1.- Introducción.

Se llama a los datos que se guardan en ficheros **datos persistentes**, porque persisten más allá de la ejecución de la aplicación que los trata. Los ordenadores almacenan los ficheros en unidades de almacenamiento secundario como discos duros, discos ópticos, etc. En esta unidad veremos, entre otras cosas, cómo hacer con Java las operaciones de crear, actualizar y procesar ficheros.

A las operaciones, que constituyen un flujo de información del programa con el exterior, se les conoce como Entrada/Salida (E/S).

Las operaciones de E/S en Java las proporciona el paquete estándar de la API de Java denominado **java.io** que incorpora ***interfaces*** *(1)*, clases y excepciones para acceder a todo tipo de ficheros.

La **librería java.io** contiene las clases necesarias para gestionar las operaciones de entrada y salida con Java. Estas clases de E/S las podemos agrupar fundamentalmente en:

* Clases para leer entradas desde un flujo de datos.
* Clases para escribir entradas a un flujo de datos.
* Clases para operar con ficheros en el sistema de ficheros local.
* Clases para gestionar la serialización de objetos.

**java.nio (Non- Blocking I/O)** fue introducido en el API de Java desde la versión 1.4 como extensión eficiente a los paquetes **java.io** y**java.net**. Java NIO ofrece una forma diferente de trabajar con IO que las API de IO estándar. Se basa en el “***Buffe*** *(2)****r***” y el “***Channel*** *(3)*”.

**java.io vs java.nio**

* Depende de lo que necesitemos pueden ser complementarios y utilizaremos conjuntamente ambos paquetes.
* **java.nio** permite manejar múltiples canales (archivos o conexiones de red) con uno o unos pocos hilos.
* En **java.nio** el procesamiento de datos es más complicado que usar los streams bloqueantes de java.io
* **java.nio** es la opción si necesito manejar cientos de conexiones (canales) abiertas y en cada una manejar una pequeña cantidad de datos.
* java.io es la opción si voy a manejar pocas conexiones con un alto ancho de banda (envío mucha información a la vez)
* Hasta JSE7 **java.io.File** era la clase utilizada para realizar operaciones I/O con archivos
* Esta clase tenía limitaciones como:
  + Muchos **métodos** no lanzaban excepciones por lo que era muy complicado detectar y resolver errores
  + No permitía manejar enlaces simbólicos
  + Presenta problemas de escalabilidad. Directorios largos provocaban problemas de memoria e incluso de denegación de servicio
* El paquete **java.nio.file** incorporado a partir de JSE7 resuelve estos problemas.
* Es el que debemos usar para trabajar con archivos independientemente de si realizamos I/O con streams (**java.io**) ó con buffers y channels (**java.nio**)

1. Interfaces: Colección de métodos abstractos y propiedades en las que se especifica qué se debe hacer pero no su implementación. Serán las clases que implementen estas interfaces las que describan la lógica del comportamiento de los métodos.
2. Buffer: Ubicación de la memoria de un computador reservada para el almacenamiento temporal de información, mientras que está esperando ser procesada, para evitar que el programa o recurso que los requiere se quede en algún momento sin datos.
3. Channel: Vía por donde se transfiere la comunicación
4. Métodos: Es una subrutina o subprograma asociada a una clase o a un objeto.