SISTEMA DE MANIPULACIÓN DE IMÁGENES EN JAVA

Mario Gómez Peña
(alu.124956@usj.es)

OBJETIVOS A CUMPLIR:

- Generación de estructuras de carpetas
- Creación de imágenes aleatorias con diferentes formas
- Análisis de carpetas
- Recopilación de información sobre una imagen creada

5 CLASES:

- Main.java
- App.java
- ImageLibrary.java
- ImageInfo.java
- ImageAnalizator.java

```java
// Método para crear un número determinado de imágenes en una carpeta específica
private static void createImagesInFolder(Path folderPath, ImageLibrary imageLibrary, int numberOfImages, List<Color>
    // Bucle para crear las imágenes
    for (int i = 0; i < numberOfImages; i++) {
        // Selecciona un formato de imagen aleatoriamente
        String format = formats.get(random.nextInt(formats.size()));
        // Se construye el nombre de la imagen con el formato "image_i.format"
        String imageName = "image_" + i + "." + format;
        // Se construye la ruta completa de la imagen
        String imagePath = folderPath.resolve(imageName).toString();
        // Se llama al método para crear la imagen
        imageLibrary.createImage(imagePath, width, height, format, colors, numberOfShapes);
    }
}
```

MAIN.JAVA

```java
public static void main(String[] args) {
    Random random = new Random();

    int numFolders = random.nextInt(bound:7) + 2; // Generar un número aleatorio entre 2 y 8 para las carpe

    // Creación de una instancia de ImageLibrary
    ImageLibrary imageLibrary = new ImageLibrary();

    // Creación de una lista de nombres de carpetas
    List<String> folderNames = new ArrayList<>();
    for (int i = 0; i < numFolders; i++) {
        folderNames.add("Carpeta" + random.nextInt(bound:1000)); // Generar un número aleatorio entre 0 y 1
    }

    // Creación de una lista de colores
    List<Color> colors = Arrays.asList(Color.RED, Color.GREEN, Color.BLUE, Color.YELLOW, Color.ORANGE);

    // Creación de una lista de formatos de imagen
    List<String> formats = Arrays.asList(...a:"png", "jpeg", "jpg");

    Scanner scanner = new Scanner(System.in);

    System.out.println(x:"Introduce la anchura de las imagenes:");
    int width = scanner.nextInt();

    System.out.println(x:"Introduce la altura de las imagenes:");
    int height = scanner.nextInt();

    scanner.close();
```

```java
    int numSubfolders = 5;
    try {
        // Generación de la estructura de carpetas
        for(String folderName : folderNames) {
            imageLibrary.generateFolderStructure("images/"+folderName, numSubfolders--, maxDepth:1, Arrays.asList(folderName));
        }

        // Recorrido de todas las carpetas generadas
        Files.walk(Paths.get(first:"images")).filter(Files::isDirectory).forEach(folderPath -> {
                try {
                    // Creación de imágenes en cada carpeta
                    // Hay que tener en cuenta que puede crear 0 imagenes, por lo que puede lanzar una excepción      You, 1 se
                    int numImages = random.nextInt(bound:10) + 1; // Generar un número aleatorio entre 1 y 10 para las imágenes
                    createImagesInFolder(folderPath, imageLibrary, numImages, colors, random.nextInt(bound:20) + 1, random, for
                } catch (IOException e) {
                    // Gestión de posibles errores
                    e.printStackTrace();
                }
            });
    } catch (IOException e) {
        // Gestión de posibles errores
        e.printStackTrace();
    }

    // Llamamos a App.java para que se abra automáticamente
    SwingUtilities.invokeLater(() -> {
        App app = new App();
        app.setVisible(b:true);
    });
}
}
```

