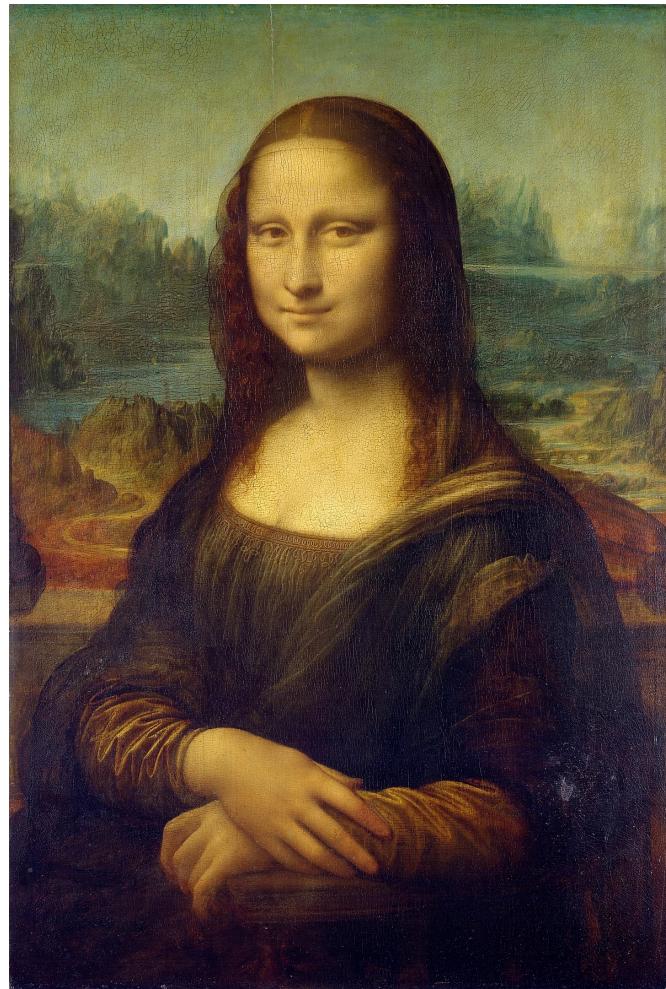


# Scaler School Of Technology

*Project -1 Image Editor in JAVA*



**Shreyas Garg**

2nd Sep, 2023  
23bcs

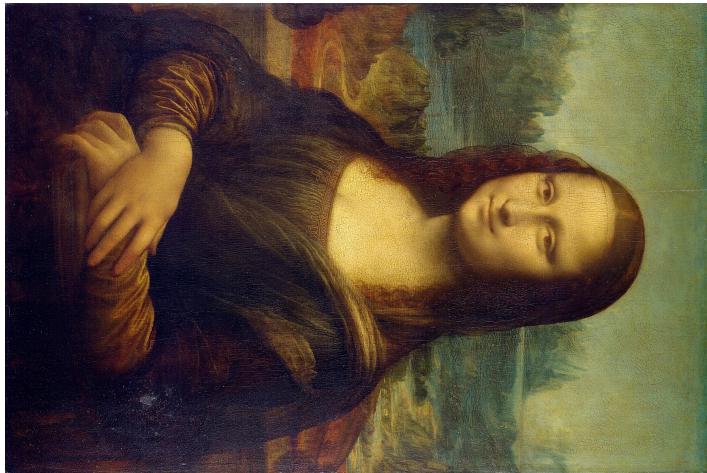
## INTRODUCTION

This image editor allows you to manipulate images. It provides several features such as converting the image to grayscale, rotating it, mirroring it and adjusting its brightness.

