Documentación Técnica: Sistema de Manipulación de Imágenes en Java

Mario Gómez Peña (alu.124956@usj.es)

ÍNDICE

1. Introducción

En esta información técnica, se proporcionará una descripción ampliada de las clases implementadas en el sistema de manipulación de imágenes en Java. Dichas clases abarcan funcionalidades como la generación de estructuras de carpetas, la creación de imágenes aleatorias con diferentes formas, el análisis de carpetas y la recopilación de información detallada sobre las imágenes creadas. A través de esta documentación, se buscará brindar una comprensión más profunda de cada clase, contribuyendo al funcionamiento integral del sistema creado para la manipulación de las imágenes.

2. Descripción de las clases

A continuación, se presentan las clases creadas para llevar a cabo la elaboración del sistema de manipulación de imágenes.

2.1 Main.java

La clase Main es la clase principal de este programa, la cual contiene el método main que es el punto de entrada de nuestra aplicación. Esta clase es la encargada de generar una estructura de carpetas y crear imágenes en cada una de ellas utilizando la clase ImageLibrery.

Esta clase, presenta dos métodos: createImagesInFolder y main:

2.1.1 createImagesInFolder

```
private static void createImagesInFolder(Path folderPath, ImageLibrary imageLibrary, int numberOfImages, List<Color> colors, int numberOfShapes, Random random, List<String> formats, int width, int height) throws IOException {}
```

Este método se encarga de crear un número determinado de imágenes en una carpeta específica recibiendo los siguientes parámetros:

- folderPath: parámetro que representa la ruta de la carpeta en la que se crearán las imágenes.
- imageLibrary: es una instancia de la clase ImageLibrary utilizada para crear las imágenes.
- numberOfImages: es el número de imágenes que se van a crear en la carpeta.
- colors: es una lista de objetos de tipo Color que representan los colores empleados en la creación de dichas imágenes.
- numberOfShapes: es el número de formas que se agregarán a cada imagen.
- random: es una instancia de la clase Random utilizada para generar valores aleatorios.
- formats: es una lista de String que representa los formatos de imagen posible.
- width: la achura de las imágenes que vamos a crear
- height: la altura de las imágenes que vamos a crear

2.1.2 main

Este método es el punto de entrada de la aplicación. Se encarga de generar una estructura de carpetas, crear imágenes en cada una de ellas y luego abrir la aplicación App.java utilizando la biblioteca Swing.

1. Crea una instancia de la clase Random para generar valores aleatorios.
2. Genera un número aleatorio entre 2 y 8 para determinar la cantidad de carpetas a crear.
3. Crea una instancia de la clase ImageLibrary.
4. Crea una lista de nombres de carpetas utilizando valores aleatorios.
5. Crea una lista de colores predefinidos.
6. Crea una lista de formatos de imagen predefinidos.
7. Solicita al usuario que ingrese la anchura y altura de las imágenes.
8. Genera la estructura de carpetas utilizando la instancia de ImageLibrary y los nombres de carpeta generados.
9. Recorre todas las carpetas generadas y crea imágenes en cada una de ellas utilizando el método createImagesInFolder.
10. Abre la aplicación App.java utilizando la biblioteca Swing.

2.1.3 Uso de las clases y bibliotecas externas

La clase Main utiliza las siguientes clases y bibliotecas externas:

- java.util.ArrayList: Para la creación de listas dinámicas.
- java.util.Arrays: Para trabajar con arreglos y listas.
- java.util.List: Para trabajar con colecciones de elementos.
- java.util.Random: Para generar valores aleatorios.

- java.util.Scanner: Para leer la entrada del usuario.
- javax.swing.SwingUtilities: Para realizar operaciones relacionadas con la interfaz gráfica de usuario (GUI).
- java.awt.Color: Para representar colores.
- java.nio.file.Files: Para trabajar con archivos y directorios.
- java.nio.file.Path: Para representar rutas de archivos y directorios.
- java.nio.file.Paths: Para obtener instancias de rutas de archivos y directorios.
- imagesCollection.ImageAnalizator: Clase perteneciente a la biblioteca imagesCollection para el análisis de imágenes.
- imagesCollection.ImageInfo: Clase perteneciente a la biblioteca imagesCollection para obtener información de imágenes.
- imagesCollection.ImageLibrary: Clase perteneciente a la biblioteca imagesCollection para la creación de imágenes.
- App: Clase de la aplicación que se abrirá utilizando la biblioteca Swing.

2.2 App.java

La clase App.java, es una clase que extiende de JFrame y representa la interfaz gráfica de usuario (GUI) de la aplicación. Utiliza la biblioteca Swing para crear los componentes visuales y gestionar eventos.

2.2.1 Componentes de la interfaz

La clase App contiene los siguientes componentes de la interfaz:

- table: Un objeto JTable que muestra los detalles de las imágenes en forma de tabla.
- tableModel: Un objeto DefaultTableModel que actúa como modelo para la tabla.
- imageLabel: Un objeto JLabel que muestra la imagen seleccionada en grande.
- thumbnailPanel: Un objeto JPanel que muestra las miniaturas de las imágenes.
- thumbnailScrollPane: Un objeto JScrollPane que permite desplazarse horizontalmente por el panel de miniaturas.
- imageList: Una lista de objetos ImageInfo que contiene información sobre las imágenes.
- imageAnalyzer: Un objeto ImageAnalizator que se utiliza para analizar las imágenes.

