

Java Standard Web Programming

Módulo 7

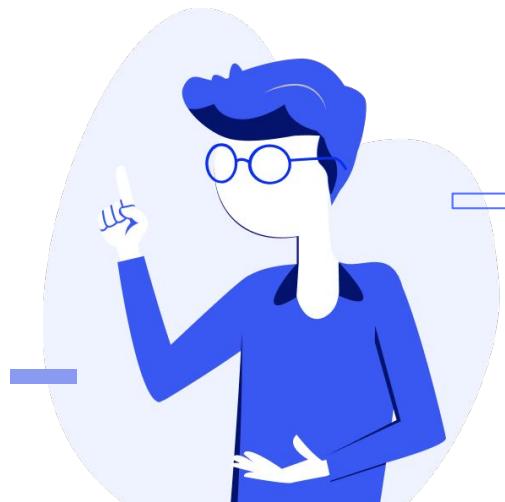
Java IO

Introducción

Java ofrece un paquete completo para gestionar la comunicación (Lectura y Escritura) de bytes.

El paquete ***IO Input*** (Entrada) y ***Output*** (Salida) permite, a través de ***Streams*** (Flujos), realizar la **comunicación con la consola, archivos, y otros.**

Podemos dividir en dos secciones la comunicación: **Orientados a Carácter** y **Orientados a Bytes.**



File

Antes de comenzar con la lectura y escritura de archivos, veremos la clase **File** que se encuentra también en el paquete **Java.io** que proporciona una serie de métodos para obtener información de los directorios y archivos que serán de mucha utilidad.

Veamos estos métodos en las siguientes diapositivas.



Métodos de la clase File

Tipo	Método	Descripción
boolean	canRead()	Comprueba si la aplicación puede leer el archivo indicado por esta ruta.
boolean	canWrite()	Comprueba si la aplicación puede modificar el archivo indicado por esta ruta.
boolean	createNewFile()	Crea de forma atómica un archivo nuevo y vacío si y solo si aún no existe un archivo con este nombre.
boolean	delete()	Elimina el archivo o directorio indicado por esta ruta.
boolean	exists()	Comprueba si existe el archivo o directorio indicado por esta ruta.
String	getName()	Devuelve el nombre del archivo o directorio indicado por esta ruta.
String	getAbsolutePath()	Devuelve la cadena de nombre de ruta absoluta de esta ruta.

Métodos de la clase File

Tipo	Método	Descripción
boolean	<code>isDirectory()</code>	Comprueba si el archivo indicado por esta ruta es un directorio.
boolean	<code>isFile()</code>	Comprueba si el archivo indicado por esta ruta es un archivo normal.
boolean	<code>isHidden()</code>	Comprueba si el archivo nombrado por esta ruta es un archivo oculto.
<code>String[]</code>	<code>list()</code>	Devuelve una matriz de cadenas que nombran los archivos y directorios en el directorio indicado por esta ruta.
<code>File[]</code>	<code>listFiles()</code>	Devuelve una matriz de nombres de ruta abstractos que denotan los archivos en el directorio indicado por esta ruta.
boolean	<code>mkdir()</code>	Crea el directorio nombrado por esta ruta.

Algoritmo

Flujos de lectura

El algoritmo utilizado, por lo general, para **leer flujos de datos** es: instanciar el objeto, recorrerlo mientras (**while**) exista un elemento que leer y, por último, cerrar el flujo.

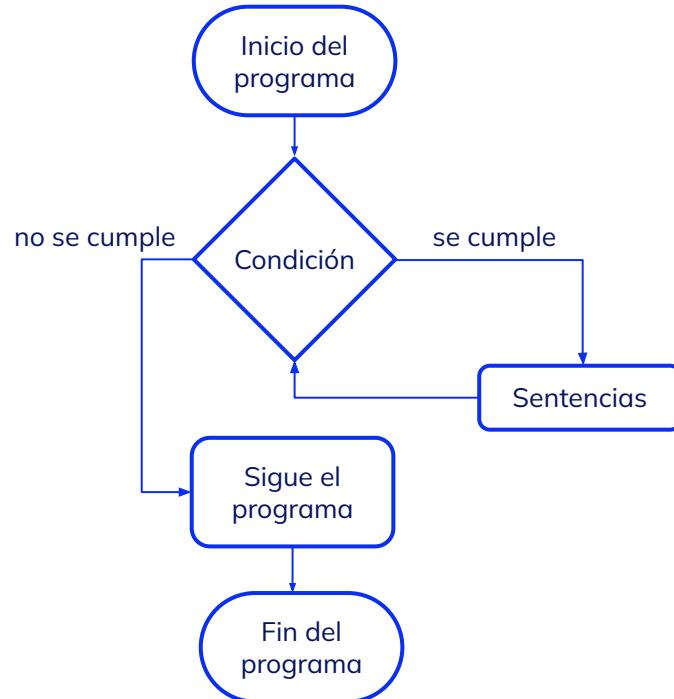
Los flujos de lectura proporcionan el método **read** que retorna el byte o carácter a leer.

Flujos de escritura

El algoritmo utilizado, por lo general, para **escribir flujos de datos** es: instanciar el objeto, recorrerlo para (**for**) cada elemento que exista escribir y por último cerrar el flujo.

Los flujos de escritura proporcionan el método **write** que escribe el dato u objeto a escribir.

