Andrei Paul-Iulian

converting a color image to greyscale

# introduction

The purpose of the application is to process a color image in BMP format and convert it into an image in shades / grayscale. The user of the application specifies both the location of the image that wants to be converted and the location and name of the grayscale image. It receives warning messages if the source and destination addresses are not valid and information about the performance of the application.

# Description of the required application

As the title of this paper says, the application has the role of converting from an RGB image to a Grayscale one. The image must have the format .bmp, and the size of a pixel to be 24 bits (8 bits x 3 channels). The conversion from color to black and white will have to be done using the arithmetic mean method. The application must be accompanied by a document explaining its functionalities. She must also record the time of each operation she performs reading, processing, writing.

# THEORETICAL PART

An image can be considered a matrix of pixels. A pixel has 4 characteristics represented by numerical values (between 0 and 255):

1. A (Alpha) – pixel transparency
2. R (Red) – the red color value of the pixel
3. G (Green) – the value of the pixel's green color
4. B (Blue) – the value of the pixel blue color

A pixel that has different R, G, B values is considered a color pixel. Grayscale is obtained by calculating an average between the values R, G, B and replacing each one with the calculated average. Consequently, a gray pixel has the 3 equal values.

Multithreading is a key concept in programming, which allows the execution of multiple control flows simultaneously in a single program. This can lead to a more efficient use of system resources and improve the performance of the application.

For multithreading I made a "Producer-Consumer" model with threads. This model is useful when there is a manufacturer that generates data and a consumer who processes this data. In this case, the manufacturer reads an image from a specific location and divides it into four equal parts, and the consumer processes each part at the time it is read. The benefits of this model are that it can reduce latency and improve system performance through parallel data processing.

# Descrierea applicationi

## Application components

1. **Abstract image** class that implements **Interface**. It cannot be instantiated.
2. **FormatPixel** class that inherits the Image class. It has as an attribute *format* that specifies whether the image is color or grayscale. Its value is established by the *verification* method.
3. **The Greyscale** class that inherits the FormatPixel class. In *the greey* method, the value of the format attribute is checked and depending on this, the image is converted to greyscale.
4. **Read** and **Write** classes.
5. **Main** class in which the main method is found that instantiates greey and BufferedImage objects
6. **Producer and Consumer** classes to implement multithreading and to implement the communication requirement through Pipes.
7. **WriterResults** class exclusively for the help of Pipes communication with the help of the Producer and Consumer classes

## How it works

The address of the image is read from the command line. It treats the exception in which the address is invalid by displaying a message in the console. The format of the image pixels is checked. If the image is already grayscale, a message is displayed in the console and the execution of the program ends. If the image is color, it is converted to Greyscale by the method explained in the theoretical part. It is then read the destination where the converted image is written. Also, if the address is invalid, a message is displayed in the console.

The execution type of each stage is displayed in the console in nanoseconds.

Each component of the code is commented so you can find additional information in the source code from the project archive.

For multithreading I created two threads, one for reading the source image (Producer Thread) and one for its processing (Consumer Thread). I used the Thread class in the java.lang package to create these threads.

Producer Thread read the source image from a specified path and transmitted it to the Consumer Thread via a FormatPixel class. This class contains methods to check if the image is color or Grayscale and to convert a color image to Grayscale.

To prevent threads from being blocked, we used the synchronization mechanism. We implemented this synchronization through the wait() and notify() methods of the Object class. In this example, the Producer thread enters Not Runnable after reading each quarter of the image and waits for the Consumer thread to process this information before continuing reading.

We also used communication through Pipes, which allows data transmission between two threads through an input/output flow. We used the PipedOutputStream and PipedInputStream classes in the java.io package to create a pipe between the Producer thread and the Consumer thread. I partitioned the image into 4 segments and displayed a message when transmitting and receiving each segment to highlight the stages of communication.

Finally, we created a WriterResult class that deals with writing the processed image in the file. This class uses the ImageIO class in the javax.imageio package to write the image in JPEG format in the specified file.

We illustrated the use of synchronization and communication mechanisms through Pipes to manage communication between the threads of the Producer-Consumer application. The use of threads and synchronization mechanisms allows efficient processing of the source image and preventing thread blocking. The use of communication through Pipes allows the transmission of information between threads without the need for a global variable or another method of communication.

# Demonstration

Color image:



Image in Greyscale:

A tree with the sun shining through it

Description automatically generated with medium confidence

# Conclusions

The Java programming language is a powerful tool in digital image processing through algorithms and mathematical operations. It can be used in many types of applications, from the simplest, such as adjusting contrast or transparency, to the most complex ones, such as facial recognition.

# Bibliography

https://whatis.techtarget.com/definition/grayscale

<https://www.oreilly.com/openbook/javawt/book/ch12.pdf>

<https://www.geeksforgeeks.org/image-processing-java-set-1-read-write/>