MAIN.JAVA

IMAGELIBRARY.JAVA

```java
// Método para generar la jerarquía de carpetas con un número aleatorio de carpetas en cada nivel
public void generateFolderStructure(String rootPath, int maxFoldersPerLevel, int maxDepth, List<String> folderNames) th
    // Creo la carpeta raíz
    generateFolderStructure(rootPath, maxFoldersPerLevel, maxDepth, folderNames, currentDepth:0);
}

// Método que simula el trabajo real de crear una jerarquía de carpetas
private void generateFolderStructure(String parentPath, int maxFoldersPerLevel, int maxDepth, List<String> folderNames,
    // Si se alcanza la profundidad máxima, no se crean más carpetas
    if (currentDepth >= maxDepth) {
        return;
    }

    // Decido aleatoriamente cuántas carpetas se crearán en este nivel
    int numberOfFolders = random.nextInt(maxFoldersPerLevel) + 1;

    for (int i = 0; i < numberOfFolders; i++) {
        // Elijo el nombre que le voy a poner a las carpetas
        String folderName = folderNames.get(random.nextInt(folderNames.size()));
        // Genero la ruta para la nueva carpeta
        String newPath = parentPath + File.separator + folderName + "_" + currentDepth + "_" + i;
        // Creo la carpeta
        Files.createDirectories(Paths.get(newPath));
        // Llamo recursivamente al método para crear las carpetas de los niveles inferiores
        generateFolderStructure(newPath, maxFoldersPerLevel, maxDepth, folderNames, currentDepth + 1);
    }
}
```

```java
    // Creo un nuevo contexto de imagen y gráficos
    BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
    Graphics2D graphics = image.createGraphics();

    // Dibuja el fondo blanco
    graphics.setColor(Color.WHITE);
    graphics.fillRect(x:0, y:0, width, height);

    // Dibuja el número especificado de formas
    for (int i = 0; i < numberOfShapes; i++) {
        // Elijo un color aleatorio
        Color color = colors.get(random.nextInt(colors.size()));
        graphics.setColor(color);

        // Elijo aleatoriamente si dibujaré un rectángulo o un círculo
        int shapeType = random.nextInt(bound:2);

        if (shapeType == 0) {
            // Dibuja un rectángulo
            double x = random.nextDouble() * width;
            double y = random.nextDouble() * height;
            double w = random.nextDouble() * (width - x);
            double h = random.nextDouble() * (height - y);
            Rectangle2D rectangle = new Rectangle2D.Double(x, y, w, h);
            graphics.draw(rectangle);
        } else {
            // Dibuja un círculo
            double x = random.nextDouble() * width;
            double y = random.nextDouble() * height;
```

IMAGELIBRARY.JAVA

IMAGEINFO.JAVA

```java
public ImageInfo(String path) throws IOException {
    this.path = path;

    // Leemos la imagen y obtiene su ancho y alto
    BufferedImage image = ImageIO.read(new File(path));
    this.ancho = image.getWidth();
    this.alto = image.getHeight();

    // Obtenemos la fecha de creación del archivo
    File file = new File(path);
    // Convertimos los milisegundos a LocalDateTime
    this.fechaCreacion = convertMillisToLocalDateTime(file.lastModified());
}
```

```java
public void analizarCarpeta(String path) throws IOException {
    // Crea un objeto File con la ruta de la carpeta
    File folder = new File(path);

    // Comprueba si el archivo es una imagen y si es así, recoge su información
    if (folder.isFile() && esImagen(folder)) {
        // Añade la imagen a la lista
        imagenes.add(new ImageInfo(folder.getAbsolutePath()));
    } else if (folder.isDirectory()) {
        // Si es una carpeta, recorre todos sus archivos y subcarpetas
        for (File file : folder.listFiles()) {
            // Llama recursivamente al método para analizar la carpeta
            analizarCarpeta(file.getAbsolutePath());
        }
    }
}

// Método para comprobar si un archivo es una imagen
private boolean esImagen(File file) {      You, 3 weeks ago • Exersice 2 of the Individual Project …
    // Obtiene el nombre del archivo y lo convierte a minúsculas
    String name = file.getName().toLowerCase();
    // Comprueba si el nombre del archivo termina con ".png", ".jpg" o ".jpeg"
    return name.endsWith(suffix:".png") || name.endsWith(suffix:".jpg") || name.endsWith(suffix:".jpeg");
}
```

