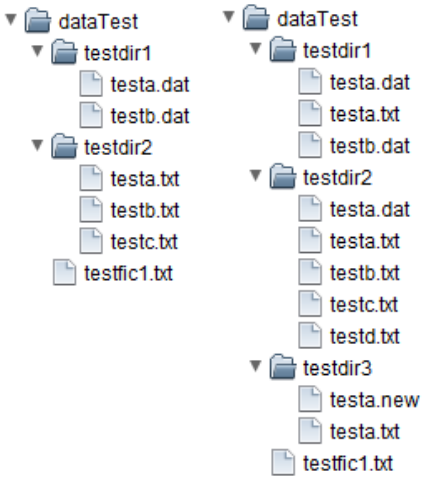
Examen final Prog. III - Junio 2017

- Sólo se permite la conexión a internet durante una hora. Aprovecha ese tiempo para hacer todas las búsquedas que necesites. **A partir de ese momento debes desconectar wifi/red**.
- En la web de la asignatura ("Entregables y Evaluación") tienes los ficheros con los que se va a trabajar en un fichero comprimido. Crea un proyecto en eclipse y descomprime ese fichero en su carpeta src.
- Para entregar el examen, comprime todo el directorio src, y renombra el fichero (.zip o .rar) con tu dni, nombre y apellido. Asegúrate que tiene los ficheros que has modificado. Entrega por ALUD.

1. Planteamiento

Partimos para este examen de una utilidad que conocerás, `DiskUseTool.java`. Permite revisar una carpeta de disco y analizar los ficheros y espacio contenidos en cada una de sus subcarpetas, con un árbol (JTree) visual. Para las pruebas de las diversas mejoras se suministra también una clase (*DiskUseTest*) que crea un par de estructuras de ejemplo de carpetas y ficheros, con los que puedes probar la utilidad (los verás al ejecutar *DiskUseTool* en la lista de selección como Test 1 y Test 2). A la derecha puedes ver los árboles que se crean en cada caso (test 1 a la izquierda, test 2 a la derecha), siempre bajo la carpeta "dataTest" que se creará en el directorio en curso (la carpeta del proyecto de Eclipse).
Se pide que mejores este proyecto con las siguientes tareas (puntuación sobre 10 entre corchetes).



2. Tareas

Tarea 1 [2,5]. JUnit + Recursividad. *DiskUseToolTest.java*. Unas 40 líneas de código.
Completa la prueba de unidad de la clase *DiskUseToolTest* para que se pruebe lo siguiente:

- Se crean los ficheros del test 1
- Se lanza la herramienta sobre la carpeta dataTest (lanzarla de forma no interactiva)
- Se comprueba que el árbol visualizado en la herramienta es correcto para los ficheros del test 1. Para ello se deberá empezar en la carpeta dataTest y en el nodo inicial del árbol (*NodoConTamanyo*) al que puedes acceder desde la herramienta lanzada (`.getVentana().getRaiz().getFirstChild()`) y recursivamente:
  - Los paths absolutos deben ser iguales
  - Cada carpeta debe tener un nodo equivalente
  - El número de ficheros que hay en la carpeta debe ser igual al que contiene el nodo
- Se libera la ventana.

Hacer lo mismo con el test 2.

Observa que hay dos estructuras recursivas que recorrer a la vez. Una es la que debe dirigir el proceso (los ficheros físicos del disco), la otra (los nodos) la deberás ir consultando a la par para comprobar que los datos son correctos. De los nodos te pueden ser útiles los métodos *getChildCount()*, *getChildAt()* y *getDatos()*.

Tarea 2 [1,5]. BD. *DiskUseTool.java*. Unas 15 líneas de código.
Tienes una clase BD.java ya preparada para gestionar dos tablas con sqlite, estos son los diseños:

```
Tabla fichero
----------------------
pathAbs string PRIMARY KEY // Path absoluto
nombre string  // Nombre de fichero
tamanyo bigint // Tamaño del fichero en bytes
fecha bigint   // Fecha (long) de ult.modif.
ultRevision bigint // Fecha (long) ult.revisión
```

```
Tabla historiaCambios
-------------------------
fichero_pathAbs string    // Foreign key

tamanyo bigint       // Tamaño anterior fich (bytes)
fecha bigint         // Fecha anterior de ult.modif.
ultRevision bigint   // Fecha de ult.revisión
```

Modifica el programa para que según se vayan procesando los ficheros (método *procesaFichero*), se vayan creando filas en la tabla fichero con cada fichero nuevo (entendiendo el path absoluto como la clave única de cada fichero), y si el fichero ya existía en la tabla, se pasan los datos anteriores a la tabla de historia de cambios y se

modifican los datos dejando los más actuales en la tabla de ficheros (puedes chequear este proceso lanzando el test 1 y luego el test 2, que añade cinco ficheros y modifica *testdir2\testb.txt*).

