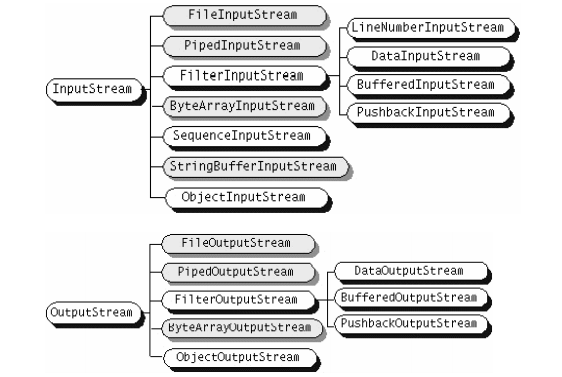
1. **TIPOS DE FLUJOS O STREAMS.**

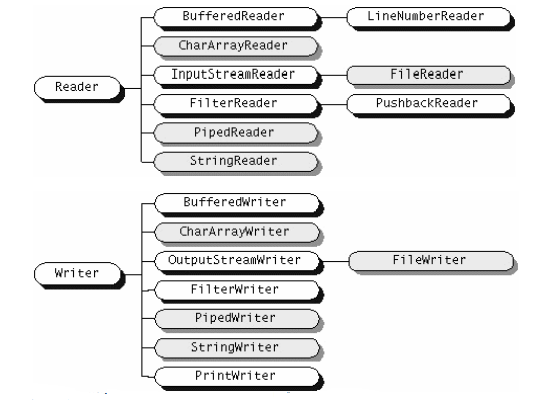
El sistema de entrada/salida en Java presenta una gran cantidad de clases que se implementan en el paquete java.io. Usa la abstracción del flujo (stream) para tratar la comunicación de información entre una fuente y un destino.

* Flujos de bytes (8 bits): Orientados a la lectura/escritura de **datos binarios**. Todas las clases descienden de *InputStream* y *OutputStream*
* Flujos de caracteres (16 bits): realizan operaciones de entrada/salida de **caracteres**. Gobernados por las clases *Reader* y *Writer*.
  1. **JERARQUÍA DE CLASES PARA FLUJOS DE BYTES**



Las clases **FileInputStream** y **FileOutputStream** que manipulan los flujos de bytes provenientes o dirigidos a ficheros en disco son las que utilizaremos.

* 1. **JERARQUÍAS DE CLASES PARA FLUJOS DE CARACTERES**



1. **FORMAS DE ACCESO A UN FICHERO**

* **Secuencial**: los datos o registros se leen y escriben en orden. Si se quiere acceder a un dato o registro hay que pasar por todos los anteriores. La escritura se hará a partir del último escrito, no es posible hacer inserciones entre los datos que ya hay escritos.
* **Directo o aleatorio**: acceso a la información en cualquier orden. Los datos se almacenan en registros de tamaño conocido.

1. **OPERACIONES SOBRE FICHEROS**

* Creación del fichero.
* Apertura del fichero.
* Cierre del fichero.
* Lectura de datos del fichero.
* Escritura de datos en el fichero.
* Altas.
* Bajas.
* Modificaciones.
* Consultas.

1. **FICHEROS DE TEXTO**

Estos ficheros son los que normalmente se generan con un editor, almacenan caracteres alfanuméricos en un formato estándar (ASCII, UNICODE, UTF8, etc.). Se manejan con las clases FileReader y FileWriter.

Las clases **FileReader y FileWriter** permiten leer y escribir, respectivamente, en un fichero.

Lo primero que debemos hacer es importar estas clases y las que controlan las excepciones.