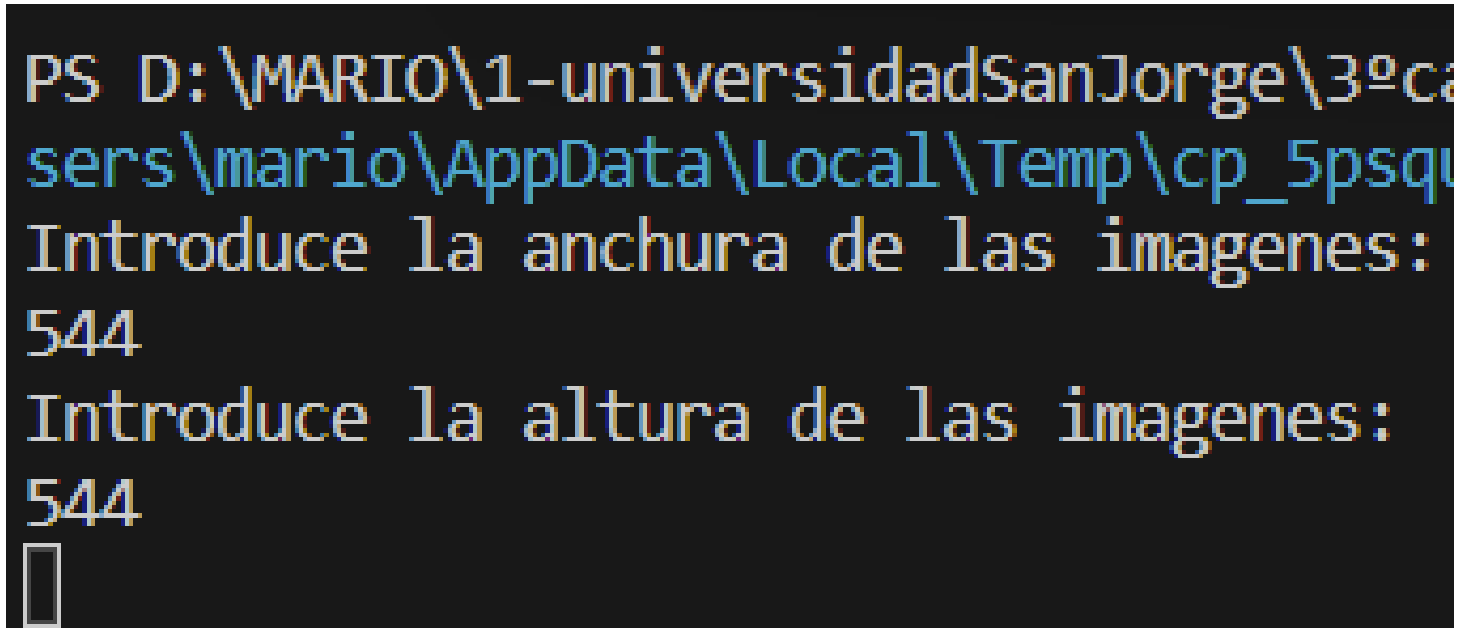
IMAGEANALIZATOR.JAVA

```java
// Método para ordenar las imágenes por fecha de creación
public void ordenarPorFecha(boolean ascendente) {
    // Se crea un comparador para comparar las fechas de creación
    Comparator<ImageInfo> comparator = Comparator.comparing(ImageInfo::getFechaCreacion);
    // Si se quiere ordenar de forma descendente, se invierte el comparador
    if (!ascendente) {
        comparator = comparator.reversed();
    }
    imagenes.sort(comparator);
}

// Método para filtrar las imágenes por ancho
public List<ImageInfo> filtrarPorAncho(int anchoMinimo) {
    return imagenes.stream().filter(imagen -> imagen.getAncho() >= anchoMinimo).collect(C
}

// Función para obtener la lista de imágenes
public List<ImageInfo> getImagenes() {
    return imagenes;
}
```

