

Image Editor

01.09.2023

Sandip Dey

Scaler School Of Technology

Overview

This is an Image editor build totally on java programming language. This image editor provide various features to user to take an image and apply different features and convert the image to desired image. This image editor provide various features to user to apply.

The features includes :

1. Convert to GrayScale.
2. Change brightness.
3. Flip Horizontally.
4. Flip vertically.
5. Transpose.
6. Rotate Clockwise.
7. Rotate Anti-clockwise.
8. Pixelete.

This is an Command line based image editor.

- Usage Instruction

The user will be directed to a terminal where some commands will appear that will tell user to choose some options from 1 to 9 which have different features and user will be told to give the path of the file where the original image is present. After that user will be told to choose some option from above given options , after choosing one options output image will be generated with the specified feature applied and this will continue to ask user path and operations until user stop the program.

To exit the program user will have to choose 9 as will be shown in the options.


```
Enter 1 to convert to grayscale :  
Enter 2 to increase brightness :  
Enter 3 to flip the image horizontally :  
Enter 4 to flip the image vertically :  
Enter 5 to convert to transpose :  
Enter 6 to rotate the image clockwise :  
Enter 7 to rotate the image Anti-clockwise :  
Enter 8 to pixelete the image :  
Enter 9 to exit :  
Enter the path of the image:  
/home/sandip/Pictures/projectImage/image.jpg  
Enter the choice : 7  
Anti-clockwise Rotated image created as outputImage.jpg  
Enter the path of the image:  
□
```

-Project Structure

The project is totally based on java programming languages.

The image editor will take input an image from user's local storage. The program will take input the image file in a file object created as inputFile . Then that file will be converted to buffered Image and will be stored in an Buffered Image object created named image and an outputImage bufferedimage will also be created initialized with NULL .

Then after user chooses the option to apply a features the image will be directed to the specified function and inside that function all the functionalities will be applied to the current image and the new BufferedImage newImage will be created and will be returned from the function with applied the specific feature.



After that the returned image will be stored in `outputImage` and will be converted to `outputImage.jpg` and will be stored in user's local storage .

And also a message will be displaying that the image with applied feature has been generated as `outputImage.jpg` in the terminal.

And will also ask user for another path as input and the loop will run until user end the program.

-Deep dive into the Project

-Imported Libraries



To manipulate the input image and to apply different features starting from taking input the file from local storage , converting the file to buffered image , handling different errors related to file operations , image writing from buffered image and several different functions we needed to import the above given libraries into our program.

Supported operations

-Convert to grayScale



```
public static BufferedImage
convertToGrayScale(BufferedImage inputImage) {
    int height = inputImage.getHeight();
    int width = inputImage.getWidth();
    BufferedImage outputImage = new
    BufferedImage(width, height, BufferedImage.TYPE_BYTE_GRAY);
    for (int i=0; i<height; i++) {
        for (int j =0 ; j<width; j++) {
            outputImage.setRGB(j, i,
            inputImage.getRGB(j, i));
        }
    }
    return outputImage;
}
```

This function takes input a Buffered image inputImage and make an buffered image newImage.

The function go to every pixel of the input buffered image , takes the pixel of every pixel of the image and set the pixel value of the created newImage and store it

in one byte value only consisting the value of white intensity. By this way this function create an gray scale image of the passed as the input to the function.

Visual representation :



Original image



GrayScale image

-Change brightness

```
Adjust Brightness

public static int truncateBrightness(int color) {
    if (color < 0) return 0;
    if (color > 255) return 255;
    return color;
}

public static BufferedImage
increaseBrightness(BufferedImage inputImage, int percent){
    int width = inputImage.getWidth();
    int height = inputImage.getHeight();
    BufferedImage outputImage = new
    BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
    for (int i=0; i<height ; i++) {
        for (int j =0; j<width; j++) {
            Color pixel = new
            Color(inputImage.getRGB(j, i));

            int red = pixel.getRed();
            red = (int)(red*(1+percent/100d));
            red = truncateBrightness(red);
            int green = pixel.getGreen();
            green = (int)(green*(1+percent/100d));
            green = truncateBrightness(green);
            int blue = pixel.getBlue();
            blue = (int)(blue*(1+percent/100d));
            blue = truncateBrightness(blue);

            outputImage.setRGB(j, i, ((red << 16) |
            (green << 8) | blue));
        }
    }
    return outputImage;
}
```



This function takes input a Buffered image named `inputImage` and a integer value `percentage` .

