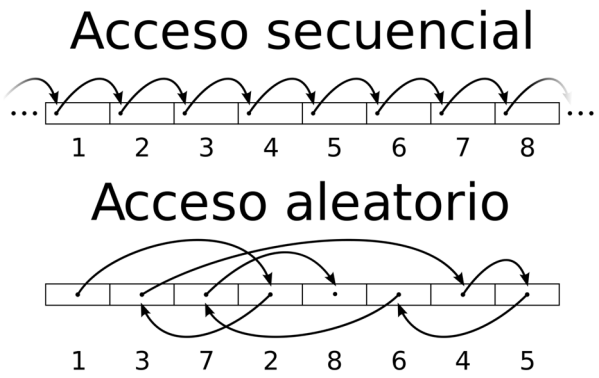
UA01

4. Formas de acceso a un fichero

Hemos visto que en Java puedes utilizar dos tipos de ficheros (de texto o binarios). Veremos a continuación que existen dos tipos de acceso a ellos: **secuencial** o **aleatorio**.



- **Acceso aleatorio:** los archivos de acceso aleatorio, al igual que lo que sucede usualmente con la memoria (RAM = Random Access Memory), permiten acceder a los datos en forma no secuencial, desordenada. Esto implica que el archivo debe estar disponible en su totalidad al momento de ser accedido, algo que no siempre es posible.
- **Acceso secuencial:** En este caso los datos se leen de manera secuencial, desde el comienzo del archivo hasta el final (el cual muchas veces no se conoce a priori). Este es el caso de la lectura del teclado o la escritura en una consola de texto, no se sabe cuándo el operador terminará de escribir.

Operaciones básicas sobre ficheros de acceso secuencial

Como operaciones más comunes en ficheros de acceso secuencial, tenemos el acceso para:

- Crear un fichero o abrirlo para grabar 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.

Ejemplo de lectura secuencial:

LeerFichSecuencial.java

Este ejemplo lee un texto de dos en dos caracteres, de forma secuencial, y los almacena en un array que muestra, finalmente, en líneas diferentes.

Operaciones básicas sobre ficheros de acceso aleatorio

A menudo, no necesitas leer un fichero de principio a fin, sino simplemente acceder al fichero como si fuera una base de datos, donde se salta de un registro a otro; cada uno en diferentes partes del fichero. Java proporciona una clase **RandomAccessFile** para este tipo de entrada/salida.

Esta clase:

- Permite leer y escribir sobre el fichero, no es necesario dos clases diferentes.
- Necesita que le especifiquemos el modo de acceso al construir un objeto de esta clase: sólo lectura o bien lectura y escritura.
- Posee métodos específicos de desplazamiento como seek(long posicion) o skipBytes(int desplazamiento) para poder movernos de un registro a otro del fichero, o posicionarnos directamente en una posición concreta del fichero.

Por esas características que presenta la clase, un archivo de acceso directo tiene sus registros de un tamaño fijo o predeterminado de antemano.

La clase posee dos constructores:

- `RandomAccessFile(File file, String mode).`
- `RandomAccessFile(String name, String mode).`

En el primer caso se pasa un objeto File como primer parámetro, mientras que en el segundo caso es un **String**. El modo es: "**r**" si se abre en modo **lectura** o "**rw**" si se abre en modo **lectura y escritura.**

El ejemplo que se muestra a continuación inserta datos de empleados en un fichero aleatorio. Por cada empleado también se insertará un identificador que coincidirá con el índice +1 con el que se recorren los arrays. La longitud del registro de cada empleado es la misma (36 bytes) y los tipos que se insertan y su tamaño en bytes es el siguiente:

- Se inserta en primer lugar un entero, que es el identificador, ocupa 4 bytes.
- A continuación una cadena de 10 caracteres, es el apellido, cada carácter Unicode ocupa 2 bytes, por tanto el apellido ocupa 20 bytes.
- Un tipo entero que es el departamento, ocupa 4 bytes.
- Un tipo Double que es el salario, ocupa 8 bytes.

Tamaño de otros tipos: short (2 bytes), byte (1 byte), long (8 bytes), boolean (1bit), float (4 bytes), etc.
El fichero se abre en modo "rw" para lectura y escritura:

EscribirFichAleatorio.java

El siguiente ejemplo toma el fichero anterior y visualiza todos los registros. El posicionamiento para empezar a recorrer los registros empieza en 0, para recuperar los siguientes registros hay que sumar 36 (tamaño del registro) a la variable utilizada para el posicionamiento:

LeerFichAleatorio.java

Para consultar un empleado determinado no es necesario recorrer todos los registros del fichero, conociendo su identificador podemos acceder a la posición que ocupa dentro del mismo y obtener sus datos

Para comprobar estos ejemplos:

Conversor de punto flotante <https://gregstoll.com/~gregstoll/floattohex/>

Conversor de Byte a String <https://codebeautify.org/byte-to-string>