Ficheros Acceso Aleatorio

Supongo que a estas alturas te habrás dado cuenta de que, para buscar un elemento dentro un fichero, tienes que ir leyendo de forma secuencial hasta que das con él. Esto es manejable cuando tienes ficheros pequeños, pero se vuelve ineficiente cuando tienes ficheros enormes de miles de entradas.

Evidentemente, este problema ya lo han tenido cientos de programadores desde hace décadas, así que obviamente, ya existe una solución. El acceso aleatorio. Que viene a significar 'acceder al punto que quieras del fichero'.

Aprovechamos para que entendáis lo **importante** que es saber manejarse con estos conceptos. Los habéis visto por encima en Programación de 1ro, cuando os enseñaron lo que son las Pilas, las Colas, FIFO, LIFO… Es importante entender las características de estas estructuras para luego sacar provecho de ellas. No todas permiten hacerlo todo, ni son igual de eficientes haciendo una cosa que otra. Por ejemplo:

- Lista: las listas en java (como el ArrayList) son secuencias de objetos. Puedes acceder a cualquier elemento de la secuencia libremente, y añadirlo o eliminarlo de cualquier punto. Permiten acceso aleatorio.

- Colas: como tal no existen en Java, pero sí en otros lenguajes. Sólo permiten añadir elementos al final y recuperarlos del principio, luego no permiten acceso aleatorio. Son estructuras FIFO.

- Pilas: como tal no existen en Java, pero sí en otros lenguajes. Sólo permiten añadir elementos al principio y recuperarlos del principio, luego no permiten acceso aleatorio. Son estructuras LIFO

Adicionalmente, Java dispone de otras estructuras que deberíais revisar:

- Interfaz List: Java dispone de un varias de clases que implementan List, de las cuales ArrayList es solamente una. En PSP de 2ndo se enseñan varias de ellas.

- Hashmap: en Java, permiten almacenar objetos de forma clave-valor. Son muy eficientes a la hora de acceder de forma aleatoria mediante la clave, pero muy ineficientes si pretendes recorrer el Hashmap como si fuese una lista.

Entre otras. Recomendable que les deis un vistazo por vuestra cuenta. Os ayudará a manejaros bien con el acceso a ficheros.

Fichero acceso aleatorio - escritura

Disponemos de la clase **RandomAccessFile** para acceder a cualquier punto dentro de un fichero. Esto es, a cualquier byte del fichero. Aunque la clase permita hacer esto, luego tú puedes optar por trabajar con el fichero como prefieras. Puedes ir leyendo secuencialmente el fichero o acceder a una posición en concreto. Eso ya depende de tus necesidades.

Como nota interesante, al declarar un RandomAccessFile es necesario indicar el **modo de acceso** al fichero: sólo lectura 'r' o lectura y escritura 'rw'.

Los métodos de escritura principales son diferentes versiones de **write ()**.

A modo de ejemplo.

```
String apellido[] = { "FERNANDEZ", "GIL", "LOPEZ" };
Double salario[] = { 1000.45, 2400.60, 3000.0 };

public void writeFile() {
    File file = null;
    RandomAccessFile randomAccessFile = null;
    StringBuffer buffer = null;
    try {
        // Create a File
        file = new File("example.txt");

        // Create a
        randomAccessFile = new RandomAccessFile(file, "rw");

        // Let's write the file...
        int n = apellido.length;
        for (int i = 0; i < n; i++) {
            // We use this as a fake ID
            randomAccessFile.writeInt(i + 1);

            // 10 Characters
            buffer = new StringBuffer(apellido[i]);
            buffer.setLength(10);
            randomAccessFile.writeChars(buffer.toString());

            // The money
            randomAccessFile.writeDouble(salario[i]);
        }
    } catch (Exception e) {
        System.out.println("An error occurred.");
    } finally {
        // Close the RandomAccessFile
        try {
            if (randomAccessFile != null)
                randomAccessFile.close();
        } catch (IOException e) {
            // Nothing to do here...
        }
    }
}
```

En el ejemplo, cada 'registro' que escribimos tiene un id, un texto y un Double. Date cuenta de que lo que estamos haciendo es escribir información de forma secuencial en forma de conjuntos de bytes (posiciones), no existen objetos como tales aquí.