2.2.2 Métodos principales

- main: Es el punto de entrada de la aplicación. Crea una instancia de la clase App y la hace visible en el hilo de eventos de Swing utilizando el método invokeLater().
- App: Es el constructor de la clase App. Configura la apariencia y el diseño de la ventana principal de la aplicación. Crea los componentes de la interfaz y los añade al marco principal.
- openFolder: Abre un cuadro de diálogo de selección de carpeta (JFileChooser) para que el usuario pueda seleccionar una carpeta. Luego, utiliza la clase ImageAnalizator para analizar la carpeta seleccionada y obtener información sobre las imágenes contenidas en ella. Actualiza la tabla y las miniaturas con los resultados del análisis.

- analyzeImages: Ordena las imágenes por fecha de creación utilizando el método ordenarPorFecha() de la clase ImageAnalizator. Luego, actualiza la tabla y las miniaturas con la nueva ordenación.
- updateTable: Actualiza la tabla con los datos de las imágenes contenidas en la lista imageList. Elimina todas las filas existentes en el modelo de la tabla y luego agrega una nueva fila por cada imagen.
- updateThumbnails: Actualiza el panel de miniaturas con las miniaturas de las imágenes contenidas en la lista imageList. Elimina todas las miniaturas existentes en el panel y luego agrega una nueva miniatura por cada imagen.

2.2.3 Swing y JFrame

Debido a su madurez, estabilidad y disponibilidad de componentes, Swing y JFrame a menudo se usan en el desarrollo de aplicaciones de interfaz gráfica en Java. Swing ofrece una amplia gama de componentes y le permite adaptar la apariencia de la interfaz. También hay una amplia documentación y recursos. Aunque hay otras bibliotecas para las interfaces gráficas en Java, Swing sigue siendo popular debido a su estabilidad e introducción en la comunidad de desarrollo.

2.3 ImageLibrary.java

La clase ImageLibrary se encarga de generar estructuras de carpetas y crear imágenes aleatorias con formas geométricas. A continuación, se detalla la descripción de los métodos y su funcionamiento:

- `generateFolderStructure(String rootPath, int maxFoldersPerLevel, int maxDepth, List<String> folderNames)`: Este método genera una jerarquía de carpetas con un número aleatorio de carpetas en cada nivel. Recibe como parámetros la ruta raíz donde se creará la estructura de carpetas (rootPath), el número máximo de carpetas por nivel (maxFoldersPerLevel), la profundidad máxima de la estructura (maxDepth) y una lista de nombres de carpetas (folderNames). Esta última lista contiene los posibles nombres que se asignarán a las carpetas creadas. El método utiliza recursión para crear las carpetas de los niveles inferiores.
- `createImage(String path, int width, int height, String format, List<Color> colors, int numberOfShapes)`: Este método crea una imagen aleatoria con formas geométricas. Recibe como parámetros la ruta y nombre del archivo donde se guardará la imagen (path), el ancho y alto de la imagen (width y height respectivamente), el formato de la imagen (format), una lista de colores para las formas (colors) y el número de formas que se dibujarán en la imagen (numberOfShapes). El método utiliza la biblioteca java.awt para crear y manipular la imagen. Genera un contexto de imagen y gráficos, dibuja un fondo blanco y luego dibuja el número especificado de formas geométricas (rectángulos y círculos) en colores aleatorios. Finalmente, guarda la imagen en el archivo especificado.

Es importante tener en cuenta que ambos métodos pueden lanzar excepciones de tipo IOException en caso de error al crear las carpetas o al guardar la imagen.

La clase también tiene una instancia de la clase Random para generar números aleatorios y un constructor vacío.

2.4 ImageInfo.java

La clase ImageInfo se utiliza para obtener información de una imagen especificada por su ruta. A continuación, se detalla la descripción de los atributos y métodos de la clase:

- Atributos:
  - path: Representa la ruta de la imagen.
  - fechaCreacion: Almacena la fecha de creación del archivo.
  - ancho: Guarda el ancho de la imagen.
  - alto: Guarda la altura de la imagen.
- Constructor:
  - ImageInfo(String path): Crea una nueva instancia de la clase ImageInfo a partir de la ruta de la imagen especificada. Dentro del constructor, se lee la imagen utilizando la biblioteca javax.imageio.ImageIO y se obtiene su ancho y alto. Además, se obtiene la fecha de creación del archivo utilizando la clase File. Se utiliza el método convertMillisToLocalDateTime() para convertir los milisegundos obtenidos de file.lastModified() a un objeto LocalDateTime.
- Métodos:
  - getPath(): Retorna la ruta de la imagen.
  - getFechaCreacion(): Retorna la fecha de creación de la imagen en formato LocalDateTime.
  - getAncho(): Retorna el ancho de la imagen.
  - getAlto(): Retorna la altura de la imagen.
  - convertMillisToLocalDateTime(long millis): Método privado que convierte milisegundos a un objeto LocalDateTime. Se utiliza para obtener la fecha de creación del archivo a partir de los milisegundos obtenidos.

