IMAGE EDITOR PROJECT DOCUMENTATION

By Arjun Kshirsagar

ID No. - Arjun.23BCS10157

Batch 1

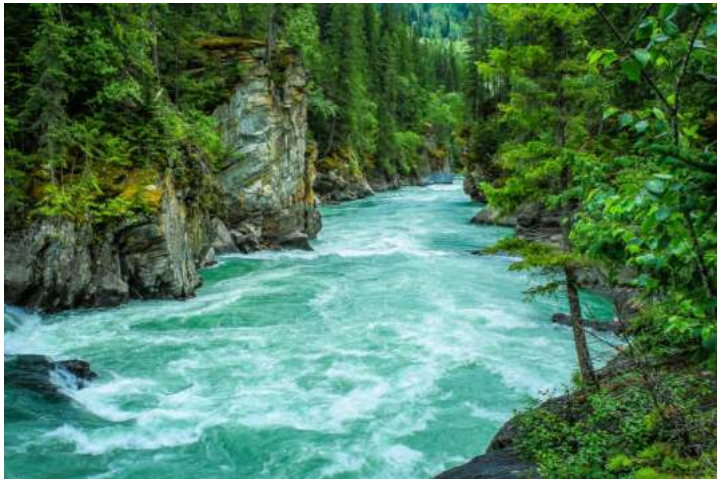
Date of Submission: 3rd Sept, 2023

Instructor: Mr. Kshitij Mishra

Overview

This document provides an explanation of the Java program named ‘ImageEditorByArjunKshirsagar`. This program is designed to perform various image processing operations on a given image file. It includes functions to convert an image to grayscale, change its brightness, flip it horizontally or vertically, and rotate it clockwise or anti-clockwise.

Default Input Image:- image.jpg



How to Use the Program?

1. Run the program.
2. When prompted to input the file name, provide the path location of the file.
3. Now the program will run and ask for the desired operation, select the Index Number of Operation (e.g., 1 for Grayscale conversion).
4. If the operation requires additional input (e.g., brightness adjustment), enter the required value (e.g., 50 for a 50% brightness increase).
5. Specify the output filename (e.g., `output_image.jpg`).
6. The program will process the image and save the resulting image as `output_image.jpg`.

Libraries(Import Statements)

```
import java.awt.Color;
```
This imports the `Color` class from the `java.awt` package, allowing you to work with colors in Java graphical applications.

```
import java.awt.image.BufferedImage;
```
This imports the `BufferedImage` class from the `java.awt.image` package, which is used to create and manipulate images in Java applications.

```
import java.io.File;
```
This imports the `File` class from the `java.io` package, enabling you to interact with files and directories in your Java program.

```
import java.io.IOException;
```
This imports the `IOException` class from the `java.io` package, which is used to handle exceptions related to input and output operations, such as file reading and writing.

```
import java.util.Scanner;
```
This imports the `Scanner` class from the `java.util` package, which allows you to read input from various sources, such as the keyboard or files, in a Java program.

```
import javax.imageio.ImageIO;
```
This imports classes from the `javax.imageio` package, which provides a set of APIs for reading and writing images in various formats, like JPEG or PNG, in Java applications.

Features

1: Convert to Grayscale
2: Change Brightness
3: Inversion
4: Mirror Image
5: Rotate Clockwise
6: Rotate Anti-Clockwise
7. Blur
8. Circle Crop

In the subsequent slides you can get the overview of all the functions and their output images upon the successful execution of the code.

(To understand the code, please read the ImageEditor.java *file as it has extensive comments with explanation of each code)*

GreyScale

In Java, BufferedImage.TYPE_BYTE_GRAY is a constant that represents a type of BufferedImage with grayscale color information. It's one of the image types available in Java BufferedImage class, and it indicates that each pixel in the image is represented by a single byte (8 bits) of data, which typically ranges from 0 (black) to 255 (white) to represent different shades of gray.

Here's how you can create a BufferedImage with this type:

GREY-SCALE OUTPUT IMAGE

```
BufferedImage grayImage = new BufferedImage(width, height, BufferedImage.TYPE_BYTE_GRAY)
```

Brightness Adjustment

In Java, `BufferedImage.TYPE_3BYTE_BGR` is one of the predefined image types for the `BufferedImage` class. It represents a buffered image with 24 bits of color information per pixel arranged in the Blue-Green-Red (BGR) format. Each pixel in this image type is typically represented using 3 bytes, where the first byte represents the blue component, the second byte represents the green component, and the third byte represents the red component.

Here's a breakdown of the format:

- Byte 1 (Blue): Represents the intensity of the blue color channel, usually in the range of 0 to 255.
- Byte 2 (Green): Represents the intensity of the green color channel, also in the range of 0 to 255.
- Byte 3 (Red): Represents the intensity of the red color channel, once again in the range of 0 to 255.