Cuando creamos un objeto, abrimos un **stream** entre nuestro programa y el exterior, cuando dejemos de usarlo debemos cerrar el **stream** con el método **close().**

* 1. Lectura de un fichero de texto

|  |
| --- |
| **Constructor and Description** |
| [FileReader](http://docs.oracle.com/javase/7/docs/api/java/io/FileReader.html#FileReader(java.io.File))([File](http://docs.oracle.com/javase/7/docs/api/java/io/File.html) file)  Creates a new FileReader, given the File to read from. |
| [FileReader](http://docs.oracle.com/javase/7/docs/api/java/io/FileReader.html#FileReader(java.io.FileDescriptor))([FileDescriptor](http://docs.oracle.com/javase/7/docs/api/java/io/FileDescriptor.html) fd)  Creates a new FileReader, given the FileDescriptor to read from.  (Trabaja a un nivel de sistema más bajo) |
| [FileReader](http://docs.oracle.com/javase/7/docs/api/java/io/FileReader.html#FileReader(java.lang.String))([String](http://docs.oracle.com/javase/7/docs/api/java/lang/String.html) fileName)  Creates a new FileReader, given the name of the file to read from. |

**Métodos de lectura de la clase FileReader**

(estos métodos devuelven el número de caracteres leídos o -1 si se ha llegado al final del fichero)

* int read(): Devuelve el ASCII del carácter leído
* int read (char buff[]): Lee buff.length caracteres y los almacena en buff
* int read (char [] buff, int desplazamiento, int n): lee n caracteres de datos y los almacena en buff a partir de la posición buff[desplazamiento]

Un ejemplo:

FileWriter fw=**new** FileWriter("D:\\fichero1.txt");

FileReader fr=**new** FileReader("D:\\fichero1.txt");

Si usas **FileWriter** y escribes una ruta de fichero que no existe lo crea, para **FileReader** si que debe existir el fichero, sino lanzara una excepción. Usamos doble barra (\\) por que es un carácter de escape, para poner \.

Para escribir, usaremos el método **write** de **FileWriter**, este método puede usar como parámetro un String con lo que queremos escribir o un número que se corresponderá un carácter de la tabla ASCII.

Para leer, usaremos el método **read** de **FileReader**, este método no tiene parámetros pero devuelve un número que si le hacemos un casting a **char** este sera legible por nosotros. Esto lo podemos mostrar por pantalla o incluso pasarlo a otro fichero (crear otro objeto).

Cuando se termina el fichero, el método read devuelve **-1.**

**Ejemplo1**

**import java.io.FileReader;**

**import** java.io.FileWriter;

**import** java.io.IOException;

**public** **class** prueba1 {

**public** **static** **void** main(String[] args) {

**try**{

//Abro stream, crea el fichero si no existe

FileWriter fw=**new** FileWriter("D:\\fichero1.txt");

//Escribimos en el fichero un String y un caracter 97 (a)

fw.write("Esto es una prueba");

fw.write(97);//el carcater a

//Cierro el stream

fw.close();

//Abro el stream, el fichero debe existir

FileReader fr=**new** FileReader("D:\\fichero1.txt");

//Leemos el fichero y lo mostramos por pantalla

**int** valor=fr.read();

**while**(valor!=-1){

System.***out***.print((**char**)valor);

valor=fr.read();

}

//Cerramos el stream

fr.close();

}**catch**(IOException e){

System.***out***.println("Error E/S: "+e);

}

}

}

La idea es abrir un **fichero**, escribir o leer y cerrar el **fichero**. No es necesario que hagamos nada para mover el puntero del fichero, cuando leemos un carácter automáticamente el puntero se mueve.

Podemos tambien usar el método **flush()** de **FileWriter** que hace que se guarden los cambios. Veamos como queda:

**public** **static** **void** main(String[] args) {

**try {**

FileWriter fw=**new** FileWriter("D:\\fichero1.txt");

            FileReader fr=**new** FileReader("D:\\fichero1.txt")){

            //Escribimos en el fichero un String y un caracter 97 (a)

            fw.write("Esto es una prueb");

            fw.write(97);

            //Guardamos los cambios del fichero evita tener que cerrar el fichero

            fw.flush();

 //Leemos el fichero y lo mostramos por pantalla además el flush coloca el puntero al inicio del archivo

