FLUJOS O STREAMS. TIPOS

El sistema de entrada/salida en Java tiene varias clases que se implementan en el paquete **java.io**. Usa la abstracción del flujo (**stream**) para tratar la comunicación de información entra una fuente y un destino; dicha información puede estar en un fichero en el disco duro, en la memoria, en algún lugar de la red, e incluso en otro programa. Cualquier programa que tenga que obtener información de cualquier fuente necesita abrir un stream, de la misma forma si necesita enviar información abrirá un stream y se escribirá la información en serie. La vinculación de este stream al dispositivo físico la hace el sistema de entrada y salida de Java.

Se defines dos tipos de flujos:

a) **Flujos de bytes (8 bits):** realizan operaciones de entradas y salidas de bytes y su uso está orientado a la lectura/escritura de datos binarios. Todas las clases de flujos de bytes descienden de las clases **InputStream** y **OutputStream**, cada una de ellas tienen subclases que controlan las diferencias entre los distintos dispositivos de entrada/salida que se pueden utilizar.

b) **Flujos de caracteres (16 bits):** realizan operaciones de entradas y salidas de caracteres. El flujo de caracteres está gobernado por las clases **Reader** y **Writer**. La razón de ser de estas clases es la internacionalización; la antigua jerarquía de flujos de entrada/salida solo soporta flujos de 8 bits, no manejando caracteres Unicode de 16 bits.

Ejemplos de flujos de entrada serían cuando utilizamos la clase `BufferedReader`.

Flujos de Bytes (Byte Streams)

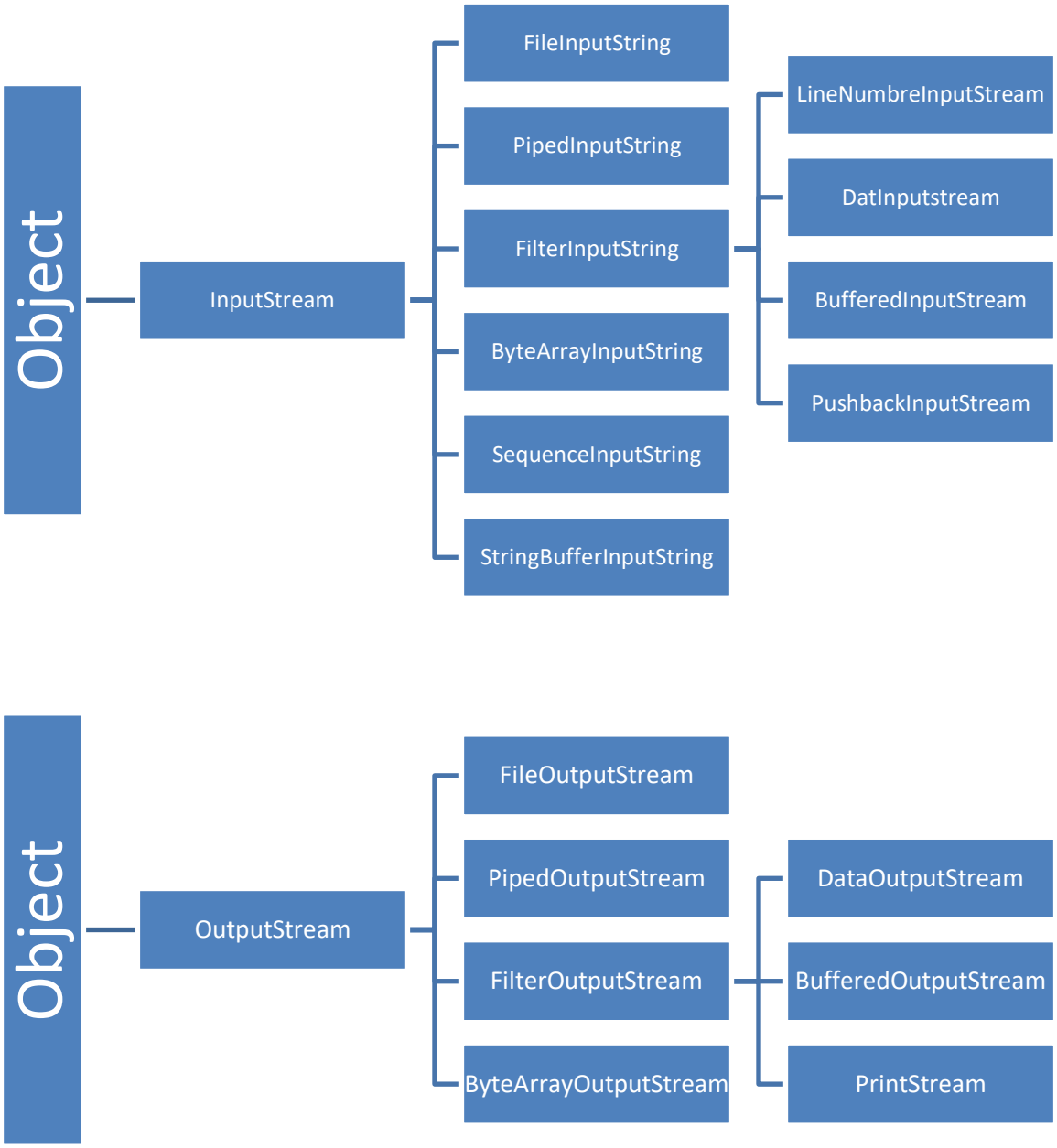
La clase InputStream representa las clases que producen entradas de distintas fuentes: un array de bytes, un objeto String, un fichero, otras fuentes como una conexión a Internet, etc. Los tipos de **InputStream** se resumen en la siguiente tabla:

CLASE	FUNCIÓN
ByteArrayInputStream	Permite usar un espacio de almacenamiento intermedio de memoria.
StringBufferInputStream	Convierte un String en un InputStream.
FileInputStream	Para leer información de un fichero.
PipedInputStream	Implementa el concepto de “tubería”.
FilterInputStream	Proporciona funcionalidad útil a otras clases InputStream.
SequenceInputStream	Convierte dos o más objetos InputStream en un InputStream único.

Los tipos OutputStream incluyen las clases que deciden donde irá la salida: a un array de bytes, a un fichero, etc. Se resumen en la siguiente tabla:

CLASE	FUNCIÓN
ByteArrayOutputStream	Crea un espacio de almacenamiento intermedio en memoria. Todos los datos que se envían al flujo se ubican en este espacio.
FileOutputStream	Para enviar información a un fichero.
PipedOutputStream	Cualquier información que se desee escribir aquí acaba automáticamente como entrada del PipedInputStream asociado. Implementa el concepto de “tubería”.
FilterOutputStream	Proporciona funcionalidad útil a otras clases OutputStream.

La siguiente figura muestra la jerarquía de clases para lectura y escritura de datos: InputStream y OutputStream



Las clases **FileInputStream** y **FileOutputStream** manipulan los flujos de bytes provenientes o dirigidos hacia los ficheros en disco.

