplo página 229).

LECTURA Y ESCRITURA EN FICHEROS DE TEXTO CON LA CLASE SCANNER

LECTURA CON LA CLASE SCANNER

La clase Scanner (paquete java.util) permite leer número y texto de un fichero de texto y de otras fuentes. Permite la lectura de ficheros línea a línea (muy similar a BufferedReader). Permite la lectura sencilla de números y palabras separados por algún separador específico. El separador por defecto es cualquier espacio en blanco (espacio, salto de línea, tabulador, etc.). Permite reconocer patrones de texto conocidos como expresiones regulares. Además, la clase Scanner puede leer de un InputStream y de un FileReader.





EXCEPCIONES DE LA CLASE SCANNER

El uso de estas operaciones puede generar algunas excepciones tales como:

* NoSuchElementException: no quedan más palabras.
* IllegalStateException: el scanner está cerrado.
* InputMismatchException: el dato leído no es del tipo esperado.