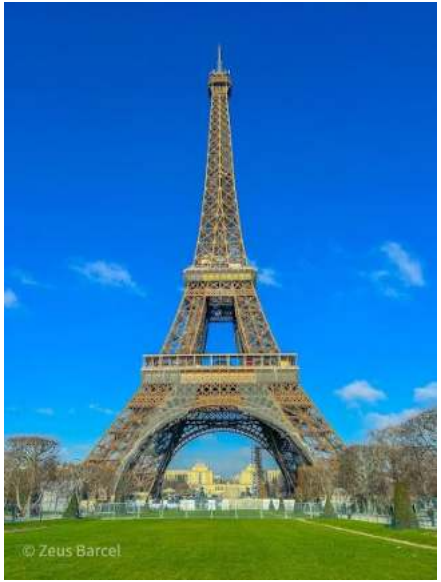
Brightened Output Image(+50%)

Here's how you can create a BufferedImage with this type:

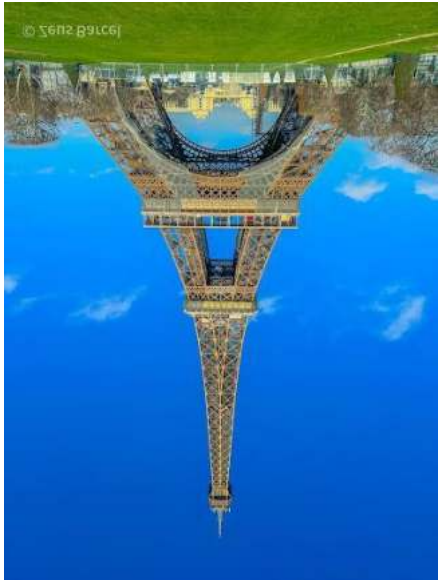
```
BufferedImage outputImage = new BufferedImage(width, height,BufferedImage.TYPE_3BYTE_BGR);
```

Upside-Down (Inversion)

This function, takes a BufferedImage as input and horizontally mirrors it by swapping pixel values from left to right. It achieves this by iterating through the input image's pixels, retrieving their RGB values, and **reversing the order of rows,and keeping columns constant** in the output image of the same dimensions and type (3-byte BGR).



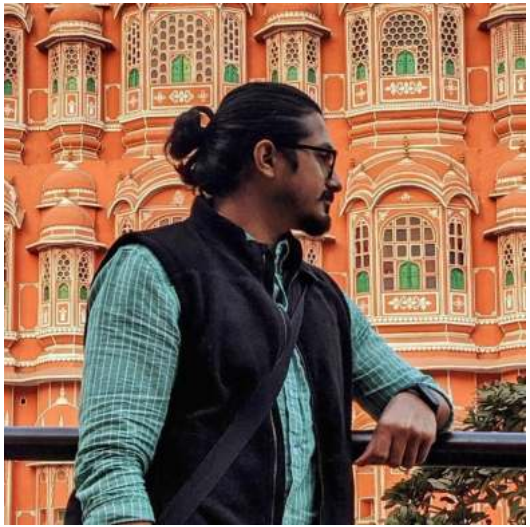
Input Image: eiffeltower.jpg



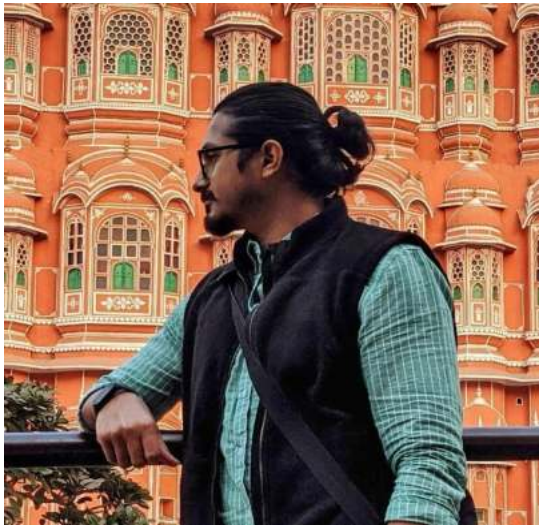
Output Image: inverted_eiffel_tower.jpg

Mirror Image

This method, mirrorImage, takes a BufferedImage as input and produces a horizontally mirrored version of the image. It achieves this by iterating through the input image's pixels, retrieving their RGB values, and **reversing the order of columns,and keeping rows constant in the output image** of the same dimensions and type (3-byte BGR), and returning a new BufferedImage with the mirrored content.



Input Image: kshitijsir.jpg



Output Image: mirrorimage.jpg

Rotate Clockwise

This code defines a method called `Rotate_Clockwise` that rotates an input image 90 degrees clockwise. It does this by creating a **new output image with swapped width and height dimensions**. Then, it iterates through each pixel of the input image, copying the pixel's color information to the corresponding position in the output image but with adjusted coordinates to achieve the clockwise rotation. Finally, it returns the rotated output image.

