Report
2 September 2023

Java Image Editor

Guided By: Kshitij Mishra

Made By: Krishna Patidar

Index

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.

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:-

1. Convert To GrayScale (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.

Input

Output

2. Increase Brightness (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.

Input

Output

3. Rotate Clockwise (BufferedImage inputImage)

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

- **Parameters** :

- *'inputImage'* (BufferedImage) : The input image 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.

Input

Output

4. Rotate Anti-Clockwise (BufferedImage inputImage)

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

- **Parameters** :

  - *‘inputImage’* (BufferedImage) : The input image to be Rotated.

- **Returns :**

  - A clockwise 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.

Input

Output

5. Blur The Image (BufferedImage inputImage, int density)

- **Description** : Convert the input image to blur image.

- **Parameters** :

  - *‘inputImage’* (BufferedImage) : The input image to be blurred.

  - ‘Density’ (int) : value that controls the level of blurring.

- **Returns :**
  - A blurred `*BufferedImage*`.
- **Logic Explanation** :
  - The code uses two nested loops to traverse the input image. The outer loop iterates over rows (height), and the inner loop iterates over columns (width).
  - The *density* parameter controls the step size for these loops, determining the size of the matrix used for blurring. A higher *density* value results in a coarser blur.
  - For each position `(i, j)` in the input image, a smaller matrix (block) is selected, starting from `(i, j)` and extending `density` pixels in both the vertical and horizontal directions.
  - Within this matrix, the code calculates the average red, green, and blue (RGB) values for all the pixels in the matrix.
  - Once the average RGB values are computed, they are assigned to all the pixels in the selected matrix.

Input

Output

6. Inverse vertically (BufferedImage inputImage)

- **Description** : Flips the input image vertically.
- **Parameters** :
  - *'inputImage'* (BufferedImage) : The input image 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.

Input

Output

B. Main Method:-

1. The ‘main’ method serves as the entry point of the program.

2. 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.

3. Functions available in this Editor are :-

   - **Convert To Grayscale**
   - **Change Brightness**
   - **Rotate Clockwise**
   - **Rotate Anti-Clockwise**
   - **Blur The Image**
   - **Inverse Vertically**

4. The modified image is saved as “OutputImage.jpg” in the same directory as the program.

Dependencies:—

1. **Java AWT :-** Java AWT (Abstract Window Toolkit) is used for handling images and colors.

2. **Java IO :-** Java IO is used for file input and output operations.

3. **Java Util :-** java.util.Scanner is used to take input from the user through the console.

4. **JavaX ImageIO:-** javax.imageio.ImageIO is used to read and write images in various formats.

Usage Instruction:—

1. **Compile and run the image: :-**
    - Compile the program using ‘javac index.java’.
    - Run the program using ‘java index’.
2. **Select an operation :-**
    - Choose one of the image manipulation options by entering the corresponding option.
3. **Provide Additional Input :-**
    - Follow the prompts to provide additional input, such as the brightness adjustment percentage.
4. **View the modified Image :-**
    - The modified image will be saved as "OutputImage.jpg" in the program's directory.

Show Stopper:—

1. Ensure that the input image file is named "image.jpg" and is located in the same directory as the program for the operations to work.

2. 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.

Thankyou!!