AAD

Acceso a Datos
UD02T01 – Ficheros

IES Ciudad Escolar
Desarrollo de Aplicaciones Multiplataforma

Índice

- Ficheros y tipos.
- Formas de acceso a un fichero.
- Clases asociadas a las operaciones de gestión de ficheros (secuenciales, aleatorios) y directorios: creación, borrado, copia, movimiento, entre otras.

Ficheros

- Un **fichero** es el método de almacenamiento de información más elemental.

- Todos los mecanismos de almacenamiento, por complejos que sean, terminan de algún modo almacenando los datos en ficheros (bases de datos, xmls, csv, json, …)

- Los ficheros se identifican generalmente por un nombre, una extensión y una ubicación (jerarquía de directorios donde se localiza)

  Ejemplo: /home/jsala/agenda.dat

- Atendiendo al contenido de los ficheros, estos pueden ser:
  - **Ficheros de texto**: contienen exclusivamente caracteres (letras, números, signos de puntuación, espacios, tabuladores, saltos de línea, etc.) utilizando una determinada codificación (ASCII) que permite representar el texto en una secuencia de bytes. El contenido de dichos ficheros es interpretable por cualquier sistema operativo.

  - **Ficheros binarios**: contienen cualquier tipo de información (imágenes, videos, audios, hojas de cálculo, documentos de Word, pdfs, ejecutables binarios, etc). Cualquier fichero no considerado de texto plano, es binario. El contenido solo es interpretable mediante programas especiales.

Clase java.io.File

- La clase **File** de Java permite trabajar con ficheros obteniendo información relativa a directorios y ficheros dentro de un filesystem (sistema de ficheros) y realizar operaciones sobre ellos tales como:

  - Creación de ficheros y directorios.
  - Borrado.
  - Renombrado.
  - Evaluar si existe.
  - Evaluar si se trata de un directorio o un fichero.
  - Obtener el directorio padre.
  - Listar el contenido de un directorio.
  - Obtener el tamaño de un fichero.
  - Obtener el nombre del fichero o directorio.
  - Etc.

Formas de acceso a ficheros

- Existen dos formas de acceso al contenido de los **ficheros:**
  - **Acceso secuencial:** Se accede siempre comenzando desde el principio del fichero independientemente de la información que estemos buscando dentro de él.

    Las operaciones más comunes en ficheros de acceso secuencial son:

    - Creación de un fichero o apertura para escribir datos.
    - Leer datos del fichero.
    - Borrar información de un fichero.
    - Copiar datos de un fichero a otro.
    - Búsqueda de información en un fichero.
    - Cerrar un fichero.
  - **Acceso aleatorio:** Se accede directamente a los datos situados en cualquier posición de un fichero. Esto implica que el fichero debe estar disponible en su totalidad al momento de ser accedido, algo que no siempre es posible. Los ficheros de acceso directo generalmente tiene sus registros de un tamaño fijo o preestablecido.

    Las operaciones más comunes en ficheros de acceso aleatorio son:

    - Creación de un fichero o apertura para leer o escribir, o hacer ambas cosas sobre datos.
    - Escribir datos en fichero.
    - Leer datos del fichero.
    - No es posible borrar o añadir información intermedia de un fichero (sobrescribiría).
    - Posicionarnos en una dirección concreta del fichero.
    - Cerrar un fichero.

Nota: Para poder realizar el acceso aleatorio, el soporte físico en el que esté almacenado el fichero, también debe habilitarlo. Por ejemplo, las cintas magnéticas no permiten el acceso secuencial en los ficheros.

Acceso aleatorio: Java.io.RandomAccessFile

A diferencia de los ficheros con acceso secuencial, los ficheros de acceso aleatorio NO son streams (flujos). La información se almacena en forma de bytes (array de bytes). Se dispone de un puntero de lectura y escritura que indica la posición a partir de la cual se leerá o escribirá en el fichero.

La clase **RandomAccessFile** facilita el acceso aleatorio a ficheros mediante los siguientes métodos:

- **long getFilePointer()** Devuelve la posición actual del puntero del fichero. Indica la posición (en bytes) donde se va a leer o escribir.
- **long length()** Devuelve la longitud del fichero en bytes.
- **void seek(long pos)** Coloca el puntero del fichero en una posición *pos* determinada. La posición se da como un desplazamiento en bytes desde el comienzo del fichero. La posición 0 indica el principio del fichero. La posición *length()* indica el final del fichero.

Además dispone de métodos de lectura/escritura:

- **public int read()** Devuelve el byte leído en la posición marcada por el puntero. Devuelve -1 si alcanza el final del fichero. Se debe utilizar este método para leer los caracteres de un fichero de texto.
- **public final String readLine()** Devuelve la cadena de caracteres que se lee, desde la posición marcada por el puntero, hasta el siguiente salto de línea que se encuentre.
- **public xxx readXxx()** Hay un método *read* para cada tipo de dato básico: *readChar, readInt, readDouble*, etc.
- **public void write(int b)** Escribe en el fichero el byte indicado por parámetro. Se debe utilizar este método para escribir caracteres en un fichero de texto.
- **public final void writeBytes(String s)** Escribe en el fichero la cadena de caracteres indicada por parámetro.
- **public final void writeXxx(argumento)** También existe un método *write* para cada tipo de dato básico: *writeChar, writeInt, writeDouble*, etc.

Acceso aleatorio: Java.io.RandomAccessFile

Como determinar la longitud de un dato usando la clase RandomAccessFile.

Para poder determinar cuanto ocupa cada registro en un archivo de acceso aleatorio, debemos tener en cuenta los bytes de cada uno de los datos. De esta forma, dependiendo del tipo de dato que queramos almacenar, cada registro ocupará un tamaño u otro. A continuación, se muestra el tamaño en bytes que ocupa tipo de dato.

TIPO DE DATO	TAMAÑO EN BYTES
Char	2 bytes
Byte	1 byte
Short	2 bytes
Int	4 bytes
Long	8 bytes
Float	4 bytes
Double	8 bytes
Boolean	1 byte
Espacio en blanco (corresponde a un char)	1 byte
Salto de línea (corresponde a un byte	1 byte
String	2 byte por cada char

Tamaño de los tipos de datos en java para archivos de acceso aleatorio

Hay que tener en cuenta que los datos tipo String (tipo texto), en java son considerados como objetos. Por este motivo, un dato tipo String, se considera un array de caracteres tipo char. Esto quiere decir que la información de tipo String, ocupará dos bytes por cada carácter tipo char. A esta información, se le deben sumar los bytes ocupados por los espacios en blanco y los saltos de línea.

Uso de objetos en ficheros de acceso aleatorio.

Para poder almacenar objetos en archivos aleatorios con la clase RandomAccessFile, es necesario convertirlos a un array de bytes es decir serializarlos (interfaz serializable) y para poder leer un fichero que contiene objetos creados en java, debemos deserializar dicho objeto.