MANEJO DE FICHEROS INTRODUCCIÓN

Javier García-Retamero Redondo

TRABAJANDO CON FICHEROS

INTRODUCCIÓN

INTRODUCCIÓN

- Un ordenador utiliza ficheros para guardar datos. Estos datos pueden ser canciones en mp3, películas en formato mp4, etc., están almacenados en ficheros, los cuales están grabados en un soporte como son los discos duros, pendrives, etc.
- Dos conceptos importantes:
  - **Datos persistentes**: Datos que se guardan en ficheros y persisten más allá de la ejecución de la aplicación que los trata. Si cerramos la aplicación los datos permanecen
  - **Operaciones de Entrada/Salida (E/S)**: Operaciones, que constituyen un flujo de información del programa con el exterior.

INTRODUCCIÓN

- IO
- NIO
- jdk 1.4
- NIO.2
- jdk 1.7

- Las operaciones de E/S las proporciona el paquete estándar de la API de Java denominado **java.io** que incorpora interfaces, clases y excepciones para acceder a todo tipo de ficheros.
- Estas clases de E/S las podemos agrupar fundamentalmente en:
  - Clases para operar con ficheros en el sistema de ficheros local.
  - Clases para leer entradas desde un flujo de datos.
  - Clases para escribir entradas a un flujo de datos.
  - Clases para gestionar la serialización de objetos.

PROBLEMAS IO

- IO
- NIO
- jdk 1.4
- NIO.2
- jdk 1.7

- A la clase **File** le faltan algunas funcionalidades importantes, como un método de copia.
- También define muchos métodos que devuelven boolean. En caso de error, se devuelve false, en lugar de lanzar una excepción. El desarrollador no tiene forma de saber por qué falla.
- No proporciona un buen manejo en el soporte de enlaces simbólicos.
- Proporciona un conjunto limitado de atributos de archivo.
- Presenta problemas de escalabilidad. Directorios largos provocan problemas de memoria e incluso de denegación de servicio

INTRODUCCIÓN

- IO
- NIO
- jdk 1.4
- NIO.2
- jdk 1.7

- **java.nio** (Non- Blocking I/O) 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 "Buffer" y no en los flujos.
- Dependiendo de lo que necesitemos puede ser complementario a **java.io** y utilizaremos conjuntamente ambos paquetes.

INTRODUCCIÓN

- IO
- NIO
- jdk 1.4
- NIO.2
- jdk 1.7

- Permite manejar múltiples canales (archivos o conexiones de red) con uno o unos pocos hilos.
- El procesamiento de datos es más complicado que usar los streams bloqueantes de java.io
- 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).

INTRODUCCIÓN

- IO
- NIO
- jdk 1.4
- NIO.2
- jdk 1.7

- Los componentes principales incluyen:
  - **Buffers**: Almacenamiento temporal de datos durante operaciones de lectura y escritura.
  - **Canales**: Representa conexiones abiertas a dispositivos de E/S, como archivos y sockets.
  - **Selectores**: Permite que un solo hilo monitoree múltiples canales para verificar la preparación de E/S.

INTRODUCCIÓN

- IO
- NIO
- jdk 1.4
- NIO.2
- jdk 1.7

- **NIO.2** fue Introducido en el API de Java en la **versión 1.7** con mejoras y como ampliación al paquete NIO.. Ofrece, entre otras:
  - **Acceso mejorado** al sistema de archivos.
  - Operaciones de **E/S asincrónicas**.
  - Capacidad para **trabajar con enlaces del sistema** de archivos y metadatos.