Then it create a Buffered Image named `newImage`.

Then inside the function it go to every pixel of the image and take the red, green and blue values of that picture and changes the respective values of the pixel also to handle the edge case another function `truncate` brightness has been created which will truncate the red , green and blue values of the pixel and if it is out of 4 byte integer range i.e, 0 to 255.

After getting the modified values of the pixels it will put the modified pixel values to every pixels of the Buffered image created in the function named `newImage`.

After iterating to all the pixels of the input Image and storing all the pixel values to the new image , the new image will be generated with the changed brightness.

Then after generating the `newImage` the function will return the generated bufferedImage `newImage`

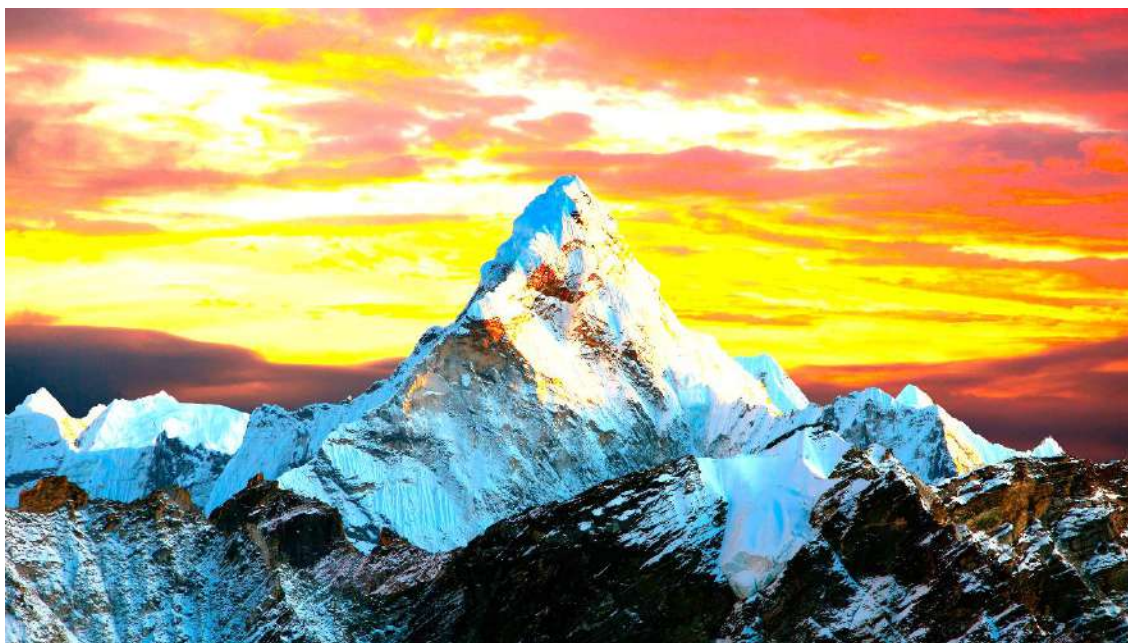
The returned image will be stored Buffered image `outputImage` and then it will be written in the `outputimage.jpg`

After that the message will be displayed "Brightness adjusted image created as outputimage.jpg".

Visual representation :



Original image



Brightness increased image



Brightness decreased image

-Flip Horizontally

```
Flip Horizontally

public static BufferedImage mirrorImage(BufferedImage
inputImage) {
    int width = inputImage.getWidth();
    int height = inputImage.getHeight();
    BufferedImage outputImage = new
BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);

    for (int i=0; i<height; i++) {
        for (int j=0; j<width; j++) {
            outputImage.setRGB(width-j-1, i,
inputImage.getRGB(j, i));
        }
    }
    return outputImage;
}
```

This function will take an Buffered image `inputImage` As input and will create a buffered image `outputImage` inside the function.

Then the function will go to every pixel and store the value of the desired pixel of the `inputImage` and store it into some desired pixel of the `outputImage` so that the output image can be created the horizontally flipped of the original image.

After that it will return the `outputImage` .