## FEATURES

- Convert image to grayscale
- Rotate image clockwise or anticlockwise
- Adjust brightness of the given image
- Create mirror image of the original

## THE CODE AND HOW IT WORKS



First feature: Rotate clockwise- In this we take the transpose of the given image and rotate it by 90 degrees in clockwise direction.

```
rotate clockwise
ic BufferedImage rotateClockwise(BufferedImage inputImage){
    int height = inputImage.getHeight();
    int width = inputImage.getWidth();

    BufferedImage outputImage = new BufferedImage(height, width, BufferedImage.TYPE_INT_RGB);

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

    outputImage = mirror(outputImage);

    return outputImage;
}
```



Second feature: Rotate anticlockwise- In this we take the rotate clockwise command three times and rotate it anticlockwise

```
//2 rotate anticlockwise
static BufferedImage rotateAntiClockwise(BufferedImage inputImage){
    BufferedImage outputImage = rotateClockwise(inputImage);
    outputImage = rotateClockwise(outputImage);
    outputImage = rotateClockwise(outputImage);
    return outputImage;
}
```

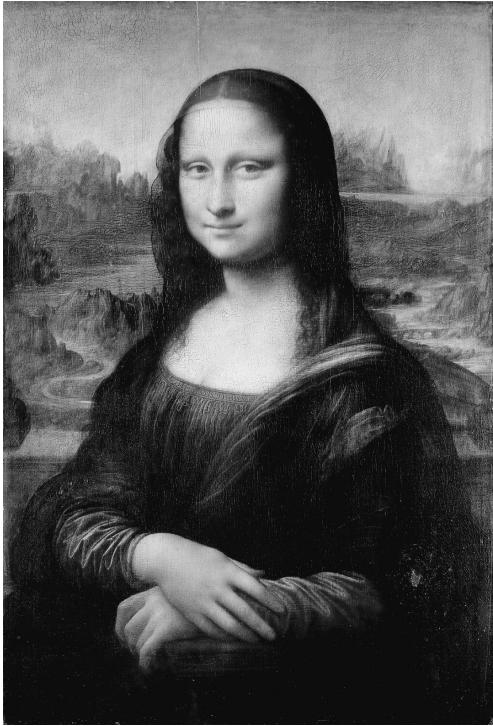


Third feature: Mirror the image- In this we exchange the colors of the i-th column and (total-1-i)th column.

```
//3 mirror
static BufferedImage mirror(BufferedImage inputImage){
    int height = inputImage.getHeight();
    int width = inputImage.getWidth();

    BufferedImage outputImage = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);

    for(int i = 0; i < height; i++){
        for(int j = 0; j < width/2; j++){
            Color pixel = new Color(inputImage.getRGB(j, i));
            outputImage.setRGB(j,i,inputImage.getRGB(inputImage.getWidth()-1-j , i));
            outputImage.setRGB(inputImage.getWidth()-1-j , i , pixel.getRGB());
        }
    }
    return outputImage;
}
```



Fourth feature: Convert image to grayscale:  
The `TYPE_BYTE_GRAY` specifies the output image to be grayscale using a set of different shades of gray.

```
//4 greyscale
static BufferedImage convertToGreyScale(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;
}
```



Fifth feature: Change the brightness of the image:  
Taking RGB values from each pixel and increase them by the given percent, if final value>250 set it to 250 and if value<0 then set it to 0, after this we store the new value of each pixel

```
//5 change brightness
static BufferedImage changeBrightness(BufferedImage inputImage , int increase){
    int height = inputImage.getHeight();
    int width = inputImage.getWidth();

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

    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();
            int blue = pixel.getBlue();
            int green = pixel.getGreen();

            red = red + (increase*red/100);
            blue = blue + (increase*blue/100);
            green = green + (increase*green/100);

            if(red > 255){red = 255;}
            if(blue > 255){ blue = 255;}
            if(green > 255){ green = 255;}

            if(red < 0){ red = 0;}
            if(blue < 0){blue = 0;}
            if(green < 0){ green = 0; }

            Color newPixel = new Color(red , green , blue);

            outputImage.setRGB(j,i,newPixel.getRGB());
        }
    }
}
```



Sixth feature:blur the image:

This takes the input for the length of the square that is to be blurred then the average of all pixels within each square is taken to create a blurred pixelated image.

## Thank You!

A very special thank you to my mentors, teachers and peers for making this wonderful project possible.