2.1. Manejo de ficheros

Acceso a Datos

Alejandro Roig Aguilar
alejandro.roig@iesalvarofalomir.org

IES Álvaro Falomir
Curso 2023-2024

Objetivo de la unidad y criterios de evaluación

RA1. Desarrolla aplicaciones que gestionan información almacenada en ficheros identificando el campo de aplicación de los mismos y utilizando clases específicas.

a) Se han utilizado clases para la gestión de ficheros y directorios.

b) Se han valorado las ventajas y los inconvenientes de las distintas formas de acceso.

c) Se han utilizado clases para recuperar información almacenada en ficheros.

d) Se han utilizado clases para almacenar información en ficheros.

e) Se han utilizado clases para realizar conversiones entre diferentes formatos de ficheros.

f) Se han previsto y gestionado las excepciones.

g) Se han probado y documentado las aplicaciones desarrolladas.

Ficheros

Un fichero es la estructura de almacenamiento más elemental.

- Consta de una secuencia de bytes
- Tiene un nombre
- Está situado en un directorio en una jerarquía de directorios en un sistema de ficheros.

Persistencia de datos en ficheros

Actualmente se utilizan en aplicaciones para guardar información simple como un fichero de configuración o un fichero log.

También en sistemas de correo electrónico, para intercambio de datos entre sistemas, (como los estándares XML, CSV y JSON que estudiaremos) o para copias de seguridad.

Además, muchísimas aplicaciones almacenan los datos con los que trabajan en ficheros con infinidad de formatos diferentes.

Tipos de ficheros según su contenido

Según la información que contienen se diferencian en:

- Ficheros de texto: contienen únicamente una secuencia de caracteres codificados (UTF-8, ASCII, ISO-8859...). Su contenido se puede visualizar y modificar con cualquier editor de texto.
- Ficheros binarios: son el resto de los ficheros. Pueden contener cualquier tipo de información (texto, imágenes, vídeos, ficheros...). En general, requiere de programas especiales para mostrar la información que contienen. Los programas también se almacenan en ficheros binarios.

Tipos de ficheros según su contenido

Tipo de fichero	Extensión	Descripción
Ficheros de texto	.txt	Fichero de texto plano
	.xml	Fichero XML
	.json	Fichero de intercambio de información
	.props	Fichero de propiedades
	.conf	Fichero de configuración
	.sql	Script SQL
Ficheros binarios	.jpg	Fichero de imagen
	.doc, .docx	Fichero de Microsoft Word
	.avi	Fichero de vídeo
	.ppt, .pptx	Fichero de Microsoft PowerPoint
	.bin, .dat	Ficheros específicos de aplicación
	.pdf	Fichero PDF

Operaciones en el sistema de ficheros

Java proporciona clases para gestionar los sistemas de ficheros y realizar operaciones básicas como:

- Comprobar si un fichero existe o no, si es regular o un directorio
- Copiar, mover y borrar ficheros
- Moverse a través del sistema de ficheros

Aunque es probable que conozcáis la librería java.io, desde su introducción en Java 7 se recomienda usar la librería NIO2, o java.nio, con las clases Path, Paths y Files.

Estas clases nos permitirán usar la API de Streams para leer y escribir ficheros.

Clases Path y Paths

Un objeto de clase Path contiene un nombre de fichero y su ruta en el sistema de ficheros. Ofrece métodos para operaciones como:

- startsWith, endsWith: comprueba si la ruta actual empieza o acaba con una subruta dada.
- getParent: obtiene el directorio padre de la ruta actual.
- getRoot: obtiene el directorio raíz de la ruta actual.
- iterator: para explorar cada directorio y subdirectorio de la ruta.
- toAbsolutePath: para obtener la ruta absoluta de la ruta.

La clase Paths es otra clase que contiene métodos estáticos para tratar con rutas.

Clase Files

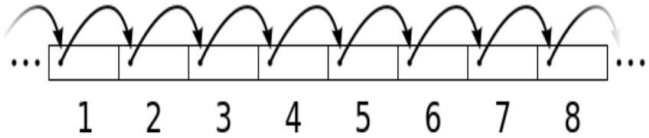
La clase Files contiene métodos estáticos para la operación sobre ficheros y directorios. Además de métodos que veremos para leer y escribir en ficheros, pero también otros muy interesantes:

- copy (Path, Path): copia un archivo (solo archivos, no directorios)
- move(Path, Path): mueve un archivo (solo archivos, no directorios)
- delete(Path): elimina un archivo (no directorio)
- createFile(Path): crea un nuevo fichero especificado por Path
- createDirectory(Path): crea un nuevo directorio especificado por Path
- exists(Path): comprueba que una ruta exista
- isRegularFile(Path): comprueba si el archivo es regular

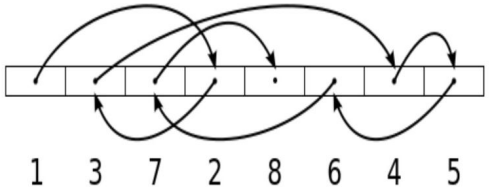
Tipos de ficheros según su forma de acceso

Existen dos formas de acceder a los contenidos de un fichero:

- Acceso secuencial: se accede comenzando desde el principio del fichero. Para llegar a cualquier parte del fichero, hay que pasar antes por todos los contenidos anteriores.



- Acceso aleatorio: se accede directamente a los datos situados en cualquier posición del fichero.



