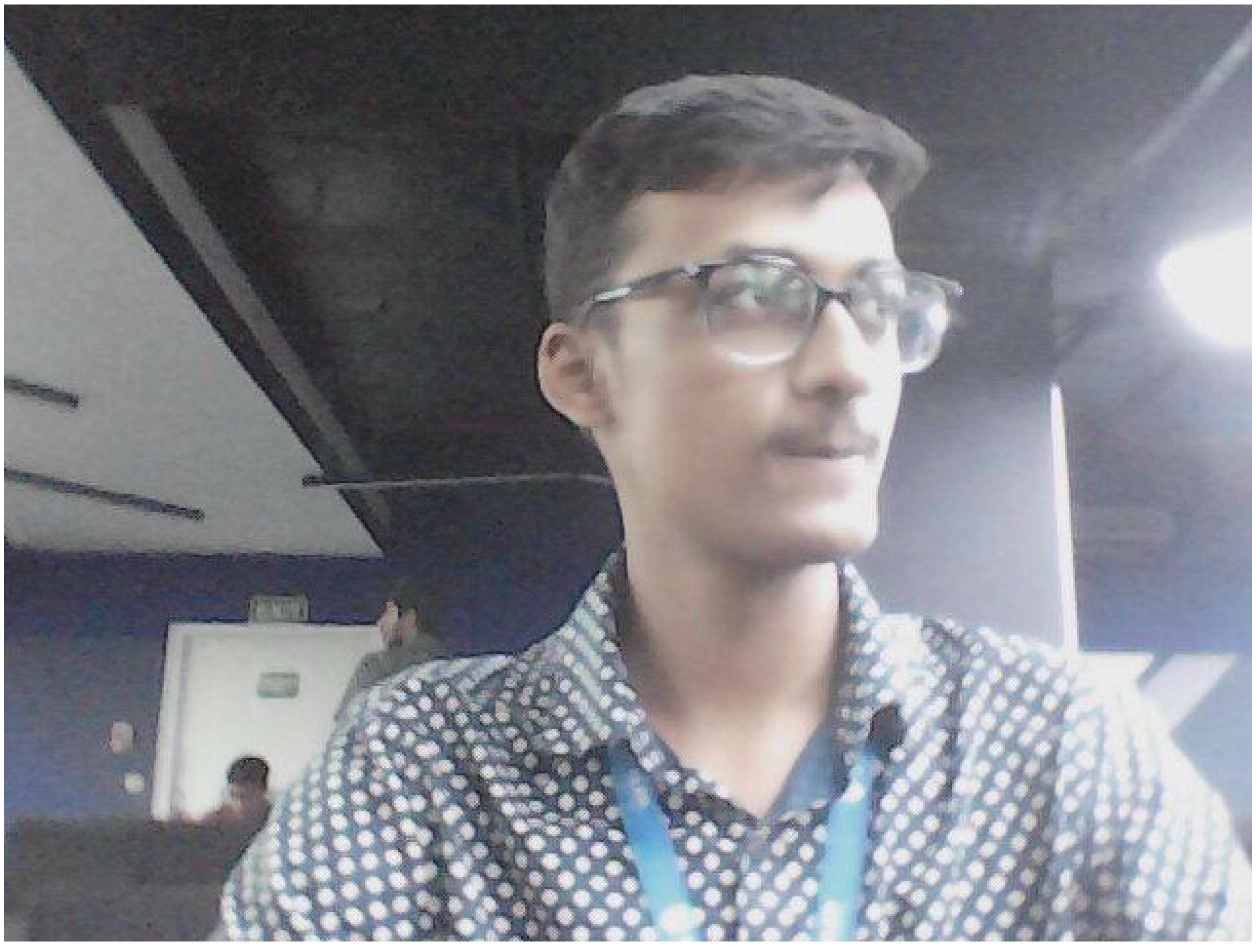
ORIGINAL IMAGE



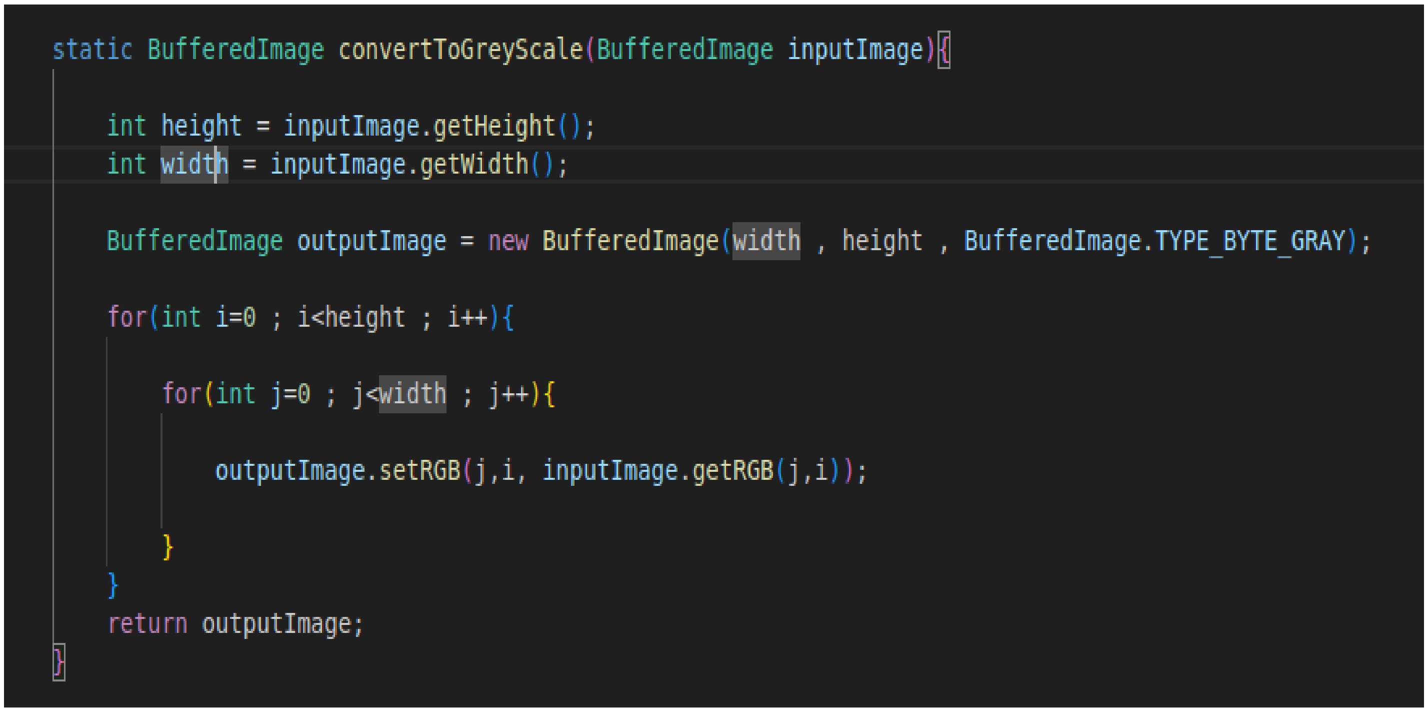
FEATURES ADDED:

1)CONVERT TO GRAY-SCALE
2)CHANGE BRIGHTNESS
3)MIRROR
4)ROTATE CLOCKWISE
5)ROTATE ANTI-CLOCKWISE
6)BLUR
7)CONTRAST
8)CROP TO CIRCLE