Visual representation :



Original Image



Horizontally flipped Image

-Flip Vertically

```
Flip Vertically

public static BufferedImage flipVertically(BufferedImage
inputImage) {
    int width = inputImage.getWidth();
    int height = inputImage.getHeight();
    BufferedImage outputImage = new BufferedImage(width
, height, BufferedImage.TYPE_INT_RGB );
    for (int i=0; i<height ; i++) {
        for (int j =0; j<width; j++) {
            outputImage.setRGB(j, height-i-1,
inputImage.getRGB(j, i));
        }
    }
    return outputImage;
}
```


This function will take an Buffered image inputImage As input and will create a buffered image outputImage inside the function.

Then the function will go to every pixel and store the value of the desired pixel of the inputImage and store it into some desired pixel of the outputImage so that the output image can be created the vertically flipped of the original image.

After that it will return the outputImage .

Visual representation :



Original Image



Vertically Flipped Image

-Transpose

```
public static BufferedImage transposeImage(BufferedImage
inputImage) {
    int width = inputImage.getWidth();
    int height = inputImage.getHeight();
    BufferedImage outputImage = new
    BufferedImage(height, width, BufferedImage.TYPE_INT_RGB);

    for (int i=0; i<height; i++) {
        for (int j=0; j<width; j++) {
            outputImage.setRGB(i, j,
inputImage.getRGB(j, i));
        }
    }
    return outputImage;
}
```


This function will take an Buffered image inputImage As input and will create a buffered image outputImage inside the function.

Then the function will go to every pixel and store the value of every pixel of the inputImage and store it into transpose pixel of the outputImage so that the output image can be created the vertically flipped of the original image.

After that it will return the outputImage .

Visual representation :



Original Image



Transposed Image

-Rotate Clockwise

```
Rotate Clockwise

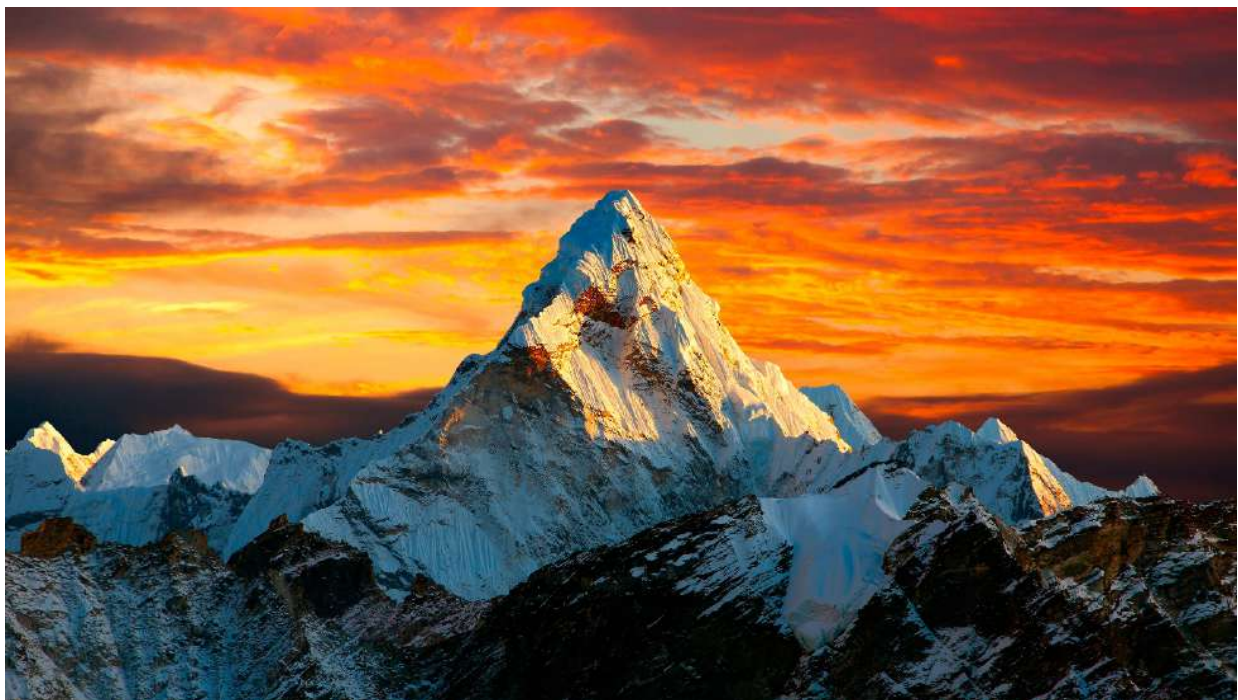
public static BufferedImage rotateClockwise(BufferedImage
inputImage) {
    int width = inputImage.getWidth();
    int height = inputImage.getHeight();
    BufferedImage outputImage = new
BufferedImage(height, width, BufferedImage.TYPE_INT_RGB);
    outputImage =
mirrorImage(transposeImage(inputImage));
    return outputImage;
}
```

This Function will take input a BufferedImage inputImage and will create an buffered image outputImage inside the function.