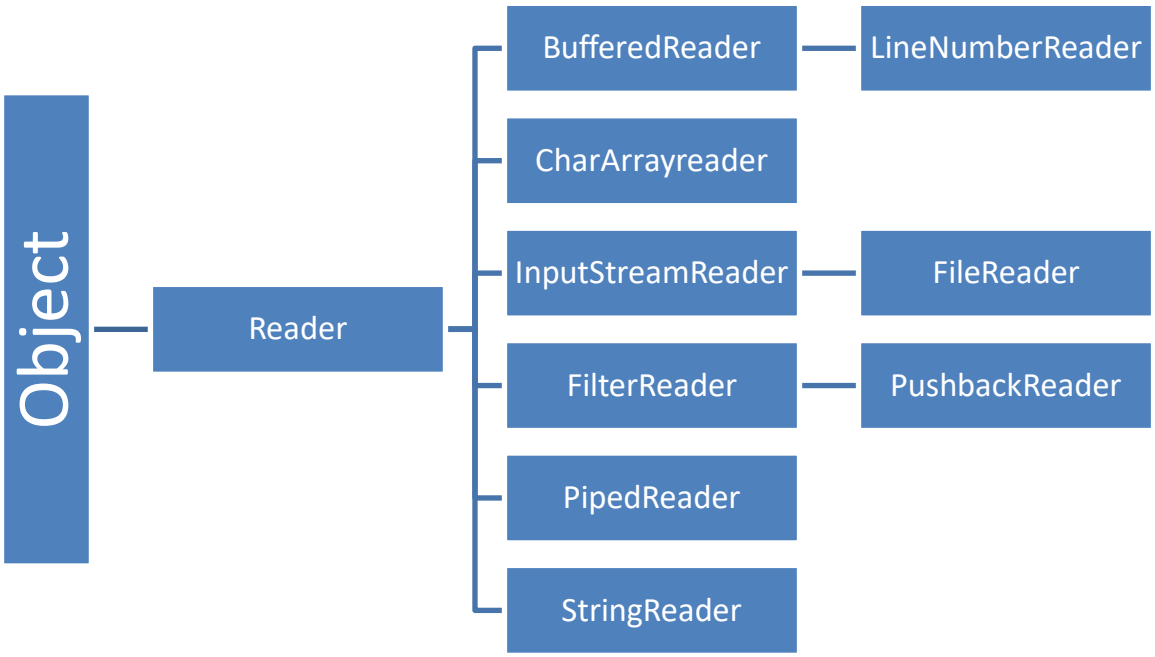
Flujos de Caracteres (Character Streams)

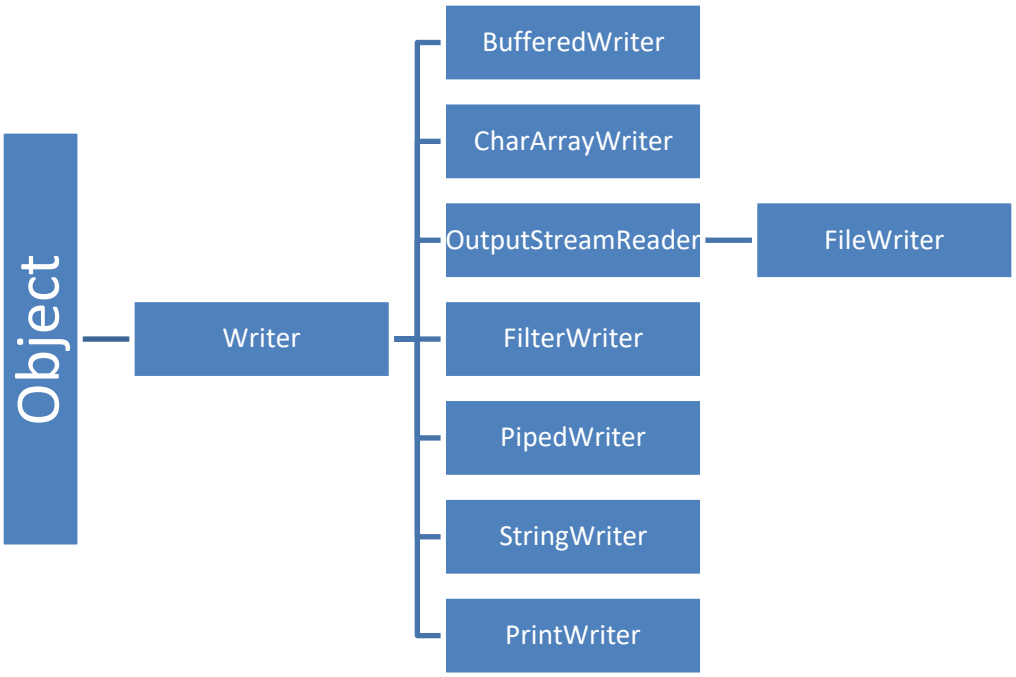
Las clases **Reader** y **Writer** manejan flujos de caracteres Unicode. En ocasiones hay que usar las clases que manejan bytes en combinación con las clases que manejan caracteres. Para lograr esto hay clases "puente" (es decir, convierte los streams de bytes a streams de caracteres): **InputStreamReader** convierte un **InputStream** en un Reader (lee bytes y los convierte en caracteres) y **OutputStreamReader** que convierte **OuputStream** en un **Writer.**

