## Conocimiento: Archivos en Java

Normalmente los programas interactúan con información que es entregada por un cliente, pudiendo ser el cliente un usuario, otro sistema o una consulta remota.

Java posee clases que permiten interactuar con la consola para el ingreso de información, estas clases están en el paquete java.io. En especial veremos la clase BufferedReader que permite leer una línea en formato String.

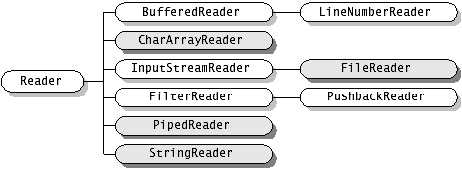
La clase BufferedReader nos provee el comportamiento base para modelar la entrada de datos. Cuando queremos leer de distintas entradas como el teclado o un archivo necesitamos utilizar o crear otros objetos intermedios (como el System.in y el FileInputStream). Además la lectura del conjunto de caracteres se puede hacer de una manera directa a través del método readLine(), y para leer otros tipos de datos necesitamos realizar pasos adicionales. Por ejemplo para leer un dato del tipo int necesitamos leer una fila usando readLine() y después convertirla a numero utilizando Integer.parseInt()

Un Stream es simplemente una fuente o destino de bytes. Los streams más comunes son los archivos. También pueden ser caracteres de un String o bloque de memoria. Java permite que el programador los trate de la misma forma que archivos.

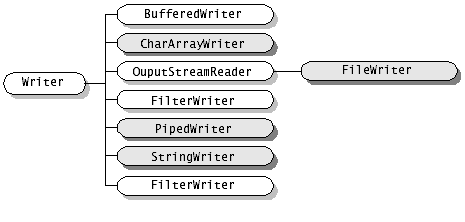
En Java se usan diferentes mecanismos para dar formato a datos. Ofrece clases orientada a manejo de bajo nivel y otras de alto nivel. Da gran flexibilidad y hace uso de “filtros en cascada” para manipular los datos de entrada o salida.

A continuación se muestra la categoría de clases con su árbol jerárquico:

* Clases para lectura de caracteres:

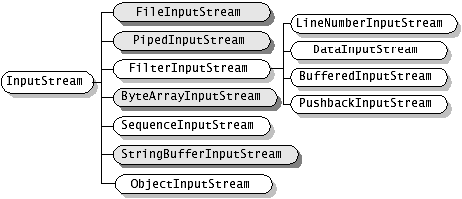


* Clases para escritura de caracteres

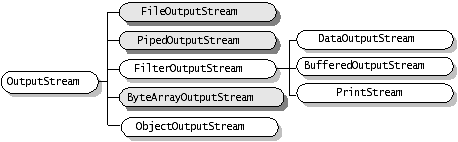
****

PrintWriter

* Lectura de bytes

****

* Escritura de bytes



La clase OutputStream es una clase abstracta (no instanciable) que especifica un número de operaciones para escribir un byte, para escribir un arreglo de bytes, y para cerrar el stream.

Clases derivadas de ésta son FileOutputStream y ByteArrayOutputStream, las cuales son clases concretas (no abstractas). Sus operaciones de write envían bytes a archivo o memoria. Para dar formato a los datos (imprimir números y Strings) se dispone de las clases “Writers”.

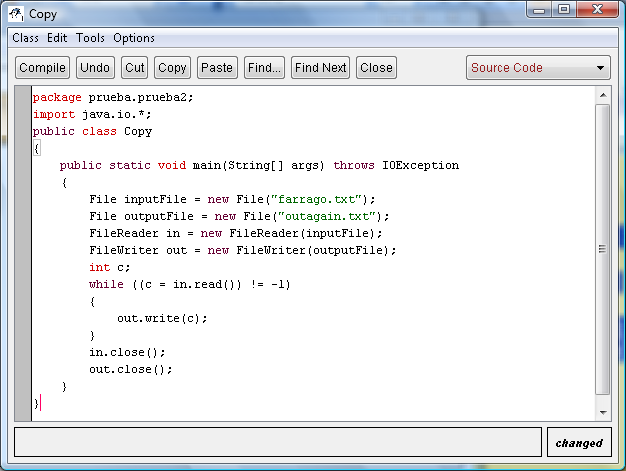
PrintWriter es una clase de esa jerarquía, la cual recibe en su constructor el objeto con el cual podrá escribir (write) los bytes.

Ejemplo sintaxis de uso:

FileOutputStream fout = new FileOutputStream (“output. dat”);

PrintWriter pout = new PrintWiter(fout);

Ejemplo: Crear una clase que copie un archivo en otro.



Además de formato podemos disponer de un buffer (o canal) para mejorar el desempeño de algunos dispositivos de entrada y salida.

Para poder crear una cadena dentro de un Buffer se usa:

PrintWriter pout = new PrintWriter(new BufferedOutputStream  
(new FileOutpurStream(“Output.dat”)));

PrintWriter

BufferedOutputStream

FileOutputStream

Los filtros se pueden construir usando cualquier combinación de clases encadenando una después de la otra.

La primera clase es aquella que permite leer o escribir objetos, y la última clase de la cadena envía o recibe los bytes. Las clases intermedias pueden hacer el trabajo que necesitemos (buffer, encriptación etc.)

Para salida binaria, usamos las clases **DataInputStream** y **DataOutputStream**. Ellas proveen las siguientes operaciones:

|  |  |
| --- | --- |
| **DataInputStream** | **DataOutputStream** |
| readInt | writeInt |
| readShort | writeShort |
| readLong | writeLong |
| readFloat | writeFloat |
| readDouble | writeDouble |
| readBoolean | writeBoolean |
| readChar | writeChar |
|  | writeChars |

* Clase BufferReader

Esta clase mejora el rendimiento dotando la entrada de un buffer o canal. Tiene dos constructores:

BufferedReader (Reader flujoEntrada)

BufferedReader (Reader flujoEntrada, int tamañoBuffer)

Para cerrar el buffer se utiliza:

void close()

Para marcar la ubicación dentro del buffer se utiliza:

void mark()

Para leer información se utilizan:

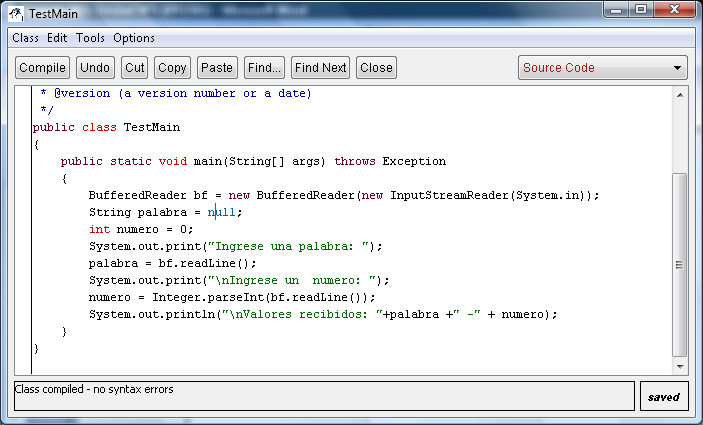
int read()

String readLine()

Para saltar un determinado número de caracteres se utiliza:

long skip(long n)

Por ejemplo: hacer un programa que pida por consola un String y un número entero, luego los imprima en la misma línea:



El resultado de la ejecución del método main es:

