Content

Chapter1:Introduction

This project implements an image grayscale conversion application based on java, which is divided into three components, Pixel, ImageFile and GrayscaleConverter.

1.1 Pixel

The Pixel component encapsulates the color information of a single pixel (red, green, blue, and transparency values), and this class provides getter methods and settter methods to obtain and modify the color information of a single pixel, ensuring that the values are always in a valid range. In addition, it records the x and y coordinate positions in the pixel word image grid.

1.2 ImageFile

The ImageFile component supports loading images from files, creating blank images to convert grayscale and manipulating individual pixels, setting up exception handling to catch errors, and finally setting up to save modified images to a file.

1.3 GrayScale

The GrayScale econverter component implements the function of converting color images to grayscale images. The makeGray method determines the grayscale value by calculating the average of the red, blue, and green components of each pixel, while the convertAndSave method loops through ImageFile through an enhancement statement for, processing multiple images in batches, and finally saving the grayscale image as a new file name with a "gray-" prefix.

Chapter2:Implementation

In this chapter, I'll explain in detail the methods of the Pixel, ImageFile, and GrayscaleConverter classes, what they do, and how they relate to each other. The Pixel is the core component of the ImageFile, and the GrayscaleConverter relies on the Pixel and ImageFile for implementation.

2.1 Pixel

Pixel encapsulates the color information (red, green, blue, and transparency) of each pixel in the image, as well as the position of the pixel in the image.Provides an interface to access and modify the color components and positions of pixels, and ensures that color values are always in the correct range

Pixel(int i, int x, int y)
Constructor: Initializes the color information of the pixel with an integer value (in ARGB format) and specifies the bit of the pixel

getRed ( ), getGreen ( ), getBlue ( )
Returns the red, green, and blue components of the pixel, respectively (range: 0-255).

getValue ( )
Returns the Alpha, red, green, and blue values of the pixel as an integer (in ARGB format).

setRed (int r), setGreen (int g), setBlue (int b)
Update the red, green, or blue components of the pixel and ensure that the values are in the 0-255 range.

setValue(int pixel)
Using displacement and bit-and-sum operations, the Alpha, Red, Green, and Blue components are extracted and assigned to the attributes of the pixel object, respectively.

getX ( ), getY ( )
Returns the horizontal and vertical coordinates of the pixel in the image.

guard(int value)
Make sure that the input color component values are within the legal range (0 to 255), and correct for negative values or values greater than 255.

2.2 ImageFile

ImageFile implements the functions of loading, manipulating and saving image data. It supports loading images from files, creating blank images to store images after gray level conversion, supporting pixel group access and modification, and providing the function of saving images to files.

ImageFile(File file)
Supports loading images from files and initializing ImageFile objects.

ImageFile(int width, int height)
Creates a blank image with the specified width and height.

getBlankImage(int width, int height)
Creates a blank image with a specified width and height.

getFileName( )
Returns the file name of the current image, or an empty string if not set.

getWidth( ), getHeight( )
Returns the width and height of the image.

getImageFromFile(String fileName)
Loads the image from the file and returns the BufferedImage object.

getPixel(int x, int y)
Returns the Pixel object with the specified coordinate (x, y) in the image.

imageToPixels(Image image)
Converts raw image data into an array of Pixel objects.

init(File f)
Load images through files and initialize Pixel data.

init(String fileName, BufferedImage image)
Initializes ImageFile with the file name and image object, setting the width, height, and pixel data.

integersToPixels(int[ ] pixels, int width, int height)
Converts an array of integer Pixel values to an array of Pixel objects.

pixels( )
Returns an iterable of all pixels in the image for traversing the pixels.

pixelsToIntegers(Pixel[ ] pixels)
Converts an array of Pixel objects to an array of integer pixel values.

save( )
Function: Save the current image to a file, call saveAs() if the file name is not set.

saveAs( )
Opens the File selection dialog box, allowing the user to select a new file name and save the image.

setFileName(String name)
Sets the file name of the image.

setPath(String fileName)
Decompose the file path into a path and a file name, stored in myPath and myFileName, respectively.

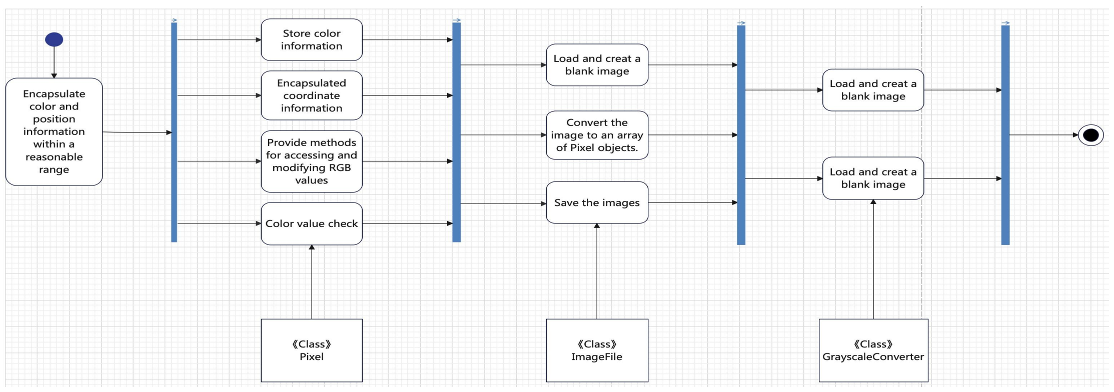
updateImage( )
Update the image object BufferedImage based on the Pixel array to reflect the latest pixel values.

2.3 GrayscaleConverter

GrayscaleConverter implements the function of converting images to grayscale images and provides the function of batch processing and saving to specific formats.

makeGray (ImageFile originalImage):
The input ImageFile is converted to grayscale image, and grayscale is achieved by calculating the average value of each pixel RGB value.

convertAndSave (Filel[ ] imageFiles):
Multiple image files are processed in batches, converted to grayscale and saved to the target directory. The file name prefix gray- is automatically added when gray image is saved.


Encapsulate color and position information within a reasonable range
Store color information
Encapsulated coordinate information
Provide methods for accessing and modifying RGB values
Color value check
Load and creat a blank image
Convert the image to an array of Pixel objects.
Save the images
Load and creat a blank image
Load and creat a blank image
《Class》 Pixel
《Class》 ImageFile
《Class》 GrayscaleConverter

Chapter3:Reflective account

During the development of this project, I encountered many problems, such as challenges of environment configuration, programming errors, and test failures.

3.1 Challenge of environment configuration

Because the Codio version is too old, there is no latest Maven version in the resource library, and the latest java version cannot be downloaded, resulting in no project development.

I downloaded the latest version from Maven official website, uploaded it to codio, and then extracted it to the specified path, so as to configure Maven 3.9.9, update java 11 to java 17, and finally configure the environment successfully.

3.2 Challenge of programming errors

Since it is the first time to write java programs, I am not familiar with this language, so I have encountered some problems when reading code and writing code.

I separated the methods of each class to understand them one by one, consulted relevant information on CSDN, and asked teachers or classmates. I first understood the functions of each of the three categories, then wrote and debugged the code, read 《Java in a Nutshell: A DesktopQuick Referenceand》 (I shall refer to it in chapter 5), finally successfully solved the problem of programming errors.

3.3 Challenge of test failures

Despite the completion of the code understanding and writing, the IDE also ran successfully, but the Codio Terminal failed to run.

I successfully passed the test by checking for logical errors and typos in the code and reconfiguring the Codio environment.

Chapter4:Conclusions

In the conclusion part, I will summarize the main points in my report.

4.1 Introduction

In the Introduction section, I introduced the core features of the project. this project uses Pixel, ImageFile and GrayscaleConverter to convert image information into RGB information and position information of a single pixel, and then upload, modify and save the information to a specific path. Finally, by setting the average value of the RGB component of the image to each pixel, and supports batch processing, the function of image gray level conversion is realized.

4.2 Implementation

In the Implementation section, I elaborate on the three core classes and main methods of the project. pass

The Pixel class represents a single pixel, the ImageFile class abstracts the entire image, and the GrayscaleConverter class realizes grayscale conversion and batch processing, and provides corresponding uml diagrams for interpretation.

4.3 Reflective account

In the Reflective account section, I review the challenges I encountered during the development of the project, such as the challenges of environment configuration, programming errors, and test failures, and suggest solutions.

Chapter5:References

For this project, I consulted the following book, which provides basic knowledge and technical background in the java field to guide the development and design of my project, help me better understand and apply java for ImageGrayscaleConversion development:

Evans, B. (2018). Java in a Nutshell: A Desktop Quick Reference (7th ed.).
Sebastopol, CA: O'Reilly Media.