

Java. Práctica 4

Ejercicio 1. Crear una aplicación `TreeDir` que se le pase como argumento un directorio y muestre un listado de todos los ficheros y directorios que contiene. Además, por cada directorio que aparezca deberá repetir la operación. El resultado será algo como:

```
DirectorioRaiz
  Fichero1
  Fichero2
  Directorio1
    Fichero11
    Fichero12
  Fichero3
  Directorio2
    Fichero21
    Directorio21
      Fichero211
      Fichero212
  Fichero4
```

El sangrado será de 5 unidades en cada caso desde un nivel al siguiente.

Ejercicio 2. Crear una aplicación `DFile` que tome un nombre de fichero y cuente cuantas veces aparece cada carácter que es una letra en el fichero. El resto de caracteres se ignora. Al final mostrará una estructura en la que aparecerá cada carácter junto al número de apariciones.

NOTA1.- Distinguir caracteres mayúsculas y minúsculas

NOTA2.- Para pasar un entero a un carácter hay que hacer “casting”

```
integer i = 45;
char c = (char)i;
```

NOTA3.- Para saber si un carácter `c` es una letra se utiliza

```
Character.isLetter(c)
```

Ejercicio 3. Completar la aplicación anterior para que tome otro argumento más que será el nombre de un fichero en el que la estructura resultante pueda ser guardada. Para ello, habrá que guardar primero el número de datos que vamos a almacenar, y luego cada carácter con su frecuencia.

NOTA1.- La estructura donde guardar la información será un `SortedMap`.

NOTA2.- Para iterar sobre un `SortedMap` puede hacerse de dos formas distintas:

- 1) Leyendo el conjunto de claves e iterando sobre este conjunto
`Iterator it = m.keySet().iterator();`
Ahora cada elemento en la iteración es una clave.
- 2) Leyendo el conjunto de entradas e iterando sobre ellas
`Iterator it = m.entrySet().iterator();`
Ahora cada elemento en la iteración es un `Map.Entry`.

Ejercicio 4. Realizar una aplicación que tome como argumento el fichero de datos anterior y reconstruya la estructura. Al final mostrará la estructura por la salida estándar.