ESTRUCTURAS DE DATOS

Curso 2018/19

PRÁCTICA 6

Ficheros

Instrucciones

- *Se debe completar en una sesión.*
- *Práctica individual.*
- *Lee el enunciado completo antes de comenzar. Los comentarios incluidos en el código también pueden proporcionar información útil.*
- *Se habilitará una tarea para que entregues el código desarrollado.*
- *La práctica será APTA si se superan todos los test de validación proporcionados.*

El objetivo de esta práctica es leer imágenes almacenas en ficheros según el formato PGM[1], modificarlas, y guardarlas en un formato de texto binario.

El formato PGM

El formato PGM (*Portable Gray Map*) es un formato de ficheros de texto para almacenar imágenes en niveles de gris. Los ficheros tienen las extensión *.pgm* y suelen tener este formato:

```
P2
4 4
15
0 0 15 15
0 0 15 7
0 0 15 7
15 15 0 0
```

La primera línea contiene un texto identificados que consiste en dos caracteres que podrán tener un valor ”P2”. A continuación, dos números enteros que indican el número de columnas (anchura) y filas (altura) de la imagen. El tercer entero representa el numero de niveles de gris de la imagen.

A partir del cuarto entero empiezan los valores de los pixeles de la imagen, un entero por cada pixel. Los pixeles se almacenan por filas. Es decir, primero se almacenan los pixeles de la fila superior de la imagen, de izquierda a derecha, y, a continuación, los de la segunda fila, y así sucesivamente. El origen de las coordenadas de las imágenes se encuentra en la esquina superior izquierda.

Ten en cuenta que no es necesario que todos los elementos anteriores estén organizados en líneas de texto como aparecen en la imagen. Simplemente necesitan separados por al menos un carácter de espacio en blanco, `' '`, tabuladores, `'\t'` o finales de línea, `'\n'` o `'\r'`.

Os proporcionamos la clase **`ImagenPGM`** que sirve para mantener en memoria imágenes almacenadas en ficheros en formato *PGM*.

[1] https://en.wikipedia.org/wiki/Netpbm_format

Nota: Para que *Eclipee* o *IntelliJ* encuentren los ficheros estos deben encontrarse alojados en el directorio raíz del proyecto que estéis usando para esta práctica.

Ejercicio 1

Implementa el constructor de la clase **`ImagenPGM.`**

```
public ImagenPGM(String nombre) throws IOException
```

El método toma el nombre del fichero que contiene la imagen, y la carga en los atributos internos de la clase. Lanza la excepción **`IOException`** en el caso de que no pueda abrir el fichero con éxito. No es necesario que el fichero está bien formateado ya que todos los que os proporcionamos lo están.

Ejercicio 2

Implementa el método **`cuadrar`**. Este método corta la imagen para que está sea cuadrada, es decir, cortar las filas o columnas que haga falta para el número de filas y columnas sean el menor de ámbos.

Para ver el resultado de esta operación podéis guardar la imagen ya cuadrada. Podéis probar el resultado ejecutando la clase **`ImagenPGM`**, ya que la clase tiene un método **`main`**.

Un formato binario para guardar imágenes

Para el siguiente ejercicio vamos a guardar varias de imágenes en niveles de gris en un fichero binario de formato propio. Este formato binario se estructurará de la siguiente forma:

- El primer **`Integer`** (4 bytes) del fichero indicará el número imágenes almacenadas en el fichero.
- A continuación, una secuencia **`Long`** (8 bytes), uno por imagen, indicando la posición dentro del fichero en la que empieza cada imagen.
- Tras esta cabecera, se guardan los datos de cada imagen. Por cada imagen se guardarán tres **`Integer`** (4 bytes), con el número de columnas (anchura), filas (altura) y niveles de gris respectivamente. Después viene un **`Integer`** (4 bytes) por cada pixel de la imagen. Los pixeles se almacenan por filas, desde la superior a la inferior, y de izquierda a derecha.

Para realizar esta operación os proporcionamos parcialmente implementada la clase **`Binarizador`**.

Ejercicio 3

Implementa el método **`binarizar`**. Este tomará una lista de nombres de ficheros de imágenes con formato *PGM*, **`pgms`**, los leerá y los guardará en fichero binario, **`destino`**, con el formato descrito arriba. Lanzará la excepción **`IOException`** en el caso de que no pueda leer alguno de los ficheros *PGM* o no puede crear el fichero destino.

```
static void binarizar(String destino, List<String> pgms) throws IOException
```

Para escribir este fichero binario debes usar la clase **`RandomAccesFile`**. El número de *bytes* que ocupa un entero o un *long* en memoria se puede obtener con las constantes **`Integer.BYTES`** y **`Long.BYTES,`** respectivamente.

Pruebas de la solución

Como en prácticas anteriores se proporciona un programa de test para verificar el correcto funcionamiento **`BinarizadorTest`**. Pero al contrario que en las prácticas anteriores no hay un test por método, sino que un único test comprueba el proceso completo. Este toma tres imágenes en ficheros con formato *PGM (*`test1.pgm, test2.pgm, test3.pgm`*)*, las cuadra y las vuelve a guardar en formato *PGM*. A continuación, toma estas tres imágenes y genera un fichero binario con ellas. El test consiste en comparar este último con uno de referencia que se os proporciona, `referencia.dat`. Ambos deben ser completamente idénticos.

A parte del test se proporcionan varios programas que permite realizar varias ejecutar diferentes fases del proceso, para poder así realizar una mejor depuración.

- **`ImagenPGM.java`**: El método **`main`** realiza el proceso de leer una imagen, cuadarla y volverla a guardar. Os proporcionamos varias imágenes para que realicéis las pruebas que estiméis pertinentes: `uji.pgm, lena.pgm, avión.pgm, y palmera.pgm.`
- Las imágenes en formato *PGM* pueden ser visualizadas con cualquier editor de imágenes. También se pueden inspeccionar con un editor de texto.
- **`Binarizador.java`**: El método **`main`** permite elegir varias imágenes *PGM* y crear un fichero en formato binario a partir de ellas.
- **`VisorBinario.java:`** Permite leer un fichero binario y mostrar información sobre las imágenes almacenadas en él.
- **`Extraer.java`**: Permite leer un fichero binario, extraer una de las imágenes contenidas en él, y guardarla en formato *PGM*.