Ficheros

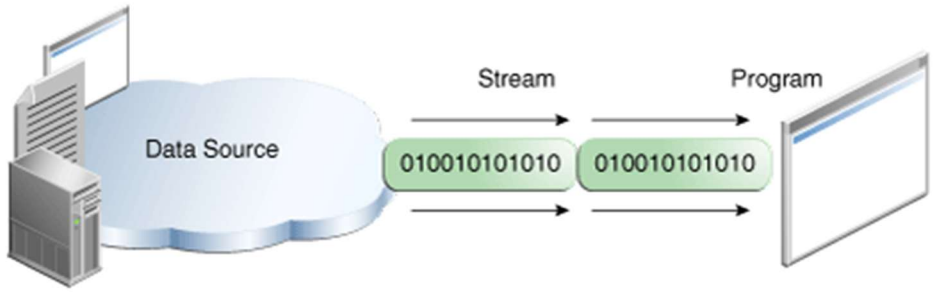
Vamos a utilizar dos tipos de ficheros, ficheros de texto y ficheros binarios pero antes para ello debemos de conocer algún concepto previo.

Flujos de E / S

Un *flujo de E / S* representa una fuente de entrada o un destino de salida. Una secuencia puede representar muchos tipos diferentes de fuentes y destinos, incluidos archivos de disco, dispositivos, otros programas y matrices de memoria, esto último se refiere a una tabla de datos directamente en la memoria RAM.

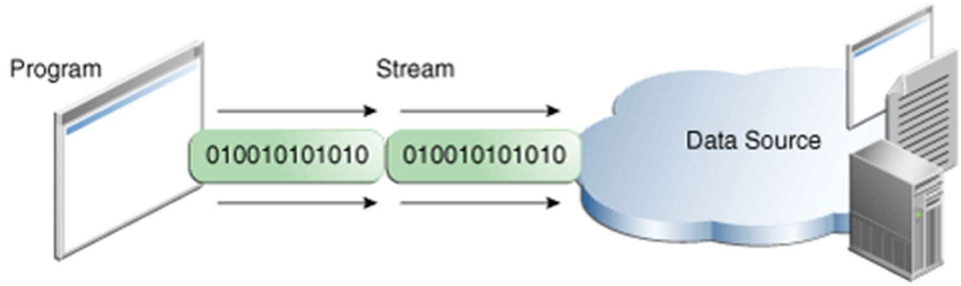
Los flujos admiten **muchos tipos** diferentes de datos, incluidos bytes simples, tipos de datos primitivos, caracteres localizados y objetos o varibles complejas. Algunas secuencias simplemente transmiten datos; otros manipulan y transforman los datos de manera útil.

Independientemente de cómo funcionen internamente, todos los flujos presentan el mismo modelo simple para los programas que los utilizan: un flujo es una secuencia de datos. Un programa usa un flujo de entrada para leer datos de una fuente, un elemento a la vez:



Leer información en un programa.

Un programa usa un flujo de salida para escribir datos en un destino, un elemento a la vez:



Escribir información de un programa.

.

La fuente de datos y el destino de los datos que se muestran arriba pueden ser cualquier cosa que contenga, genere o consuma datos.

Secuencias almacenadas en búfer

Cada solicitud de lectura o escritura es manejada directamente por el sistema operativo. Esto puede hacer que un programa sea mucho menos eficiente, ya que cada una de estas solicitudes a menudo desencadena el acceso al disco, la actividad de la red o alguna otra operación que es relativamente costosa en tiempo

Para reducir este tipo de sobrecarga, la plataforma Java implementa flujos de E/S en búfer. Los flujos de entrada almacenados en búfer leen datos de un área de memoria conocida como búfer; la API de entrada nativa se llama solo cuando el búfer está vacío. De manera similar, los flujos de salida almacenados en búfer escriben datos en un búfer y la API de salida nativa se llama solo cuando el búfer está lleno.

Un programa puede convertir una secuencia sin búfer en una secuencia con búfer usando una envoltura.

```
inputStream = new BufferedReader (nuevo FileReader ("xanadu.txt"));
outputStream = new BufferedWriter (nuevo FileWriter ("characteroutput.txt"));
```

Hay cuatro clases para almacenar en búfer que se utilizan para envolver secuencias sin búfer:

- BufferedInputStream y BufferedOutputStreamcrear secuencias de bytes.
- BufferedReader y BufferedWriter para crear secuencias de caracteres almacenadas en búfer.