Tarea 3 [1,5]. JC. *DiskUseTool.java*. Unas 15 líneas de código.
A la vez que recorremos un árbol de directorios, queremos saber si hay ficheros repetidos en carpetas diferentes (una cosa habitual en discos cuando se van utilizando un tiempo). Modifica el proceso para que según se recorran los ficheros se vayan metiendo todos (solo los ficheros, no carpetas) en una estructura de datos, y al finalizar el recorrido se utilice esa estructura para mostrar en el área de texto inferior qué ficheros aparecen repetidos (entendiendo repetido un fichero cuando tiene el **mismo nombre, longitud y fecha**). Puedes utilizar la estructura que desees, pero se valorará que utilices la estructura más eficiente para el tiempo de proceso, teniendo también en cuenta que se quiere obtener el listado por fichero, en orden alfabético de su nombre, y que se deben visualizar el tamaño del fichero y las carpetas en las cuales se encuentra. Por ejemplo para el test 2 deberá obtenerse:

```
Ficheros iguales:
testa.dat     5453010        dataTest\testdir1      dataTest\testdir2
testa.txt     527            dataTest\testdir1      dataTest\testdir2      dataTest\testdir3
```

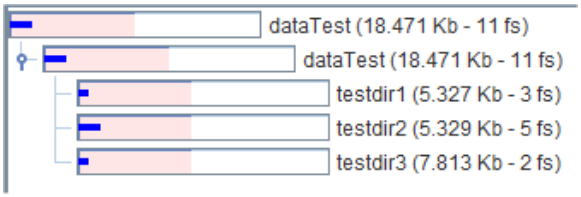
Tarea 4 [1]. Patrones. *DiskUseTool.java*. Unas 12 líneas de código.
Implementa el patrón *observer* para modificar el recorrido del árbol que hace la utilidad, de manera que se permita observar cada vez que se pasa por un fichero. Usa ese patrón, una vez implementado, para sacar en el área de texto inferior un listado de los ficheros que son mayores o iguales a 5 Mb. Por ejemplo en el test 2:

```
Ficheros mayores de 5 Mb:
D:\desarrollo\ProgIII\Examenes\dataTest\testdir1\testa.dat = 5.325 Kb
D:\desarrollo\ProgIII\Examenes\dataTest\testdir2\testa.dat = 5.325 Kb
D:\desarrollo\ProgIII\Examenes\dataTest\testdir3\testa.new = 7.812 Kb
```

Si usas nomenclatura funcional de Java 8, esta tarea se valora 0,5 puntos adicionales.

Tarea 5 [1]. Hilos. *DiskUseTool.java y VentanaDiskUseTool.java*. Unas 8 líneas de código.
Lanza la utilidad sobre una carpeta grande (c:\ por ejemplo). Observa que si cierras la ventana, la aplicación no se para. Es porque el recorrido se sigue haciendo aunque la ventana ya no esté abierta. Cambia este comportamiento incorporando una gestión adecuada de atributo de finalización, de modo que al cerrarse la ventana, esta informe al recorrido para que en cuanto pueda se detenga (haz una parada "educada").

Tarea 6 [1,5]. Swing – componente. *VentanaDiskUseTool.java y JProgressBarLin.java*. Unas 25 líneas de código.
La progressbar del JTree da una información visual del tamaño que ocupan los ficheros de la carpeta, pero el número de ficheros solo aparece en texto. Crea un nuevo componente *JProgressBarLin* que herede de JProgressBar y que permita la gestión de un doble valor, el normal y uno adicional que represente al número de ficheros. Este adicional se debe pintar como una línea azul estrecha encima de la rosada actual. El valor puedes asignarlo como el logaritmo del número de ficheros, en una escala de 0 a 20.



Tarea 7 [1,5]. Swing – renderer. *VentanaDiskUseTool.java*. Unas 10 líneas de código.
Puedes observar que al seleccionar cualquier carpeta en el árbol, sus ficheros y subcarpetas aparecen en la tabla derecha de la ventana. Pero en esta tabla no se diferencian fácil los ficheros de las carpetas (solo porque el valor de la columna 1 es vacío para las carpetas). Modifica el *renderer* de la tabla para que, cuando sean datos de carpetas, se visualice el nombre en fondo amarillo, por ejemplo así:

Tarea 8 [0,5]. Preguntas (Android).
Contesta a las preguntas de Android que encontrarás en el fichero Android.txt en el paquete de fuentes del examen. Asegúrate de que ese fichero se guarda con el resto de los fuentes al entregarlo.