IMAGE MANIPULATION PROJECT DOCUMENTATION

A java project

GitHub Repository:

https://github.com/neeldholiya04/Image_Editor

A project by NEEL DHOLIYA

—>Table of Contents:--

1. ***INTRODUCTION:--***

This Java program is designed for image manipulation. It offers various image processing operations that can be applied to input images. These operations include converting the image to grayscale, changing its brightness, rotating it clockwise or counterclockwise, and flipping it horizontally or vertically.

2. ***PROGRAM STRUCTURE:--***

The program consists of a single Java class named **index**. Below are the key components and functionalities of the program:

A. Static Method:-

a. convertToGrayScale (BufferedImage inputImage)

- **Description:** Convert the input colour image to grayscale.
- **Parameters:**
  - **'inputImage'** (BufferedImage): The input image to be converted.
- **Returns:**
  - A grayscale **'BufferedImage'**.
- **Logic Explanation:**
  - This method takes an input image and creates a new BufferedImage of the same dimensions but with the type **'BufferedImage.TYPE_BYTE_GRAY'**, which represents grayscale images.
  - It then iterates through each pixel in the input image and sets the corresponding pixel in the output

grayscale image with the same grayscale value as the original pixel.

b. changeBrightness (BufferedImage inputImage, int increase)

- **Description:** Change the Brightness of the input colour image by a specified percentage.
- **Parameters:**
  - **'inputImage'** (BufferedImage): The input image to be modified.
  - **'increase'** (int): The percentage to increase brightness.
- **Returns:**
  - A modified **'BufferedImage'** with adjusted brightness.
- **Logic Explanation:**
  - This method takes an input image and a percentage increase value for brightness adjustment.
  - It iterates through each pixel in the input image.
  - For each pixel, it retrieves the red, green, and blue components and adjusts them based on the specified percentage increase.
  - It ensures that the adjusted values do not go below 0 or above 255 to maintain valid RGB values.
  - Finally, it creates a new pixel with the adjusted RGB values and sets it in the output image.

c. **rotateClockwise (BufferedImage inputImage)**

- **Description:** Rotate the input image 90 degrees clockwise
- **Parameters:**
  - **'inputImage'** (BufferedImage): The input image is to be rotated.
- **Returns:**
  - A clockwise rotated **'BufferedImage'**.
- **Logic Explanation:**
  - This method takes an input image and creates a new **'BufferedImage'** with dimensions swapped (width becomes height, and vice versa) to accommodate the rotated image.
  - It iterates through each pixel in the input image and sets the corresponding pixel in the output image by performing a clockwise rotation.
  - The clockwise rotation logic involves exchanging rows with columns and inverting the row index.

d. **rotateAntiClockwise (BufferedImage inputImage)**

- **Description:** Rotate the input image 90 degrees anticlockwise.
- **Parameters:**
  - **'inputImage'** (BufferedImage): The input image is to be rotated.
- **Returns:**
  - An anticlockwise rotated **'BufferedImage'**.

- **Logic Explanation:**
  - ❖ This method takes an input image and creates a new **'BufferedImage'** with dimensions swapped to accommodate the rotated image.
  - ❖ It iterates through each pixel in the input image and sets the corresponding pixel in the output image by performing a counterclockwise rotation.
  - ❖ The counterclockwise rotation logic involves exchanging columns with rows and inverting the column index.

e. **flipHorizontal (BufferedImage inputImage)**

- **Description:** Flips the input image horizontally.
- **Parameters:**
  - ❖ **'inputImage'** (BufferedImage): The input image is to be flipped.
- **Returns:**
  - ❖ A horizontally flipped **'BufferedImage'**.
- **Logic Explanation:**
  - ❖ This method takes an input image and creates a new **'BufferedImage'** with the same dimensions.
  - ❖ It iterates through each pixel in the input image and sets the corresponding pixel in the output image by flipping it horizontally.

- ❖ The horizontal flip logic involves reversing the order of columns.

f. <u>flipVertically (BufferedImage inputImage)</u>

- **Description:** Flips the input image vertically.
- **Parameters:**
  - ❖ **'inputImage'** (BufferedImage): The input image is to be flipped.
- **Returns:**
  - ❖ A vertically flipped **'BufferedImage'.**
- **Logic Explanation:**
  - ❖ This method takes an input image and creates a new **'BufferedImage'** with the same dimensions.
  - ❖ It iterates through each pixel in the input image and sets the corresponding pixel in the output image by flipping it vertically.
  - ❖ The vertical flip logic involves reversing the order of rows.

B. <u>Main Method:-</u>

a. The 'main' method serves as the entry point of the program.

b. It allows the user to select an image manipulation option via the command line and provides interactive prompts for additional input, such as brightness adjustment percentage.

c. Supported Operations include:

- Convert to Grayscale.
- Change Brightness
- Rotate Clockwise
- Rotate Anticlockwise
- Flip Horizontal
- Flip Vertically

d. The modified image is saved as "OutputImage.jpg" in the same directory as the program.