La siguiente tabla muestra la correspondencia entre las clases de flujo de bytes y de caracteres:

CLASES DE FLUJOS DE BYTES	CLASE CORRESPONDIENTE DE FLUJO DE CARACTERES
InputStream	Reader, convertidor InputStreamReader
OutputStream	Writer, convertidor OutputStreamReader.
FileInputStream	FileReader
FileOutputStream	FileWriter
StringBufferInputStream	StringReader
(sin clase correspondiente)	StringWriter
ByteArrayInputStream	CharArrayReader
ByteArrayOutputStream	CharArrayWriter
PipedInputStream	PipedReader
PipedOutputStream	PipedWriter

Las siguientes figuras muestran la Jerarquía de las clases para lectura y escritura: Reader, Writer:





Las clases de flujos de caracteres más importantes son:

- Para acceso a ficheros, lectura y escritura en ficheros: **FileReader** y **FileWriter**.
- Para acceso a caracteres, leen y escribe un flujo de caracteres en un array de caracteres: **CharArrayReader** y **CharArrarWriter**.
- Para buferización de datos: **BufferedReader** y **BufferedWriter**, se utilizan para evitar que cada lectura o escritura acceda directamente al fichero, ya que utilizan un buffer intermedio entre la memoria y el stream.

FORMAS DE ACCESO A UN FICHERO

Hay dos formas de acceso a la información almacenada en un fichero: acceso secuencial y acceso directo o aleatorio:

- **Acceso secuencial**: los datos o registros se leen y se escriben en orden. Ejemplo cintas de vídeo, cintas de magnetofón. Si se quiere acceder a un dato o a un registro hay que leer todos los anteriores. La escritura de datos se hace a partir del último registro, no es posible hacer inserciones entre los datos que ya están escritos.
- **Acceso directo o aleatorio**: permite acceder directamente a un dato o registro sin necesidad de leer los anteriores y se puede acceder a la información en cualquier orden. Los datos están almacenados en registros de un tamaño conocido, nos podemos mover de un registro a otra de forma aleatoria para leerlos o modificarlos.

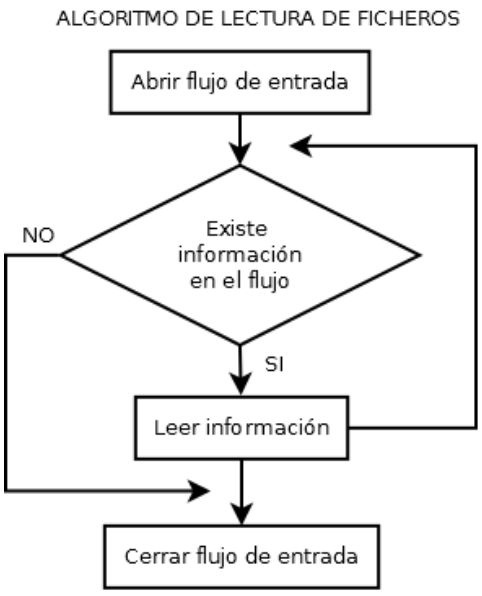
En Java el acceso secuencial más común en ficheros puede ser binario o a caracteres. Para el acceso binario se usan la clases: **FileInputStream** y **FileOutputStream**; para el acceso a caracteres (texto) se usan las clases **FileReader** y **FileWriter**. En el acceso aleatorio se utiliza la clase **RandowAccessFile**.