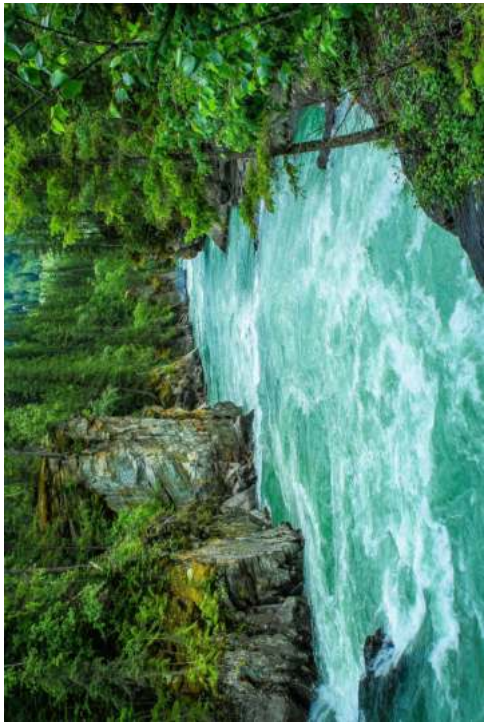
Method 2: This can also be done by taking the transpose of the image matrix & then swapping the columns, keeping rows constant.

Output Image: clockwise.jpg

Rotate Anti-Clockwise

The Rotate_AntiClockwise method exchanges the dimensions in an input image for a 90-degree anti-clockwise rotation. It swaps the width and height of the image and iterates through pixels to exchange their positions, creating the rotated output image.

We first takes the transpose of the image matrix along the secondary matrix & then swap the columns to rotate the image anti-clockwise.



Output Image: anticlockwise.jpg

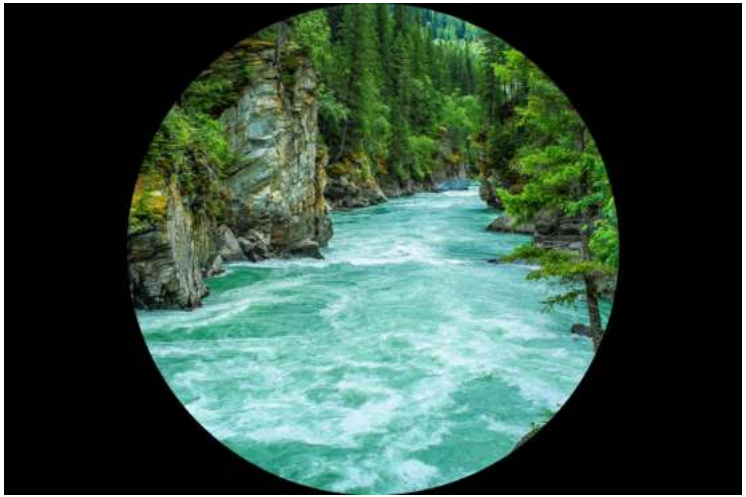
Blur

This Java code defines a blur method for image blurring. It takes an input image and a pixelCount parameter to specify the blurring window size. **The method calculates the average color values in small pixel windows and applies them to all pixels in each window, effectively blurring the image.** It starts by retrieving the input image's dimensions, creates an output image with the same size, and then iterates through the image using nested loops. Within each window, it calculates the average color and replaces the pixel colors with this average. The method returns the blurred output image, which can be controlled by adjusting the pixelCount.

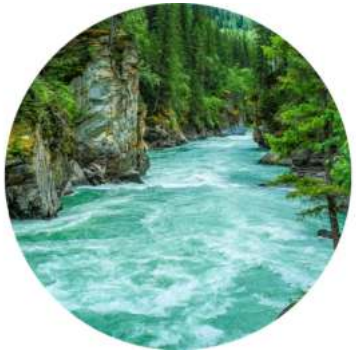
Output Image: blur25

Circle Crop

This function calculates the dimensions and radius of a circular crop within an input image. It then creates an output image with the same dimensions, calculates the center offsets, and iterates through the input image's pixels. **For each pixel, it checks if it falls within the circular region using a circle formula. If inside, the pixel's color is copied to the output image, effectively extracting the circular region.** Optionally, an else block can fill the outside with white for visual separation. Finally, the resulting BufferedImage, containing the circular crop, is returned as the output.



Output Image 1: circle_crop.jpg



Output Image 2: white_crop_circle.jpg

Conclusion

In conclusion, the "Image Editor" project is a Java-based image processing application that offers a range of essential image manipulation features. Through this project, we've successfully implemented functions for grayscale conversion, brightness adjustment, horizontal and vertical flipping, as well as clockwise and anti-clockwise rotation of images. Users can easily choose the desired operation and customize parameters when necessary, making this application user-friendly. This project provides a foundation for further image processing enhancements and serves as a valuable tool for basic image editing needs. It demonstrates the practical application of Java programming in image processing and lays the groundwork for future development in this domain.

Acknowledgment: I am deeply grateful for the invaluable help and unwavering support provided by my peers Vibhu Khullar, Aditya P Dash, and Kartik Deshpande throughout this project. Thank you for your collaboration.


Thank you