Lectura de ficheros

Los métodos más comunes para leer ficheros son:

- Files.lines, devuelve un Stream<String>. Desde Java 8, funciona bien en ficheros grandes y pequeños. Autocierre de los recursos.
- Files.readString, devuelve un String. Desde Java 11 y para ficheros pequeños (< 2 GB).
- Files.readAllLines, devuelve un List<String>. Desde Java 8 y para ficheros pequeños (< 2 GB).
- Files.readAllBytes, devuelve un byte[]. Desde Java 7.
- newBufferedReader, devuelve un BufferedReader, de java.io, que utiliza buffers para operaciones E/S más eficientes.

Lectura de ficheros

Como resumen, a la hora de leer texto de un fichero, no hay mucha diferencia entre los métodos al leer archivos pequeños, solo el tipo de retorno.

Para leer un archivo grande, lo más eficiente es usar Files.readAllBytes, aunque también podemos usar Files.lines, para obtener un Stream y/u operar de forma paralela, o el clásico BufferedReader a través de Files.NewBufferedReader.

Si queremos leer bytes de un fichero, lo más cómodo será usar el método específico Files.readAllBytes

Otra opción es usar Files.newInputStream, que devuelve un InputStream del que se puede leer byte a byte.

Escritura de ficheros

Los métodos más comunes para escribir ficheros son:

- Files.writeString: Desde Java 11, escribe caracteres tal cual. No quita nada ni añade nada. UTF-8 por defecto
- Files.write: Desde Java 7. Similar al anterior, este método escribe tanto caracteres como datos binarios.
- newBufferedWriter, devuelve un BufferedWriter, que hace más eficiente la escritura de grandes cantidades de información.

Para escribir a final de fichero, las clases Files.writeString y Files.write añaden el argumento StandardOpenOption.APPEND.

Lectura y escritura con ficheros binarios

Además de los métodos descritos, hay otras clases que permiten la lectura y escritura de datos binarios.

- InputStream y OutputStream son clases tradicionales de Java para realizar operaciones de E/S de datos binarios y se pueden combinar con Files.newInputStream y Files.newOutputStream de Java NIO2 para trabajar con archivos binarios de manera eficiente.
- FileChannel proporciona una interfaz de bajo nivel para la lectura y escritura de datos en ficheros. FileChannel trabaja con ByteBuffer, lo que lo hace adecuado tanto para datos de texto como para datos binarios.

Serialización de objetos

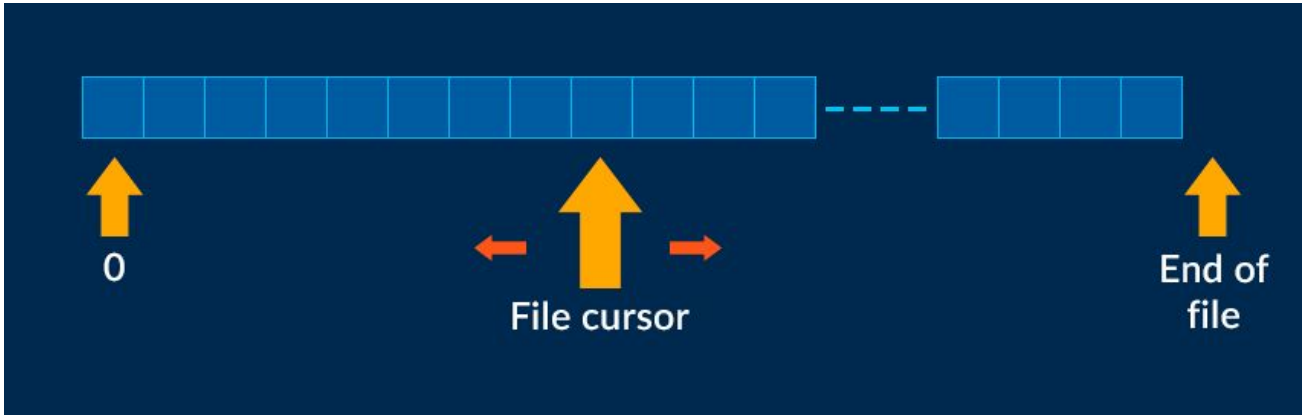
Hemos visto cómo escribir bytes y tipos de datos primitivos en un archivo binario, ... pero ¿cómo guardar un objeto en un fichero?

- La clase debe implementar la interfaz Serializable.
- La clase ObjectInputStream lee objetos (deserializa) del fichero con el método readObject, mientras que ObjectOutputStream los escribe (serializa) en él con el método writeObject.
- Se recomienda declarar la constante serialVersionUID de tipo long para asegurar la compatibilidad de los objetos en distintos sistemas.

La serialización de objetos tiene importantes alternativas como los formatos JSON o XML para el almacenamiento de datos más interoperables y seguros en algunos casos.

Acceso aleatorio

Existen situaciones donde necesitaremos acceder a un byte o grupo de bytes concretos para comprobar su valor o modificarlo. En vez de ir byte a byte secuencialmente hasta llegar a la posición deseada, podemos utilizar ficheros de acceso aleatorio.



Acceso aleatorio

La clase RandomAccessFile de Java.io permite leer y/o escribir un fichero binario y moverse hasta una posición dada para leer/modificar la información almacenada. Para esto, los métodos más útiles son:

- read, readInt, readFloat: lee información de un tipo básico dado
- write, writeInt, writeFloat: escribe información de un tipo básico dado
- getFilePointer(): devuelve la posición del puntero
- seek(long pos): establece la posición del puntero
- length(): devuelve el tamaño del fichero en bytes
- skipBytes(int salto): desplaza el puntero *salto* posiciones