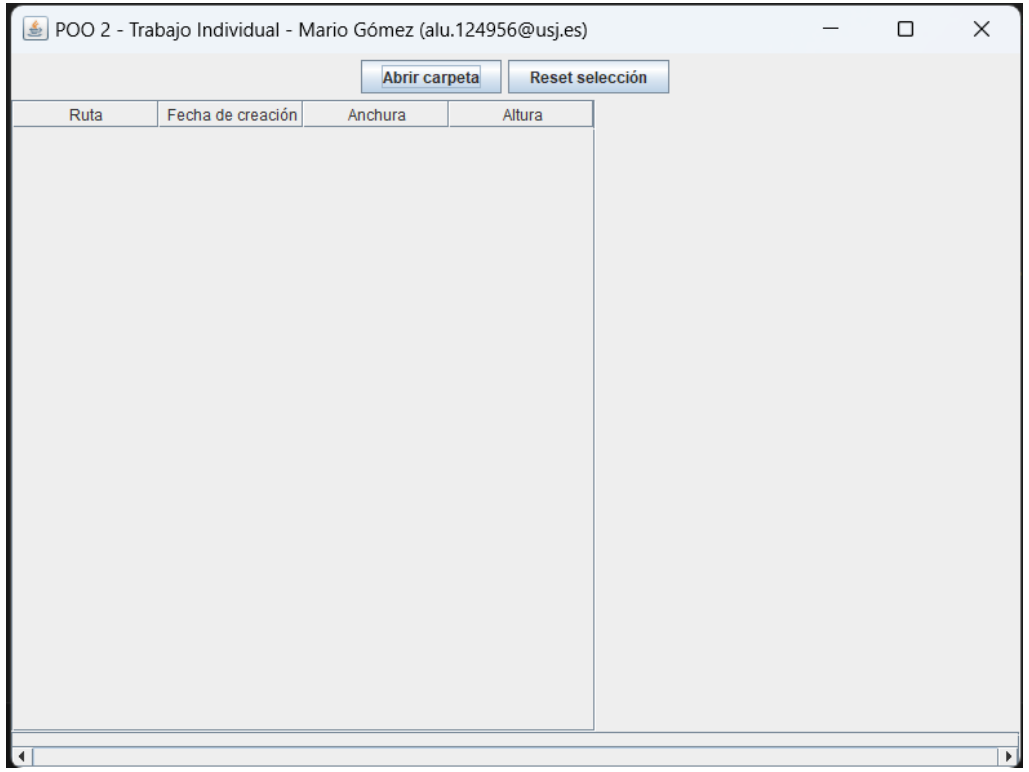
IMAGEANALIZATOR.JAVA

APP.JAVA

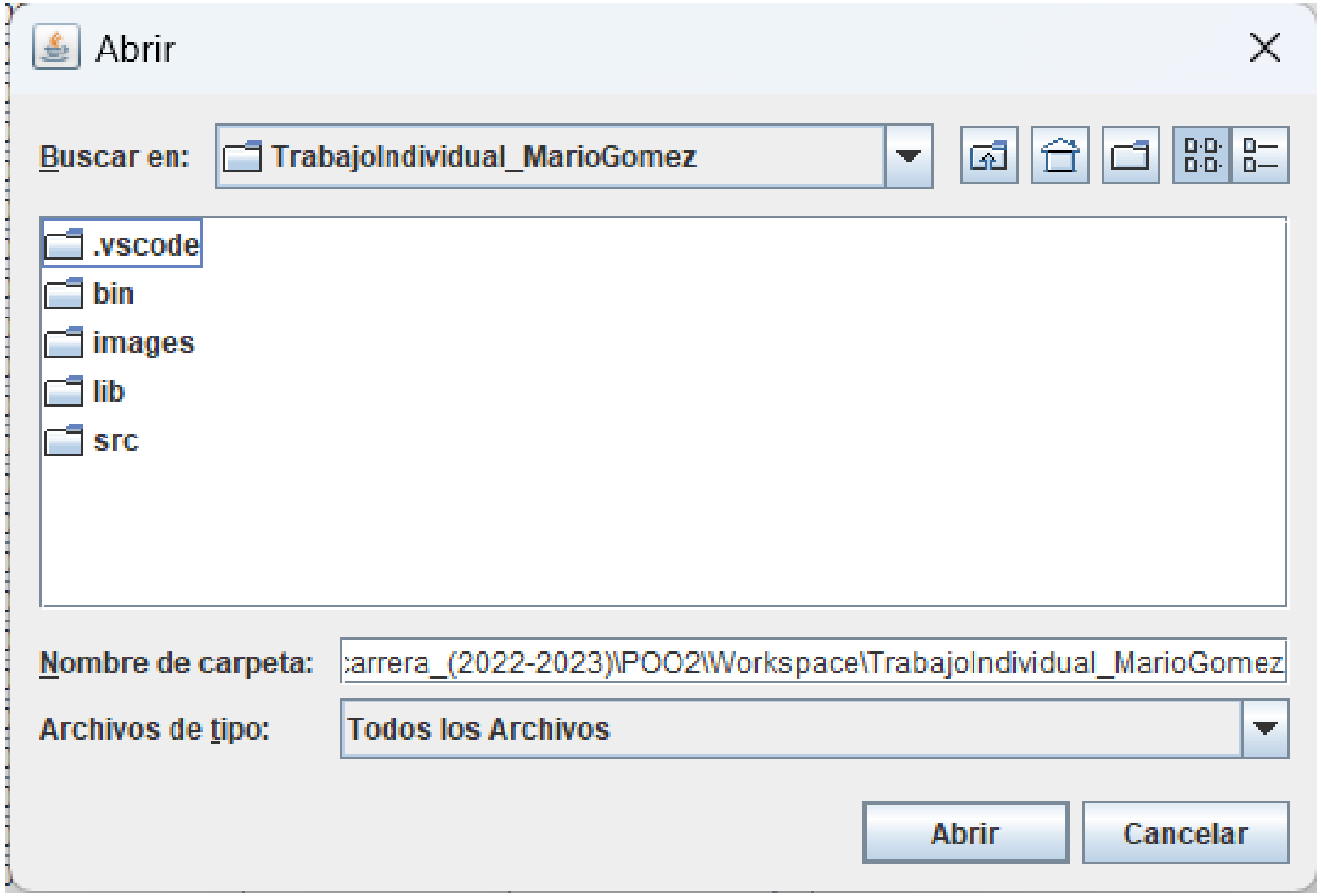
Nos puede devolver una excepción si no crea ninguna imagen.