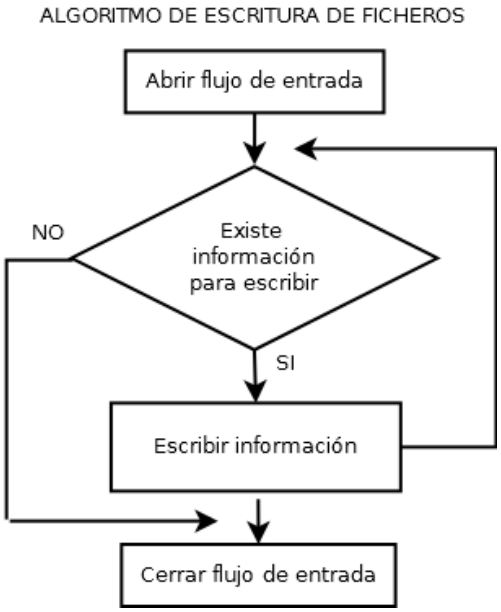
Operaciones Sobre Ficheros

Las operaciones básicas que se realizan sobre cualquier fichero independientemente de la forma de acceso al mismo son las siguientes:

- **Creación del fichero**. El fichero se crea en el disco con un nombre que después de debe utilizar para acceder a él. La creación es un proceso que se realiza una vez.
- **Apertura del fichero**. Para que un programa pueda operar con un fichero, la primera operación que hay que realizar es la apertura del mismo. El programa utilizará algún método para identificar el fichero con el que quiere trabajar, por ejemplo asignar a una variable el descriptor del fichero.
- **Lectura de los datos del fichero**. Este proceso consiste en transferir información del fichero a la memoria principal, normalmente a través de alguna variable o variables de nuestro programa en las que se depositarán los datos extraídos del fichero.
- **Escritura de datos en el fichero**. En este caso el proceso consiste en transferir información de la memoria (por medio de las variables del programa) al fichero.
- **Cierre del fichero**. El fichero se debe cerrar cuando el programa no lo vaya a utilizar. Normalmente suele ser la última instrucción del programa.

Los programas cuando quieren leer datos de un fichero, lo primero que hacen es abrir un flujo de entrada y leen la información que contiene el fichero mediante el flujo de datos de entrada. Para grabar datos la operación es similar se abre un flujo de salida y el programa va escribiendo los datos en el flujo de salida y de esta forma se almacenan los datos.





Normalmente las operaciones típicas que se realizan sobre un fichero una vez abierto son las siguientes:

- **Altas**: consiste en añadir un nuevo registro al fichero.
- **Bajas**: consiste en eliminar del fichero un registro existente. La eliminación puede ser lógica, cambiando el valor de algún campo del registro que usemos para controlar dicha situación; o bien, física, eliminando físicamente el registro del fichero. El borrado físico consiste muchas veces en reescribir de nuevo el fichero en otro fichero sin los datos que se desean eliminar y luego renombrarlo al fichero original.

- **Modificaciones**: consiste en cambiar parte del contenido de un registro. Antes de realizar la modificación será necesario localizar el registro a modificar dentro del fichero; y una vez localizado se realizarán los cambios y se reescribe el registro.
- **Consultas**: consiste en buscar en el fichero un registro determinado.

Operaciones sobre Ficheros Secuenciales

En los registros secuenciales los registros se insertan en orden cronológico, es decir, un registro se inserta a continuación del último insertado. Si hay que añadir nuevos registros estos se añaden a partir del final del fichero.

