Entrada salida.

Manejo de ficheros

E/S basada en flujos

Un flujo es una abstracción que produce o consume información.

Los programas de java realizan E/S a través de flujos.

Todos los flujos se comportan igual, aunque los dispositivos físicos que los sustentan sean diferentes.

Por tanto se pueden aplicar los mismos métodos y clases para cualquier dispositivo.

Ej: Escribir en consola es igual que escribir en un disco.

El paquete java.io

En este paquete están las clases que trabajan el flujo de datos.

Hay dos tipos de flujos de datos

- Bytes. Procesan flujos de bytes, por ejemplo para leer datos binarios.
- Caracteres: Permiten procesar la E/S de caracteres. Usan Unicode (internacionalización)

Esta distinción hace que el sistema de E/S sea mayor, ya que hay dos jerarquías de clases.

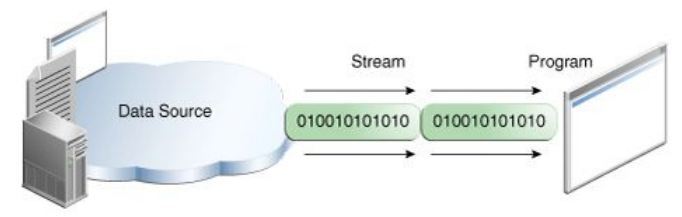
En el nivel inferior, toda la E/S sigue orientada a bytes.

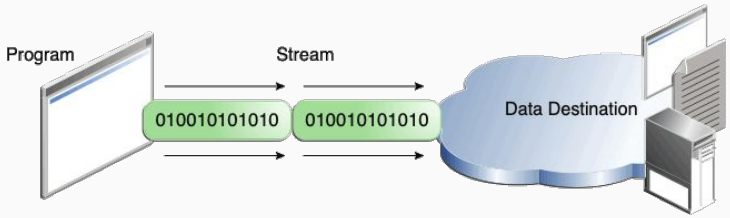
Ejemplo

```
import java.io.*;
public class Ejemplo1 {
	public static void main(String[] args) {
		//Cuenta caracteres introducidos por la entrada estándar (teclado)
		int contador = 0;
		try {
			while(System.in.read()!='\n')
				contador++;
		} catch (IOException e) {
			e.printStackTrace();
		}
		System.out.println();
		System.out.println("Tecleados " + contador + " caracteres.");
	}
}
```

Streams

Un stream representa una fuente de entrada o un destino de salida.





Tipos de streams

Representación de la información
- Flujos de bytes
- Flujos de caracteres

Propósito
- Entrada (InputStream, Reader)
- Salida (OutputStream, Writer)

Cubren el acceso secuencial, (el aleatorio sería con RandomAccessFile)

Tipo de operación

- Transferencia de datos
- Transformación de datos (realizan algún tipo de procesamiento de los datos (buffering, conversiones, filtrados)

Si estamos haciendo E/S basada en texto se utilizan los streams de caracteres. Se denominan lectores (readers) y writers.

Si estamos haciendo E/S basada en datos se utilizan streams de bytes. Se denominan streams de entrada (input stream) y de salida (output stream)

Clases de flujo de entrada

Lectura de datos de una fuente de entrada (archivo , servicio, memoria…)

- Flujo de bytes: **InputStream**, BufferedInputStream, DataInputStream, FileInputStream

- Flujo de caracteres: **Reader**, BufferedReader, FileReader

Clases de flujo de salida

Homólogas a las de entrada hacia dispositivos de salida

- Flujo de bytes: **OutputStream**, PrintStream, BufferedOutputStream, DataOutputStream y FileOutputStream

