TEORÍA DE LA GESTIÓN DE LA INFORMACIÓN BASES DE DATOS

Introducción a las estructuras de almacenamiento secundario

- El API de Java(librería de Clases).
  - Organizada en Paquetes, documentada en el API de Java y puede consultarse en
    - http://docs.oracle.com/javase/6/docs/api/index.html
  - Importar clases y paquetes de Java
    - #import nombrePaquete.NombreClase; // para importar solo una clase del paquete
    - #import nombrePaquete.*; // para importar todas las clases de ese paquete

- Excepciones
  - Java implementa el mecanismo de excepciones para facilitar la separación del código que implementa la lógica del programa en ausencia de errores del código que controla los posibles errores de ejecución Importar clases y paquetes de Java
  - En aquellas secciones de código en las que se prevea que puede llegar a producirse una excepción deberá utilizarse el bloque try/catch:

```
Try{  // segmento de código que se ejecuta en ausencia de errores
   }
catch (TipoDeLaExcepcion1 variable)
  {          // segmento de código que se ejecutará si en el bloque try
             //se produce la excepción deltipo TipoDeLaExcepcion1
   }
catch (TipoDeLaExcepcion2 variable)
  { // segmento de código que se ejecutará si en el bloque try se
   // produce la excepción del tipo TipoDeLaExcepcion2
  }....
```

Excepciones

- Si en el método en el que capturamos la excepción no queremos definir el tratamiento específico de esa excepción, deberemos lanzarla hacia el método que llamó a este que ha generado la excepción. Para ello bastará con definir en la cabecera del método:

```
public void nombreMetodo(argumentos….)  throws TipoDeLaExcepcion1, tipoDeLaExcepcion2, …
{     // cuerpo del método. No es necesario bloque try/catch
}
```

- En este caso no es necesario utilizar el bloque try/catch ya que si se produce alguna de las excepciones definidas en la cabecera del método, la máquina virtual de Java se encarga de pasarle la excepción al método que llamó a éste.

- Excepciones:

```
public LectorArchivoRandom(String nombreArchivo)
{   try
      { this.archivo = new RandomAccessFile( nombreArchivo, "rw" );       }
    catch (FileNotFoundException fnfe)
        { System.out.println("nombre del archivo incorrecto");
          System.exit(1);
        }
 }
```

- Excepciones:

```
public LectorArchivoRandom(String nombreArchivo)
{   try
     { this.archivo = new RandomAccessFile( nombreArchivo, "rw" );          }
   catch (FileNotFoundException fnfe)
         { System.out.println("nombre del archivo incorrecto");
           System.exit(1);
         }
  }
```

- Las excepciones en Java también son clases y están dentro de los paquetes del API. La excepción FileNotFoundException está en el paquete java.io (hay que importarla)
-

Ficheros en Java

- Los ficheros en Java se manejan a través de distintas clases incluidas en el paquete java.io del API de Java.
- La única clase del paquete java.io que permite trabajar con ficheros de acceso aleatorio (una vez abierto/creado el fichero podemos cambiar el puntero de L/E hacia delante/atrás según se precise) es la clase RandomAccessFile.
- En Java para crear/abrir un fichero con acceso aleatorio lo haremos creando un objeto de la clase RandomAccessFile (ver API). Concretamente usaremos el constructor al que se le pasa como parámetro el nombre del fichero y el modo de apertura del mismo ("r" para lectura y "rw" para lectura/escritura).
  - Una vez hecho esto se puede leer(read), escribir(write), desplazar el puntero de lectura mediante seek, obtener el tamaño del fichero (lenght) y por ultimo cerrarlo(close).

Clase LectorArchivoRandom y PruebasLectorArchivoRandom.

- La clase LectorArchivoRandom:
  - no es una aplicación ya que no tiene el método main.
  - Dispone de una propiedad (privada) de tipo RandomAccessFile. Esta propiedad mantendrá el enlace al fichero en disco (a través del SO). Esta propiedad es necesaria, ya que los métodos de creación, generación de contenido, lectura de contenido y cierre del fichero son distintos (si todo se realizase en un único método esta propiedad no sería necesaria).
  - El único constructor de esta clase crea nuevo el fichero cuyo nombre se indica como parámetro (si el fichero existe lo sobreescribe). Como constructor inicializa las propiedades del objeto (en este caso sólo 1).
  - El tratamiento de excepciones se realiza en esta clase (no se lanzan hacia fuera)
  - El método generaContenido genera el contenido del fichero. Este consta de una estructura lógica con dos registros y 5 campos cada uno (dos int, dos cadenas de caracteres de 25 y 30 char cada una y un boolean).
  - El método volcar posiciona el puntero de L/E al comienzo del fichero y lee el contenido del mismo (sigue el mismo formato que la escritura).
  - El método cerrar cierra el fichero. Después de cerrar el fichero cualquier intento de lectura/escritura sobre el mismo lanzará la excepción IOException
  - Dos método privados que permiten escribir o leer una cadena de caracteres de una determinada longitud sobre un fichero RandomAccessFile.

Clase LectorArchivoRandom y PruebasLectorArchivoRandom.

- La clase PruebasLectorArchivoRandom :
  - Es una aplicación ya que tiene el método main.
  - Crea un objeto LectorArchivoRandom, genera su contenido, lo vuelca por pantalla y cierra el fichero asociado a este objeto.

- Hacer:
  - Ejecutar el programa. Intentar abrir el fichero con un editor de texto. ¿cuál es el contenido?¿está mal?
  - Cambiar el código para que todo el tratamiento de excepciones (sacar error por pantalla y terminar la aplicación) se realice en PruebasLectorArchivoRandom en vez de en LectorArchivoRandom

Ejemplo de Apertura de un fichero gestionando las excepciones

```
try  {// ponemos el puntero de L/E al comienzo del fichero
        this.archivo.seek(0);
       // la lectura del fichero debe respetar el formato y el orden de tipos  al generar el    contenido
       for (int i=0; i<2; i++)
        {  c1 = this.archivo.readInt();
           c2 = this.archivo.readInt();
           c3 = this.leerCadena(25, this.archivo);
           c4 = this.leerCadena(30, this.archivo);
           c5 = this.archivo.readBoolean();
           System.out.println("R" + i + "-> c1: " + c1 + "; c2: " + c2 + "; c3: " + c3 + "; c4: " + c4 + "; c5: " + c5);
         }
        }
 catch (IOException ioe)
  {System.out.println("error en la lectura del fichero");
   System.exit(3);
  }
```