Veamos cómo se realizan las operaciones típicas:

- **Consultas**: para consultar un determinado registro es necesario empezar la lectura desde el primer registro, y continuar leyendo secuencialmente hasta localizar el registro buscado. Por ejemplo, si el registro a buscar es el 90 dentro del fichero, será necesario leer secuencialmente los 89 anteriores.
- **Altas**: en un fichero secuencial las altas se realizan al final del último registro insertado, es decir, solo se permite añadir datos al final del fichero.
- **Bajas**: para dar de baja un registro de un fichero secuencial es necesario leer todos los registros uno a uno y escribirlos en un fichero auxiliar, salvo el que deseamos dar de baja. Una vez reescritos hay que borrar el fichero inicial y renombrar el fichero auxiliar dándole el nombre del fichero original.
- **Modificaciones**: consiste en localizar el registro a modificar, efectuar la modificación y reescribir el fichero inicial en otro fichero auxiliar que incluya el registro modificado. El proceso es similar a las bajas.

Los ficheros secuenciales se usan típicamente en aplicaciones de proceso por lotes, como por ejemplo copias de seguridad o backup, o si se procesan todos los registros. La ventaja de estos ficheros es el acceso a los registros es rápida cuando se procesan todos uno detrás del otros.

Las desventajas son que no permite el acceso aleatorio es decir, que no se puede acceder directamente a un registro determinado; hay que leer todos los anteriores. Otra desventaja es que para actualizar los daros hay que hacer una copia de todo el fichero.

Operaciones sobre Ficheros Aleatorios

Las operaciones en ficheros aleatorios tienen la particularidad que para acceder a un registro hay que localizar la posición o dirección donde se encuentra. Los ficheros de acceso directo en disco manipulan direcciones relativas en lugar de direcciones absolutas (número de pista y número de sector en el disco), lo que hace al programa independiente de la dirección absoluta del fichero del disco.

Normalmente para posicionarse en un registro es necesario aplicar una función de conversión que usualmente tiene que ver con el tamaño del registro y con la clave del mismo (la clave es el campo o campos que identifican de forma unívoca a un registro). Por ejemplo, disponemos de un fichero de empleados con 3 campos: identificador, apellido y salario. Usamos el identificador como campo clave y le damos el valor 1 para el primer empleado, 2 para el segundo y así sucesivamente; para localizar al empleado con identificador X necesitamos acceder a la posición tamaño * (X-1) para acceder a los datos de dicho empleado.

Puede ocurrir que al aplicar la función al campo clave, nos devuelva una posición ocupada por otro registro, en ese caso, habría que buscar una nueva posición libre en el fichero para ubicar dicho registro o utilizar una zona de excedentes dentro del mismo para ir ubicando estos registros.

Veamos cómo se realizan las operaciones típicas:

- **Consultas**: para consultar un determinado registro necesitamos saber su clave, aplicar la función de conversión a la clave para obtener la dirección y leer el registro ubicado en esa posición. Habría que comprobar si el registro buscado está en esta posición, si no está se buscará en la zona de excedentes.
- **Altas**: para insertar un registro necesitamos saber su clave, aplicar la función de conversión a la clave para obtener la dirección y escribir el registro en la posición devuelta. Si la posición está ocupada por otro registro se insertará en la zona de excedentes.
- **Bajas**: las bajas suelen realizarse de forma lógica, es decir, se suele utilizar un campo del registro a modo switch que tenga el valor 1 cuando el registro exista y le damos el valor 0 para darlo de baja, físicamente el registro no desaparece del disco. Habría que localizar el registro a dar de baja a partir del campo clave y reescribir en este campo el valor 0.
- **Modificaciones**: para modificar un registro hay que localizarlo, necesitamos saber su clave para aplicar la función de conversión y así obtener la dirección, modificar los datos que nos interesen y reescribir el registro en esa posición.

Una de las principales ventajas de los fichero aleatorios es el rápido acceso a una posición determinada para leer o escribir un registro. El gran inconveniente es establecer la relación entre la posición que ocupa el registro y su contenido; ya que a veces al aplicar la función de conversión para obtener la posición se obtienen claves ocupadas y hay que recurrir a la zona de excedentes. Otro inconveniente es que se puede desaprovechar parte del espacio destinado al fichero, ya que se pueden producir huecos (posiciones no ocupadas) entre un registro y otro.