# T3.4. Recordatorio Flujos de Java

# Recordatorio de los flujos en Java

## InputStreams y OutputStreams

Manejan bytes a secas. Por ejemplo, si queremos leer un fichero byte a byte usaremos `FileInputStream` y si queremos escribir usaremos `FileOutputStream`.

Son operaciones a muy bajo nivel que usaremos muy pocas veces (por ejemplo, solo si quisiéramos cambiar el primer byte de un archivo). En general usaremos otras clases más cómodas de usar.

## Readers y Writers

En lugar de manejar *bytes* manejan *caracteres* (recordemos que hoy en día y con Unicode una letra como la *ñ* en realidad podría ocupar más de un byte).

Así, cuando queramos leer letras de un archivo usaremos clases como `FileReader` y `FileWriter`.

Las clases `Readers` y `Writers` en realidad se apoyan sobre las `InputStreams` y `OutputStreams`.

A veces nos interesará mezclar conceptos y por ejemplo poder tener una clase que use caracteres cuando a lo mejor Java nos ha dado una clase que usa bytes. Así, por ejemplo `InputStreamReader` puede coger un objeto que lea bytes y nos devolverá caracteres. De la misma forma `OutputStreamWriter` coge letras y devuelve los bytes que la componen.

## `BufferedReaders` y `PrintWriters`

Cuando trabajamos con caracteres (que recordemos pueden tener varios bytes) normalmente no trabajamos de uno en uno. Es más frecuente usar **líneas** que se leen y escriben de una sola vez. Así por ejemplo, la clase `PrintWriter` tiene un método `print(ln)` que puede imprimir elementos complejos como `floats` o cadenas largas.

Además, Java ofrece clases que gestionan automáticamente los *buffers* por nosotros lo que nos da más comodidad y eficiencia. Por ello es muy habitual hacer cosas como esta:

```
lectorEficiente = new
        BufferedReader(new FileReader("fich1.txt"));
escritorEficiente = new
        BufferedWriter(new FileWriter("fich2.txt"));
```

En el primer caso creamos un objeto `FileReader` que es capaz de leer caracteres de `fich1.txt`. Como esto nos parece poco práctico creamos otro objeto a partir del primero de tipo `BufferedReader` que nos permitirá leer bloques enteros de texto.

De hecho, si se comprueba la ayuda de la clase `FileReader` se verá que solo hay un método `read` que devuelve un `int`, es decir el siguiente **carácter** disponible, lo que hace que el método sea muy incómodo. Sin embargo `BufferedReader` nos resuelve esta incomodidad permitiéndonos trabajar con líneas.