Then it will store the mirror image of the transposed image of the input image so as to get the Clockwise rotated image .

Then this will return the output image.

Visual representation :



Original Image



Clockwise Rotated Image

-Rotate Anti-Clockwise

```
Rotate Anti-Clockwise

static BufferedImage rotateAntiClockwise(BufferedImage
image){
    int width = image.getWidth();
    int height = image.getHeight();
    BufferedImage outputImage = new
    BufferedImage(height, width, BufferedImage.TYPE_INT_RGB);

    for(int i=0; i<height; i++){
        for(int j=0; j<width; j++){
            outputImage.setRGB(i, j,
            image.getRGB(width-j-1, i));
        }
    }
    return outputImage;
}
```


This function will take an Buffered image inputImage As input and will create a buffered image outputImage inside the function.

Then the function will go to every pixel and store the value of every pixel of the inputImage and store it into desired pixel of the outputImage so that the output image can be created the anti-clockwise rotated of the original image.

After that it will return the outputImage .

Visual representation :



Original Image



Anti-Clockwise Rotated Image

-Pixelate :

This function takes input a buffered image `inputImage` and integer value `val` (0 to 100) which specifies the intensity of blurriness as input and create a bufferedImage `outputImage`.

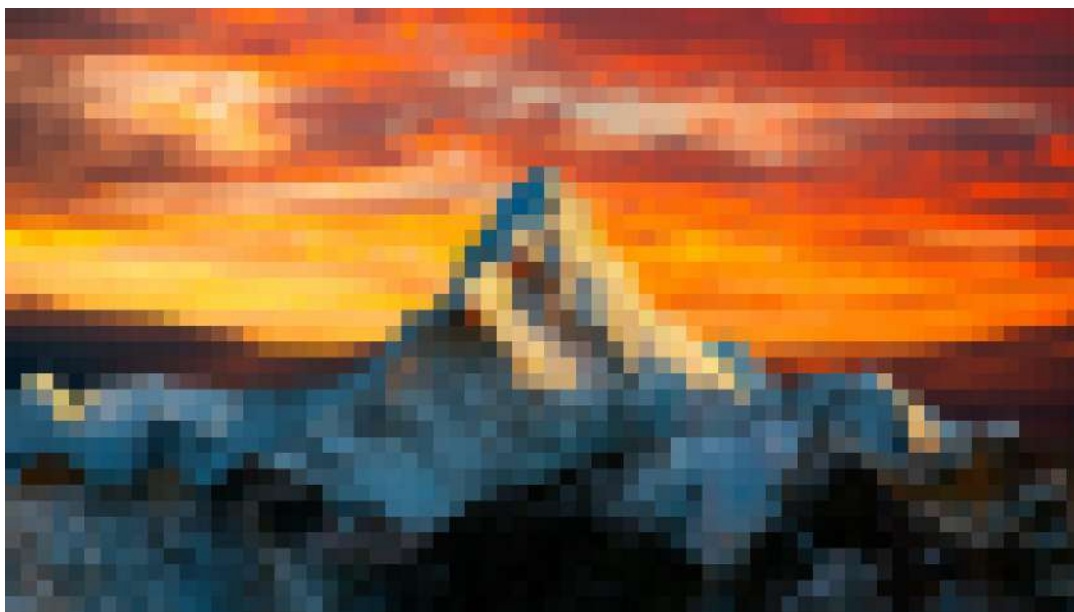
Then in this function we go to every pixel of the input image and take the average value of a square of (`val x val`) and applying the average value of all the red, green and blue to every pixel in the `outputImage`.

Then after creating the `outputImage` the function will return the `outputImage`.

Visual representation



Original Image



Pixelated Image

Summary

This project aims to create a command line user interface which can make a user able to apply different operations on their image and get their desired image created with desired properties.

This can make user to manipulate their existing image and apply various operation to get a better image.

Thank You