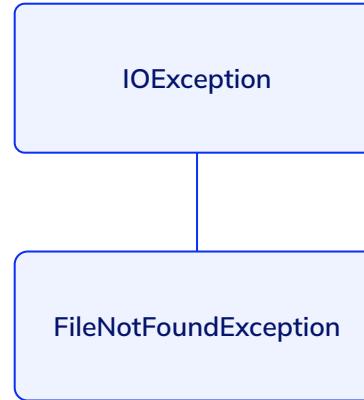
Ejemplo



Excepciones

La excepción **IOException** se encargará de **lanzar errores cuando exista un problema al tratar de leer un fichero**. El problema puede ocurrir porque se encuentra bloqueado o porque se interrumpió la comunicación.

La excepción **FileNotFoundException** se encargará de **lanzar un error si no encuentra el fichero especificado en la ruta especificada**. Generalmente, está en los constructores.

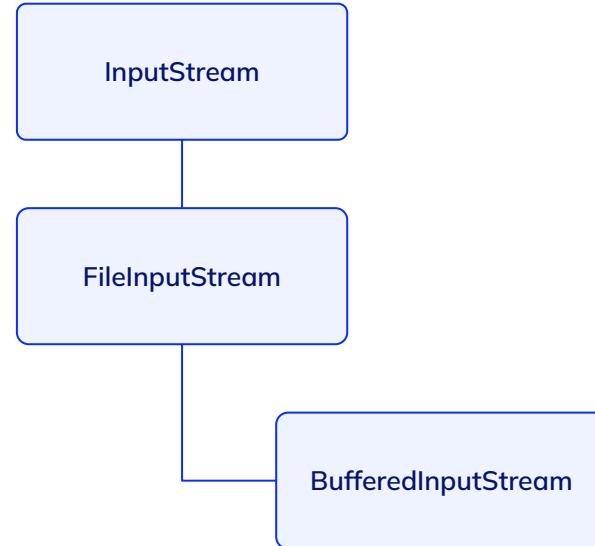


InputStream

La clase abstracta **InputStream** representa un flujo de bytes y proporciona los métodos para acceder a la información.

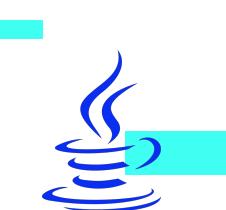
La clase **FileInputStream** reescribe todos los métodos de la clase **InputStream** proporcionando una implementación más específica y adecuada.

Esta clase lee la información en byte, que no es legible para el ser humano, y se utiliza en la mayoría de los casos para leer imágenes, audios, videos, y otros.



Métodos

Tipo	Método	Descripción
int	<code>available()</code>	Retorna el número estimado de bytes del fichero (tamaño del fichero).
void	<code>close()</code>	Cierra este flujo de entrada y libera los recursos del sistema asociados con el flujo.
int	<code>read()</code>	Lee el siguiente byte de datos del flujo de entrada.
byte[]	<code>readAllBytes()</code>	Lee todos los bytes restantes del flujo de entrada.

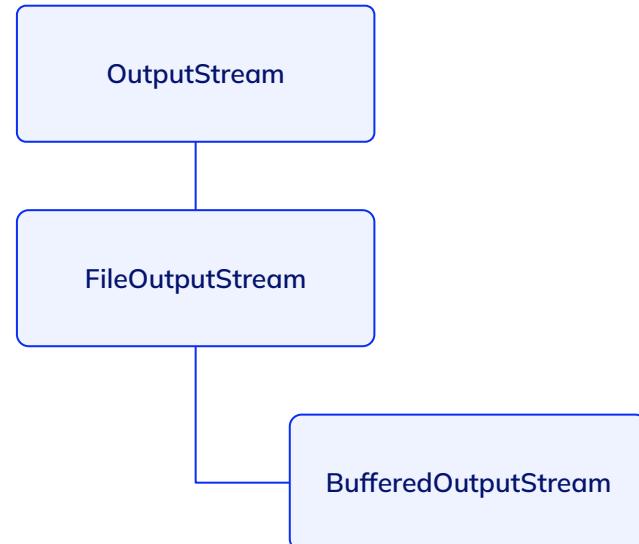


OutputStream

La clase abstracta **OutputStream** representa un flujo de bytes y proporciona los métodos para **acceder a la información**.

La clase **FileOutputStream** reescribe todos los **métodos de la clase OutputStream** proporcionando una implementación más específica y adecuada.

Esta clase **escribe la información en byte** y se utiliza, en la mayoría de los casos, para leer imágenes, audios, videos, y otros.



Métodos

Tipo	Método	Descripción
int	<code>available()</code>	Retorna el número estimado de bytes del fichero (Tamaño del fichero).
void	<code>close()</code>	Cierra este flujo de entrada y libera los recursos del sistema asociados con el flujo.
void	<code>write(byte[] b)</code>	Escribe el arreglo de bytes en el flujo de salida.
void	<code>write(int b)</code>	Escribe el byte especificado del flujo de salida.

InputStream

```
try (FileInputStream archivoBinario = new FileInputStream(fichero)) {  
    byte[] archivoBytes = archivoBinario.readAllBytes();  
} catch (FileNotFoundException e) {  
    e.printStackTrace();  
} catch (IOException e) {  
    e.printStackTrace();  
}
```

OutputStream

```
try (FileOutputStream archivoBinario = new FileOutputStream(fichero)) {  
    archivoBinario.write(bytes);  
} catch (FileNotFoundException e) {  
    e.printStackTrace();  
} catch (IOException e) {  
    e.printStackTrace();  
}
```

**¡Sigamos
trabajando!**