**int** valor=fr.read();

**while**(valor!=-1){

                System.out.print((**char**)valor);

                valor=fr.read();

            }

        }**catch**(IOException e){

            System.out.println("Error E/S: "+e);

        }finally{

fr.close();

fw.close();

}

    }

**Ejecicio 2**

**crea la clase EscribirLeer, que incluya el metodo main para probar los metodos**

**public static void escribeFichero(FileWriter fw,String cadena) throws IOException**

**public static void leeFichero(FileReader fr) throws IOException**

**En main debe pedirse al usuario que escriba una cadena, que se guardará en un fichero invocando a escribeFichero y posteriormente se leeará el fichero invocando a leeFichero**

**Ejemplo 3**

public class Ejemplo3 {

public static void main(String[] args) {

FileReader fic = null;

try {

//Para probarlo crearemos un fichero con un par de frases

// SE PUEDEN PROBAR LOS DOS MODOS DE APERTURA

// File fichero = new File("C:\\Ficheros\\fichero1.txt");

// FileReader fic = new FileReader(fichero);

fic = new FileReader("C:\\Ficheros\\fichero1.txt");

int i;

while ((i = fic.read()) != -1) //Devuelve el ASCII del carácter leído o -1 si es final

{

char caracter = (char) i;

System.out.print(caracter);

}

} catch (FileNotFoundException ex) {

System.out.println("Error de apertura de fichero");

} catch (IOException ex) {

System.out.println("Error dde lectura del fichero");

} finally {

fic.close();

}

}

* 1. **Escritura en un fichero de texto**

|  |
| --- |
| Constructor and Description |
| [FileWriter](http://docs.oracle.com/javase/7/docs/api/java/io/FileWriter.html#FileWriter(java.io.File))([File](http://docs.oracle.com/javase/7/docs/api/java/io/File.html) file)  Constructs a FileWriter object given a File object. |
| [FileWriter](http://docs.oracle.com/javase/7/docs/api/java/io/FileWriter.html#FileWriter(java.io.File,%20boolean))([File](http://docs.oracle.com/javase/7/docs/api/java/io/File.html) file, boolean append)  Constructs a FileWriter object given a File object. |
| [FileWriter](http://docs.oracle.com/javase/7/docs/api/java/io/FileWriter.html#FileWriter(java.io.FileDescriptor))([FileDescriptor](http://docs.oracle.com/javase/7/docs/api/java/io/FileDescriptor.html) fd)  Constructs a FileWriter object associated with a file descriptor. |
| [FileWriter](http://docs.oracle.com/javase/7/docs/api/java/io/FileWriter.html#FileWriter(java.lang.String))([String](http://docs.oracle.com/javase/7/docs/api/java/lang/String.html) fileName)  Constructs a FileWriter object given a file name. |
| [FileWriter](http://docs.oracle.com/javase/7/docs/api/java/io/FileWriter.html#FileWriter(java.lang.String,%20boolean))([String](http://docs.oracle.com/javase/7/docs/api/java/lang/String.html) fileName, boolean append)  Constructs a FileWriter object given a file name with a boolean indicating whether or not to append the data written. |

**Métodos de escritura de la clase FileWriter:**

* void write(int c)
* void write(char[]buff)
* void write(char []buff, int desplazamiento, int n)
* void write(String str)
* append (char c)

**Ejemplo 3**: Escritura de cadenas en un fichero. Comprobad la diferencia entre poner el segundo parámetro en FileWriter y no ponerlo.

public class Ejemplo3 {

public static void main(String[] args) **throws FileNotFoundException, IOException** {

//Se propone como ejemplo de throws

File fichero = new File("C:\\FICHEROS\\FICHERO2.TXT");

FileWriter ficW= new FileWriter(fichero,true); //Prepara el fichero para añadir en caso de que ya exista

Scanner sc= new Scanner(System.in);

String cadena;

System.out.println("Escriba una cadena de caracteres");

cadena= sc.nextLine();

ficW.write(cadena);

ficW.close();

}

}