La clase ImageInfo provee una forma conveniente de obtener información básica de una imagen, como la ruta, fecha de creación, ancho y altura. Es importante tener en cuenta que el constructor puede lanzar una excepción de tipo IOException en caso de que ocurra un error al trabajar con la imagen.

2.5 ImageAnalizator.java

La clase ImageAnalizator se utiliza para analizar una carpeta y obtener información de las imágenes contenidas en ella. A continuación, se detalla la descripción de los atributos y métodos de la clase:

- Atributos:
  - imagenes: Lista que almacena objetos ImageInfo que representan la información de las imágenes encontradas.
- Constructor:
  - ImageAnalizator(): Crea una nueva instancia de la clase ImageAnalizator y inicializa la lista de imágenes.
- Métodos:
  - analizarCarpeta(String path): Recorre de forma recursiva la carpeta especificada por la ruta path y obtiene la información de las imágenes encontradas. Si path es una imagen válida, se crea un objeto ImageInfo

con la información de la imagen y se agrega a la lista de imágenes. Si path es una carpeta, se analizan todos los archivos y subcarpetas contenidos en ella.

- esImagen(File file): Verifica si un archivo es una imagen válida. Se comprueba el nombre del archivo y se verifica si termina con las extensiones ".png", ".jpg" o ".jpeg". Retorna true si el archivo es una imagen, y false en caso contrario.
- ordenarPorFecha(boolean ascendente): Ordena la lista de imágenes por fecha de creación. Se utiliza un comparador basado en el atributo fechaCreacion de la clase ImageInfo. Si ascendente es true, se ordena de forma ascendente, de lo contrario, se ordena de forma descendente.
- filtrarPorAncho(int anchoMinimo): Filtra la lista de imágenes y retorna una nueva lista que contiene únicamente las imágenes cuyo ancho es mayor o igual a anchoMinimo.
- getImagenes(): Retorna la lista de imágenes analizadas.

La clase ImageAnalizator provee funcionalidades para analizar una carpeta y obtener información de las imágenes contenidas en ella. Permite ordenar las imágenes por fecha de creación, filtrarlas por ancho mínimo y obtener la lista de imágenes analizadas. Es importante tener en cuenta que el método analizarCarpeta() puede lanzar una excepción de tipo IOException en caso de que ocurra un error al trabajar con las imágenes.

3. Conclusión

Para poder realizar dicha práctica, hemos tenido que abordar diferentes aspectos, como la generación de estructuras de carpetas, la creación de imágenes aleatorias y el análisis de imágenes existentes.

La clase ImageLibrary proporciona métodos para generar estructuras de carpetas de forma aleatoria, así como para crear imágenes personalizadas con formas y colores aleatorios.

La clase ImageInfo permite obtener información específica de una imagen existente, como su ruta, fecha de creación, ancho y alto.

La clase ImageAnalizator brinda funcionalidades para analizar una carpeta y recopilar información de las imágenes contenidas en ella. Con esta clase, pudimos explorar las imágenes en una estructura de carpetas y ordenarlas por fecha de creación, así como filtrarlas según su ancho.

Respecto al apartado 1.3 de edición de metadatos, he consultado diversas fuentes tales como diversos foros de stackoverflow, chatgpt,.. Con esto, he llegado a la conclusión de que no lo puedo poner en mi trabajo, puesto que, al no entender qué es lo que se estaba haciendo, he preferido no poner algo que ni siquiera yo entiendo.

4. Referencias

https://docs.oracle.com/javase/tutorial/essential/io/fileAttr.html

https://stackoverflow.com/questions/18955475/add-custom-attribute-or-metadata-to-file-java

https://help.sap.com/saphelp_snc700_ehp01/helpdata/en/f7/81a2b422abeb4da9ee4c283b423f38/content.htm?no_cache=true

https://www.logicbig.com/how-to/code-snippets/jcode-java-io-files-createdirectories.html

https://docs.oracle.com/javase/8/docs/api/java/nio/file/Files.html

https://gzrobe.wordpress.com/2009/09/15/imagenes-aleatorias-en-java/

https://www.youtube.com/watch?v=yds3UoBNjjU

https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/TableView.html

https://javiergarciaescobedo.es/programacion-en-java/96-javafx/494-ejemplo-de-uso-de-imageview-en-javafx

https://docs.oracle.com/javase/8/javafx/api/javafx/scene/image/ImageView.html

https://www.youtube.com/watch?v=D2iKmxdjhTE

https://es.stackoverflow.com/questions/2033/c%C3%B3mo-crear-un-carrusel-de-im%C3%A1genes-al-estilo-de-la-pantalla-de-inicio-de-sesi%C3%B3n

https://es.javascript.info/task/carousel

https://github.com/hjohn/Carousel/blob/master/Carousel/src/main/java/hs/javafx/control/TestCoverflow.java