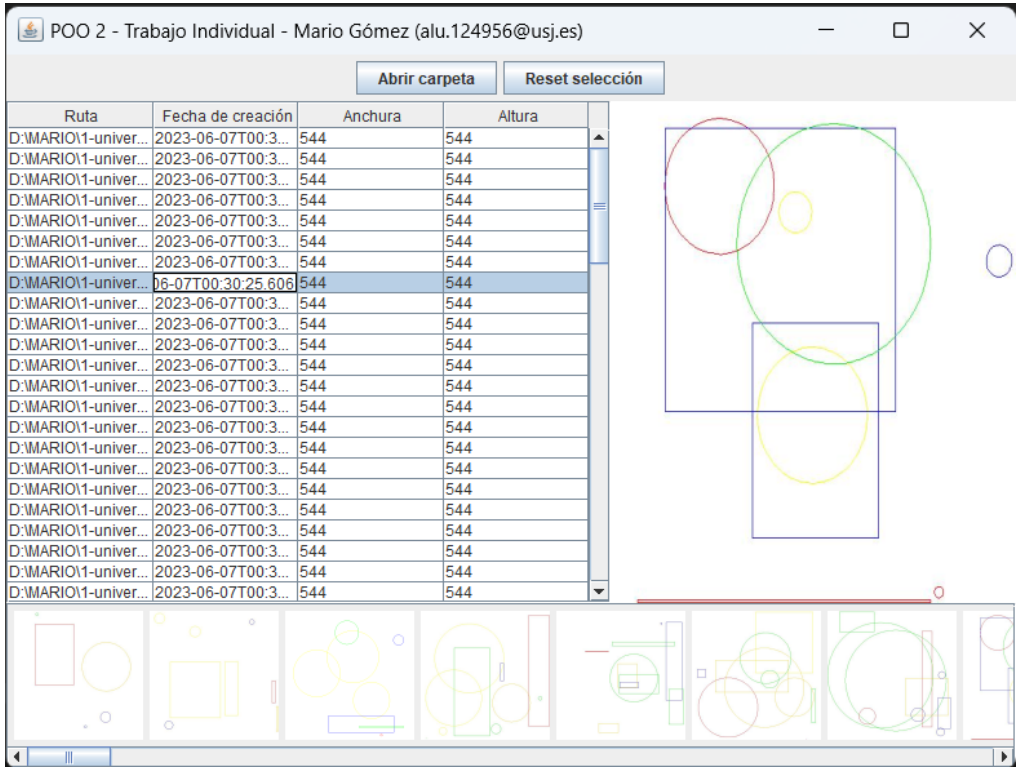
APP.JAVA


POO 2 - Trabajo Individual - Mario Gómez (alu.124956@usj.es)
Abrir carpeta
Reset selección
Ruta
Fecha de creación
Anchura
Altura

APP.JAVA


Abrir
Buscar en:
TrabajoIndividual_MarioGomez
.vscode
bin
images
lib
src
Nombre de carpeta:
:arrera_(2022-2023)\POO2\Workspace\TrabajoIndividual_MarioGomez
Archivos de tipo:
Todos los Archivos
Abrir
Cancelar

APP.JAVA


POO 2 - Trabajo Individual - Mario Gómez (alu.124956@usj.es)
Abrir carpeta
Reset selección
Ruta
Fecha de creación
Anchura
Altura
D:\MARIO\1-univer...
2023-06-07T00:3...
544
D:\MARIO\1-univer...
06-07T00:30:25.606
544