Como ya hemos dicho, con RandomAccessFile podemos hacer un acceso aleatorio a cualquier byte del fichero. Para hacer eso, sabemos que el tamaño de cada 'registro' que hemos escrito es 32 bytes:

4 bytes (id) + (10x 2) bytes (chars) + 8 bytes (Double) = **32 bytes**

Luego podemos escribir un nuevo registro al final del fichero simplemente haciendo:

```
public void writeFileEnd() {
    File file = null;
    RandomAccessFile randomAccessFile = null;
    StringBuffer buffer = null;
    try {
        // Create a File
        file = new File("example.txt");

        // Create a
        randomAccessFile = new RandomAccessFile(file, "rw");

        // Let's write the file...
        int id=4;
        long posicion = (id - 1) * 32;
        randomAccessFile.seek(posicion);

        // We use this as a fake ID
        randomAccessFile.writeInt(id);

        // 10 Characters
        buffer = new StringBuffer("GONZALEZ");
        buffer.setLength(10);
        randomAccessFile.writeChars(buffer.toString());

        // The money
        randomAccessFile.writeDouble(1230.87);

    } catch (Exception e) {
        System.out.println("An error occurred.");
    } finally {
        // Close the RandomAccessFile
        try {
            if (randomAccessFile != null)
                randomAccessFile.close();
        } catch (IOException e) {
            // Nothing to do here...
        }
    }
}
```

Pero esto no nos impide que, conociendo el orden en el que vamos escribiendo las cosas, accedamos (contando bytes) a la posición que deseemos del fichero para escribir o sobrescribir lo que necesitemos.

Fichero acceso aleatorio - lectura

De forma similar, podemos realizar una lectura tanto secuencial como aleatoria del contenido de los ficheros.

A modo de ejemplo, la lectura secuencial:

```
public void readFile() {
    File file = null;
    RandomAccessFile randomAccessFile = null;
    try {
        file = new File("example.txt");
        randomAccessFile = new RandomAccessFile(file, "r");

        // Loop to read a bytes until the end of file
        int posicion = 0;
        for (;;) {
            randomAccessFile.seek(posicion);
            // Fake ID
            int id = randomAccessFile.readInt();

            // 10 Characters
            char surname[] = new char [10];
            for (int i = 0; i < surname.length; i++) {
                char aux = randomAccessFile.readChar();
                surname[i] = aux;
            }
            String apellidos = new String(surname);

            // The money
            Double salario = randomAccessFile.readDouble();

            System.out.printf("ID: %s, Apellido: %s, Salario: %.2f %n",
                    id,  apellidos.trim(), salario);

            // Next position
            posicion= posicion + 32;

            // End of file?
            if (randomAccessFile.getFilePointer() == randomAccessFile.length())
                break;
        }
    } catch (Exception e) {
        System.out.println("An error occurred.");
    } finally {
        // Close the reader
        try {
            if (randomAccessFile != null)
                randomAccessFile.close();
        } catch (IOException e) {
            // Nothing to do here...
        }
    }
}
```

Fíjate en que, de nuevo estamos **contando bytes manualmente** para acceder a la posición que nos interesa en cada momento.

Por tanto, la lectura aleatoria es algo razonablemente simple de implementar:

```
public void readFileEnd(int registro) {
    File file = null;
    RandomAccessFile randomAccessFile = null;
    try {
        file = new File("example.txt");
        randomAccessFile = new RandomAccessFile(file, "r");

        // Loop to read a bytes until the end of file
        int posicion = (registro - 1) * 32;

        if (posicion >= file.length()) {
            System.out.printf("Registro %d no existe", registro);
        } else {
            randomAccessFile.seek(posicion);
            // Fake ID
            int id = randomAccessFile.readInt();

            // 10 Characters
            char surname[] = new char[10];
            for (int i = 0; i < surname.length; i++) {
                char aux = randomAccessFile.readChar();
                surname[i] = aux;
            }
            String apellidos = new String(surname);

            // The money
            Double salario = randomAccessFile.readDouble();

            System.out.printf("ID: %s, Apellido: %s, Salario: %.2f %n", id,
                apellidos.trim(), salario);
        }
    } catch (Exception e) {
        System.out.println("An error occurred.");
    } finally {
        // Close the reader
        try {
            if (randomAccessFile != null)
                randomAccessFile.close();
        } catch (IOException e) {
            // Nothing to do here...
        }
    }
}
```

Práctica - 3

Crea un sistema visual que permita trabajar con los Productos de un almacén. De cada Producto tenemos que saber:

- Id: entero
- Nombre: String tamaño 10
- Empresa: String tamaño 10
- Precio unitario: entero
- Unidades en almacén: Double

Al arrancar el programa, se muestra un JTable con todos los Productos del fichero. Un botón de filtrado nos permitirá buscar un producto por nombre. Adicionalmente, debemos de poder:

- Añadir un producto al final.
- Borrar cualquier producto.
- Editar cualquier producto.