3. ***SUPPORTED OPERATIONS:--***

The program supports the following image manipulation operations:

a. Convert to GrayScale:

- **Description:** Convert the input colour image to grayscale

INPUT IMAGE

OUTPUT IMAGE

b. Change Brightness:

- **Description:** Change the brightness of the input colour image by a specified percentage.
- **Additional Input required:** Percentage adjustment.

INPUT IMAGE

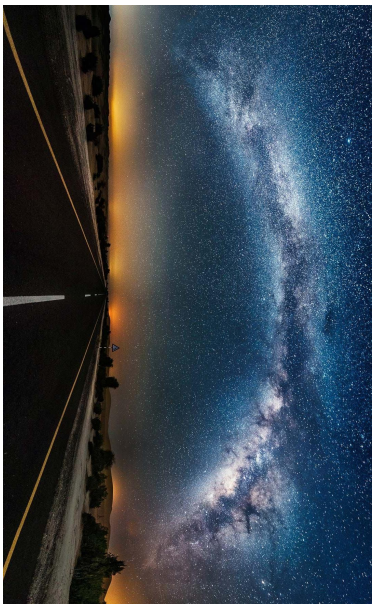
OUTPUT IMAGE

BRIGHTNESS INCREASED BY 40%

c. Rotate Clockwise:

- **Description:** Rotate the input image 90 degrees clockwise.

INPUT IMAGE

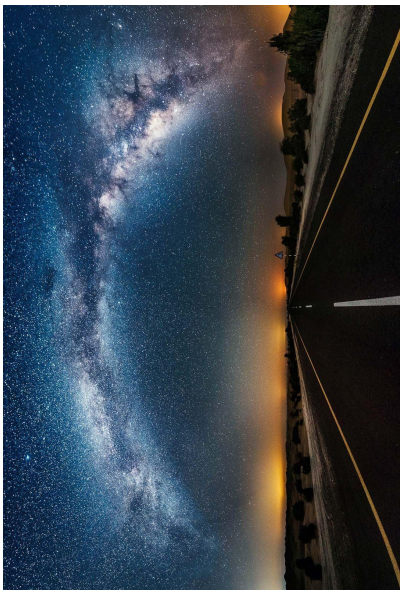


OUTPUT IMAGE

d. Rotate Anticlockwise:

- **Description:** Rotate the input image 90 degrees anticlockwise.

INPUT IMAGE

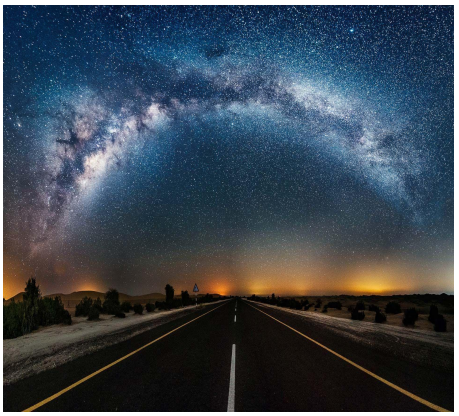


OUTPUT IMAGE

e. Flip horizontally:

- **Description:** Flips the image horizontally (left to right).

INPUT IMAGE

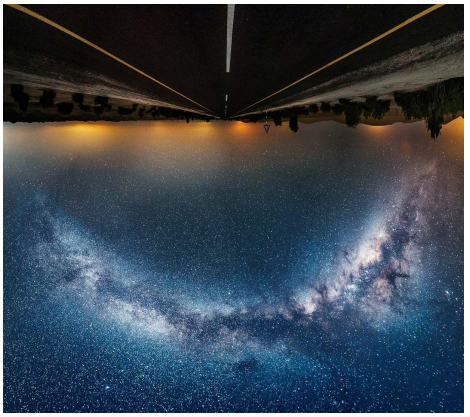


OUTPUT IMAGE

f. Flip vertically:

- **Description:** Flips the image vertically (top to bottom).

INPUT IMAGE



OUTPUT IMAGE

4. *USAGE INSTRUCTIONS:--*

a. Compile and run the image:

- Compile the program using **'javac index.java'**.
- Run the program using **'java index'**.

b. Select an operation:

- Choose one of the image manipulation options by entering the corresponding option.

c. Provide Additional Input:

- Follow the prompts to provide additional input, such as the brightness adjustment percentage.

d. View the modified Image:

- The modified image will be saved as "OutputImage.jpg" in the program's directory.

5. *DEPENDENCIES:--*

- The program uses the **'java.awt.Color'** class and **'java.awt.image'**.BufferedImage for image manipulation.

- It also imports classes from the **'javax.imageio'** package for reading and writing images.

6. ***SAMPLE USAGE:--***

- To convert an image to grayscale, enter '1'.
- To change the brightness, enter '2', and provide the percentage adjustment.
- To rotate 90 degrees clockwise, enter '3'.
- To rotate 90 degrees anticlockwise, enter '4'.
- To flip horizontally, enter '5'.
- To flip vertically, enter '6'.

7. ***NOTES:--***

- Ensure that the input image file is named "43911.jpg" and is located in the same directory as the program for the operations to work.
- The program provides a simple command-line interface for performing various image manipulation operations, making it user-friendly and accessible for basic image processing tasks.