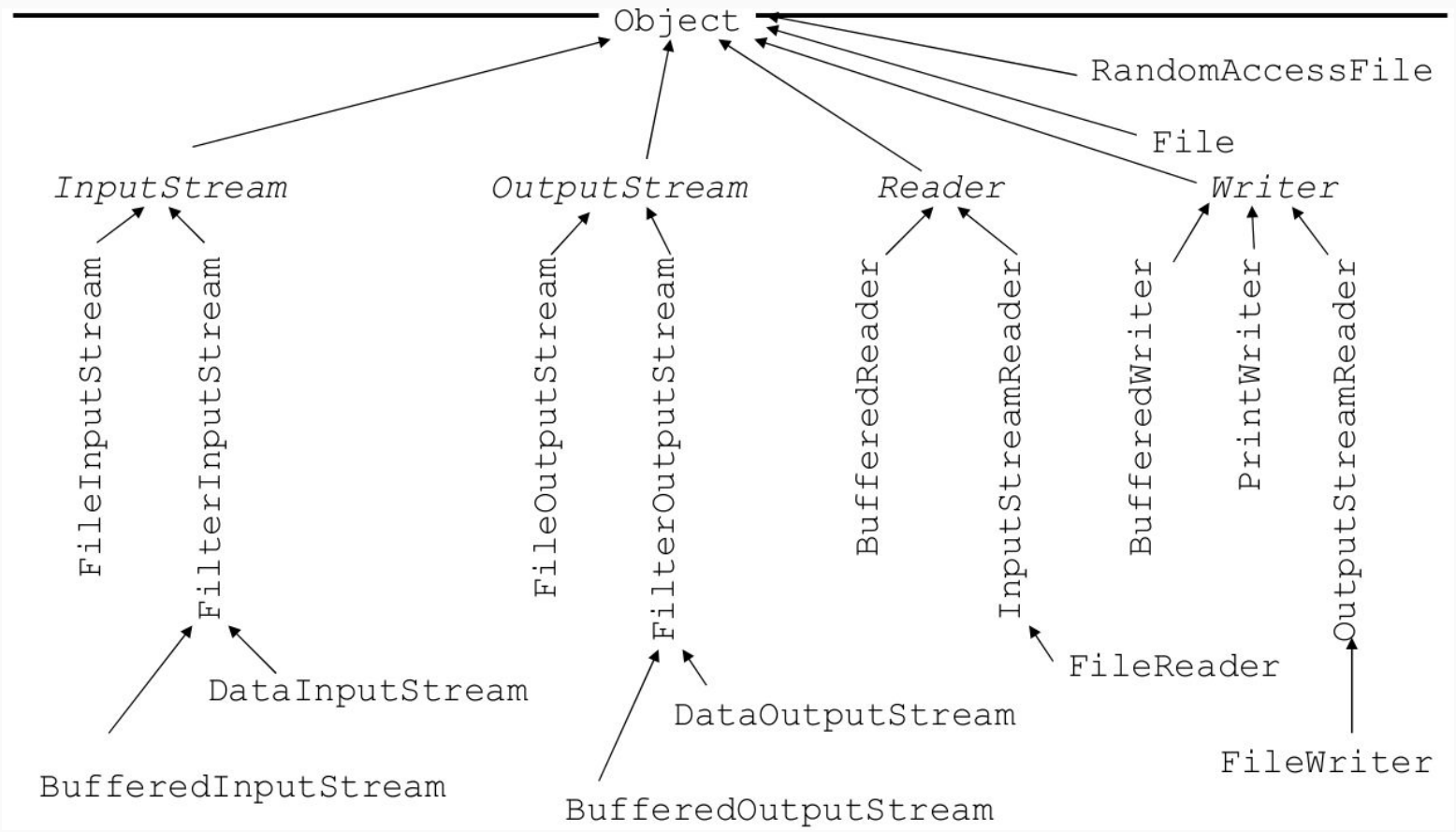
- Flujo de caracteres: **Writer**, PrintWriter, BufferedWriter, FileWriter

Clases de archivo

File

RandomAccessFile

Jerarquía de las clases en java.io



Métodos de InputStream

int read()	Lee el siguiente byte del flujo de entrada y lo devuelve como un **entero**. Cuando alcanza el final del flujo de datos devuelve -1 (**EOF**)
int read (byte b[ ])	Lee hasta b.length bytes y los almacena en la matriz b. Devuelve el número de bytes leídos o -1 cuando alcanza el final del flujo de datos.
int read (byte b[ ], int off, int cantidad)	Lee hasta “cantidad” de bytes de datos del flujo de entrada, empezando desde la posición indicada “off”, y los almacena en la matriz b. Devuelve el número de bytes leído
available()	Devuelve el número de bytes que se pueden leer de un flujo de entrada sin que se produzca un bloqueo por causa de una llamada a otro método que utiliza el mismo flujo de entrada.
skip (long n)	Omite la lectura de n bytes de datos de un flujo de entrada y los descarta
close()	Cierra el flujo de entrada y libera los recursos del sistema utilizados por el flujo de datos.

OutputStream

void write (int b)	Escribe b en un flujo de datos de salida.
void write (byte b[ ])	Escribe b.length bytes de la matriz b en un flujo de datos de salida.
void write (byte b[ ], int off, int cantidad)	Escribe “cantidad” de bytes de la matriz b en el flujo de datos de salida, empezando en la posición dada por el desplazamiento “off”
flush()	Vacía el flujo de datos y fuerza la salida de cualquier dato almacenado en el búfer.
close()	Cierra el flujo de datos de salida y libera cualquier recurso del sistema asociado con él.

Streams sobre ficheros

FileInputStream: Como InputStream pero especializada en leer ficheros

- FileInputStream(String nombre)
- FileInputStream(File nombre)

FileOutputStream: Como OutputStream pero especializada en escribir en archivos.

- FileOutputStream(String nombre)
- FileOutputStream(String nombre, boolean append) //append == true
- FileOutputStream(File nombre)

Esquema de trabajo con streams

Entrada de datos (leer datos)	Salida de datos (escribir datos)
1. Se crea un objeto flujo de datos de lectura 2. Se leen datos de él con los métodos apropiados 3. Se cierra el flujo de datos	1. Se crea un objeto flujo de datos de escritura 2. Se escriben datos de él con los métodos apropiados 3. Se cierra el flujo de datos