9)EDGE-DETECTION
10)INVERSION
11)PENCIL SKETCH

Link to github:
https://github.com/androemeda/image-editor

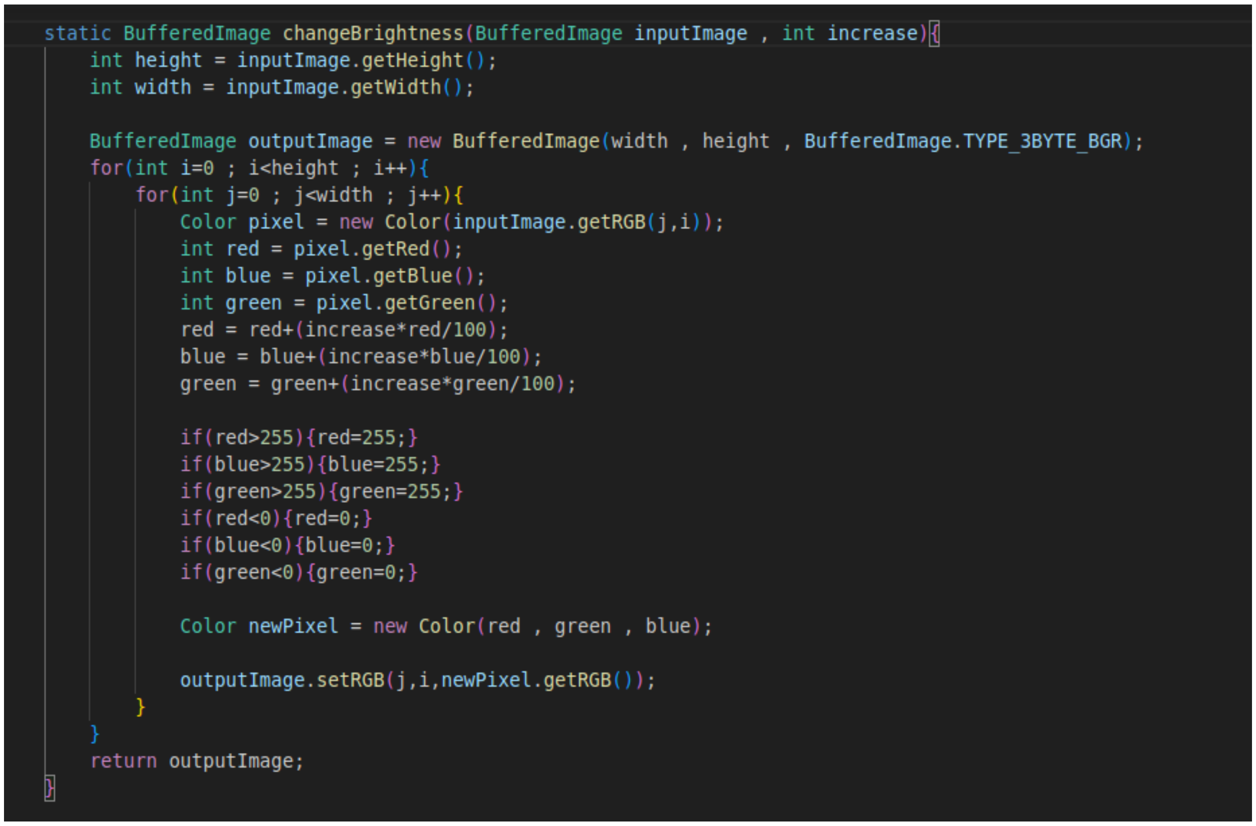
1)CONVERT TO GRAY-SCALE



```java
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;
}
```

The `TYPE_BYTE_GRAY` type specifies that the output image will be in grayscale format, using 8 bits per pixel to represent different shades of gray.

2)CHANGE BRIGHTNESS

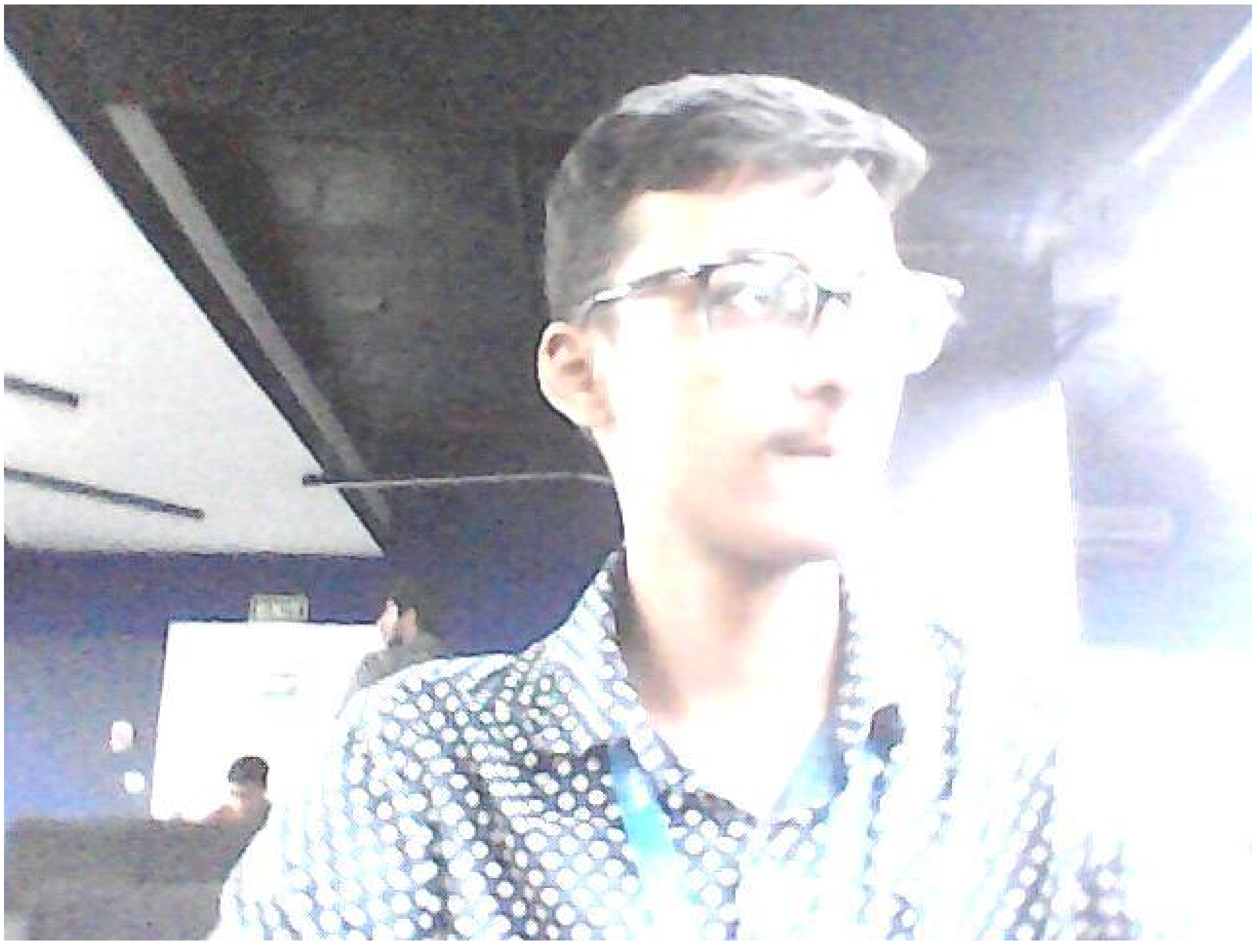


```java
static BufferedImage changeBrightness(BufferedImage inputImage , int increase){
    int height = inputImage.getHeight();
    int width = inputImage.getWidth();

    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());
        }
    }
    return outputImage;
}
```

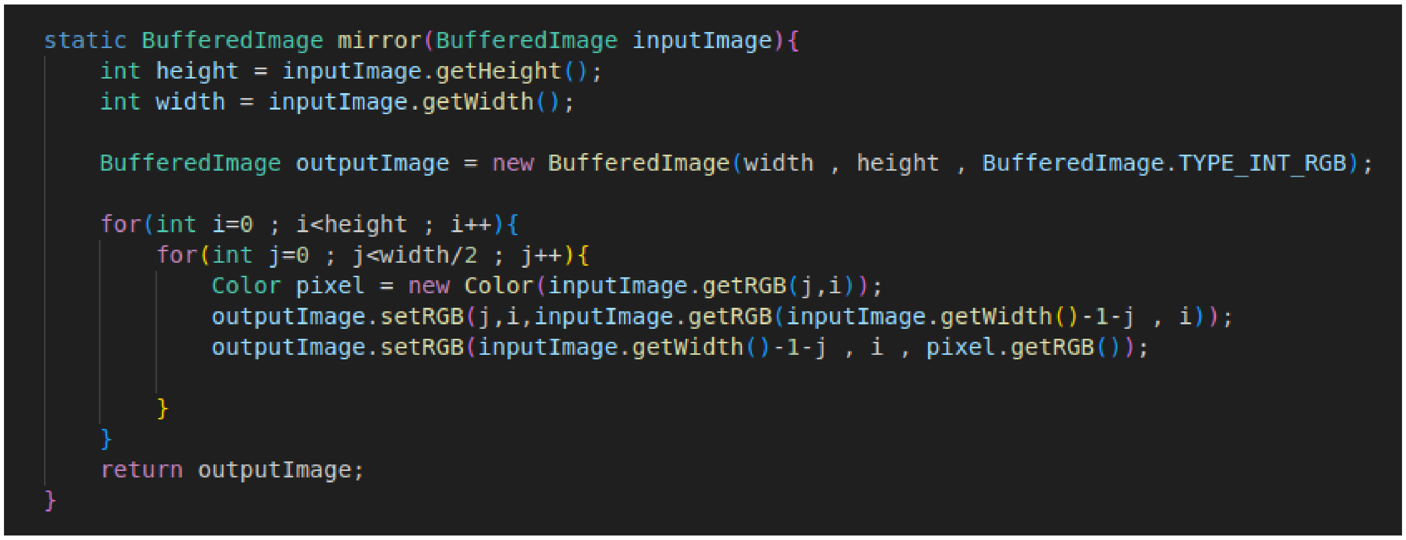
Taking 2 parameters - input image and percentage by which brightness is to be changed.

Take R,G and B values from each pixel and increase them by percentage.

If final value is greater than 250 , set it to 250.
Percentage can be -ve. So if final value is less than zero , set it to zero.
Store the new values in every pixel.



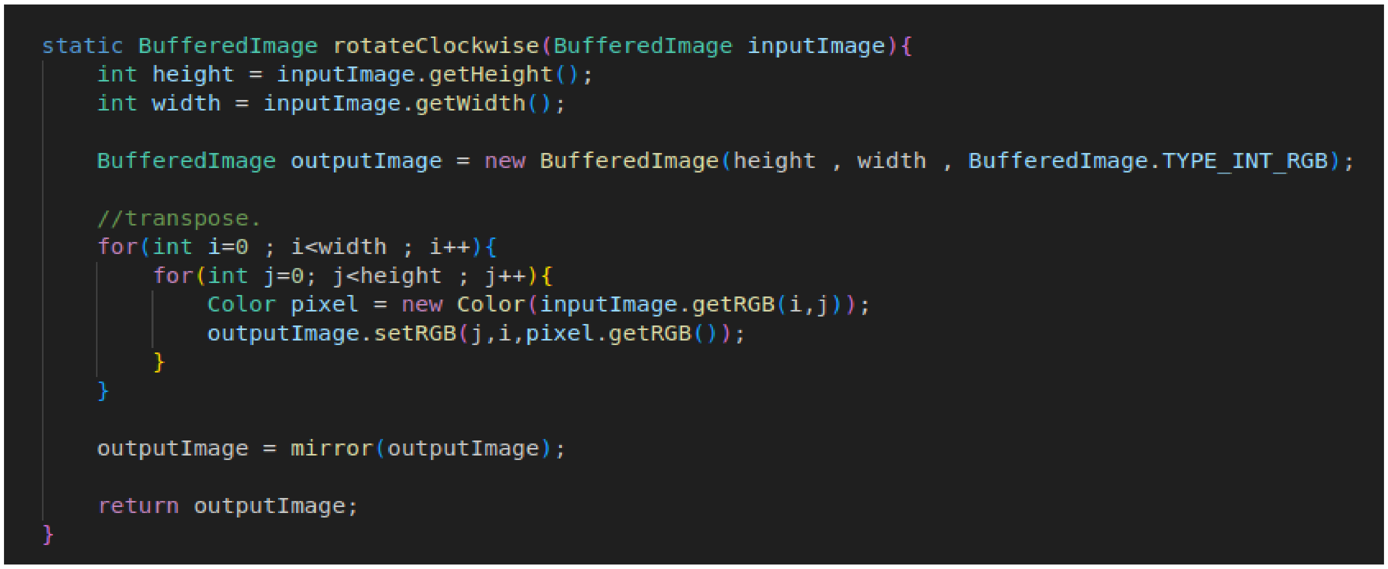
3)MIRROR



```
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;
}
```

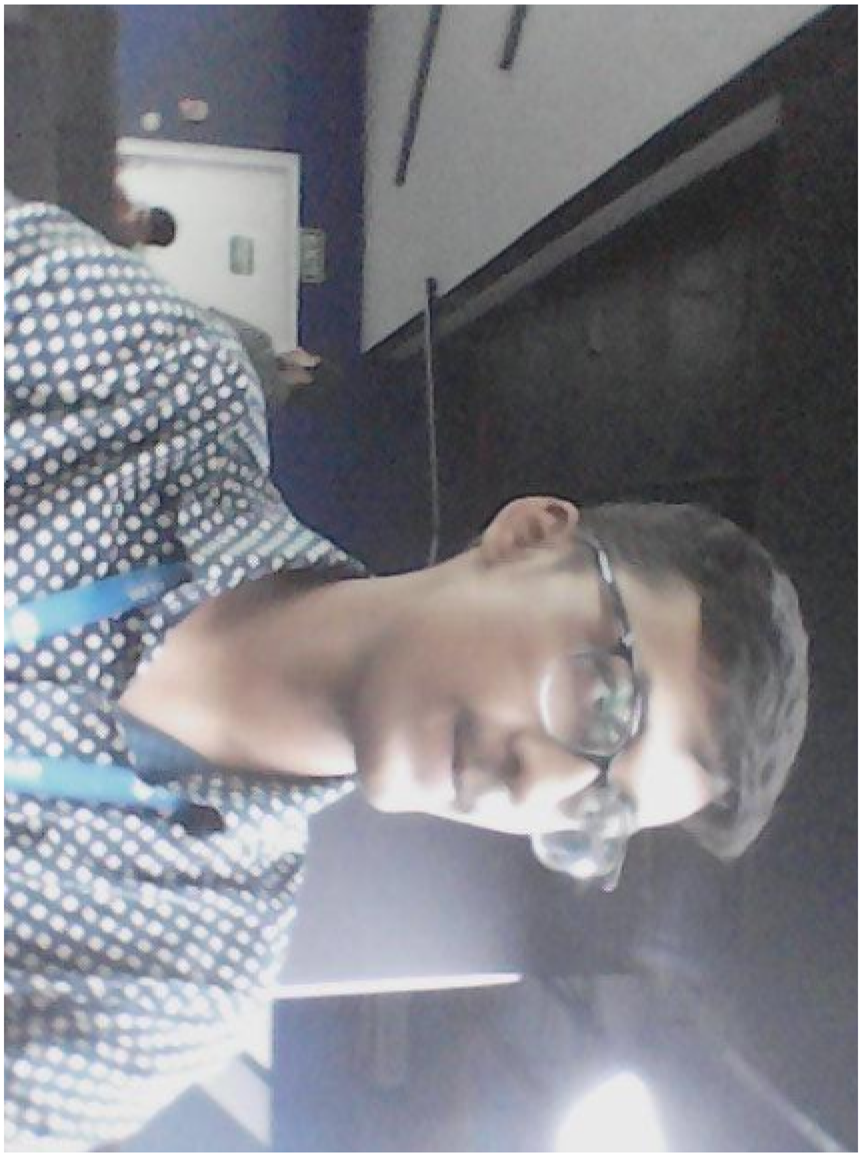
To create mirror image , exchanged the colors of ith column and (total-1-i)th column.

4)ROTATE CLOCKWISE

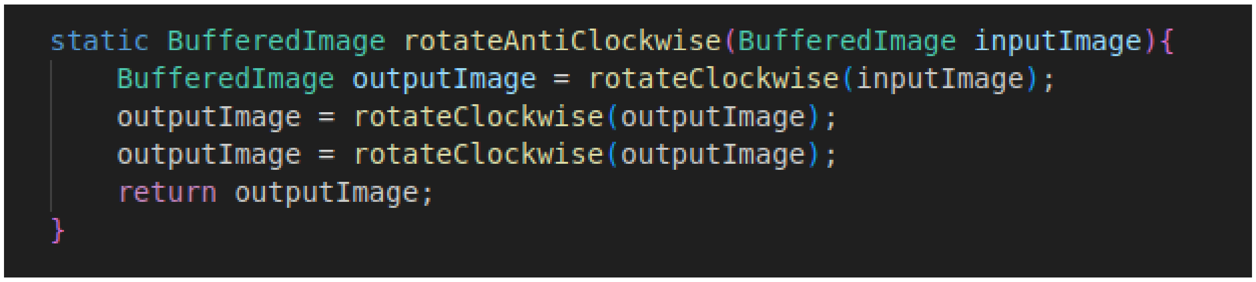


```java
static BufferedImage rotateClockwise(BufferedImage inputImage){
    int height = inputImage.getHeight();
    int width = inputImage.getWidth();

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

    //transpose.
    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;
}
```

First take transpose of the image , then take its mirror image.

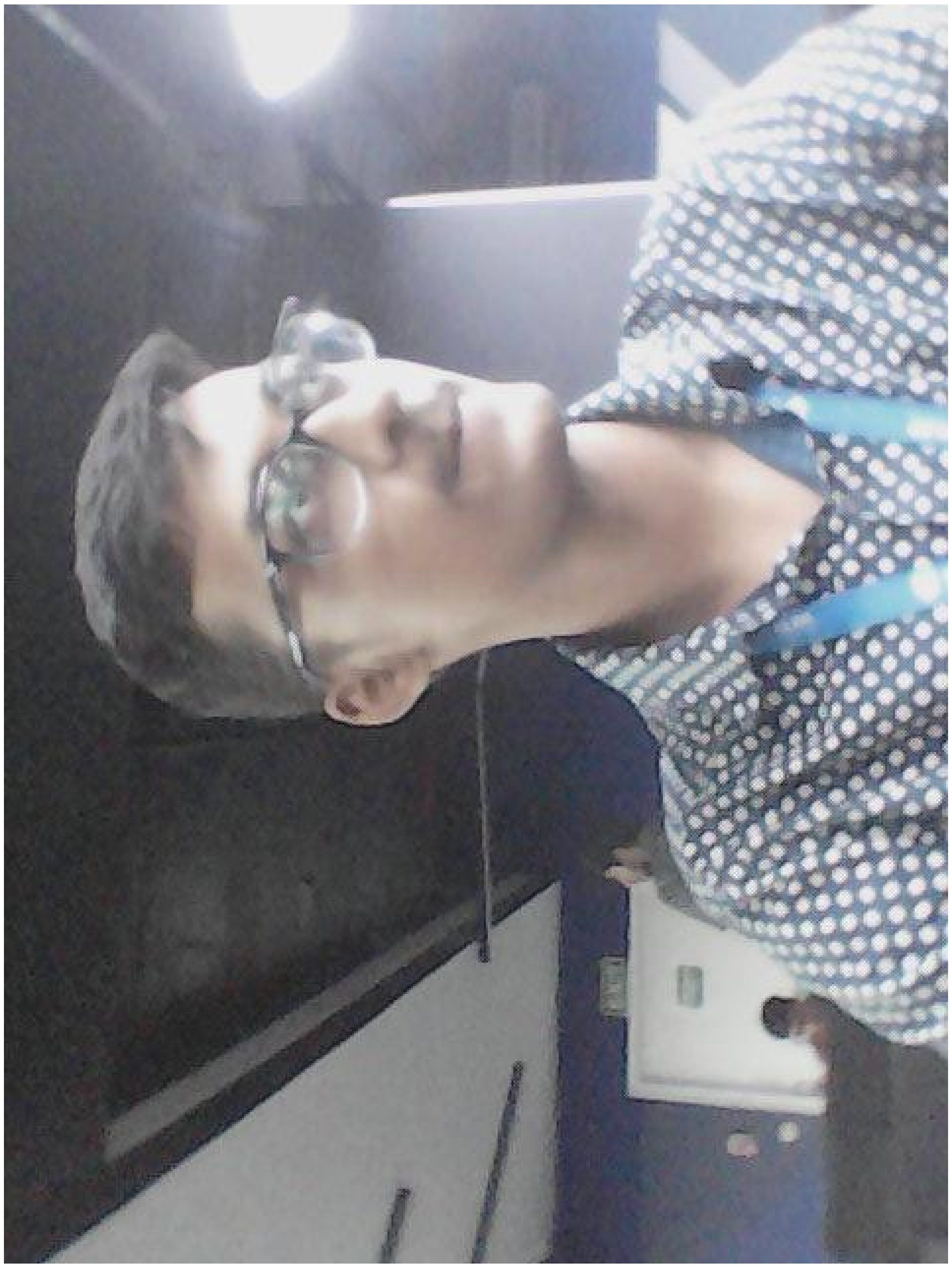


5)ROTATED ANTI-CLOCKWISE

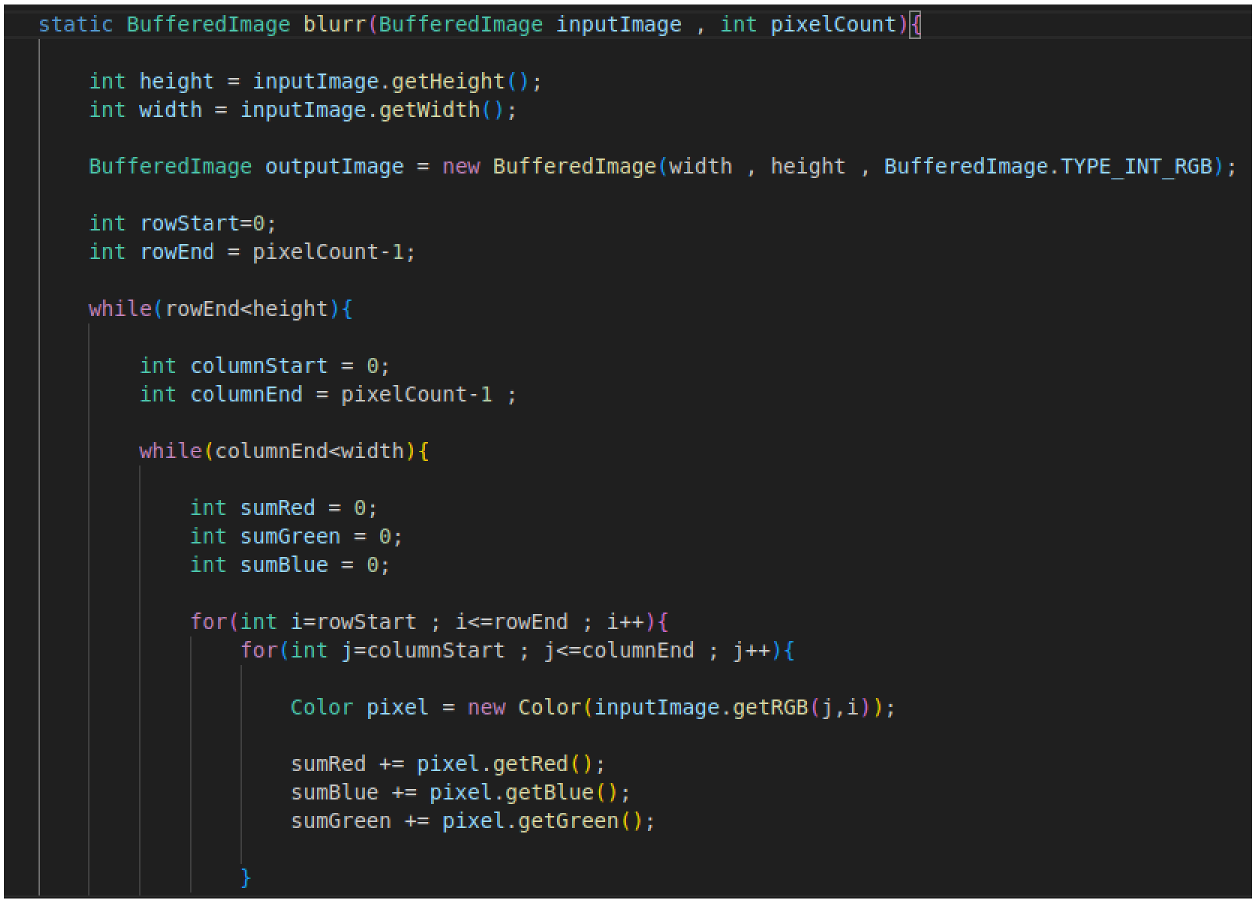


```
static BufferedImage rotateAntiClockwise(BufferedImage inputImage){
    BufferedImage outputImage = rotateClockwise(inputImage);
    outputImage = rotateClockwise(outputImage);
    outputImage = rotateClockwise(outputImage);
    return outputImage;
}
```

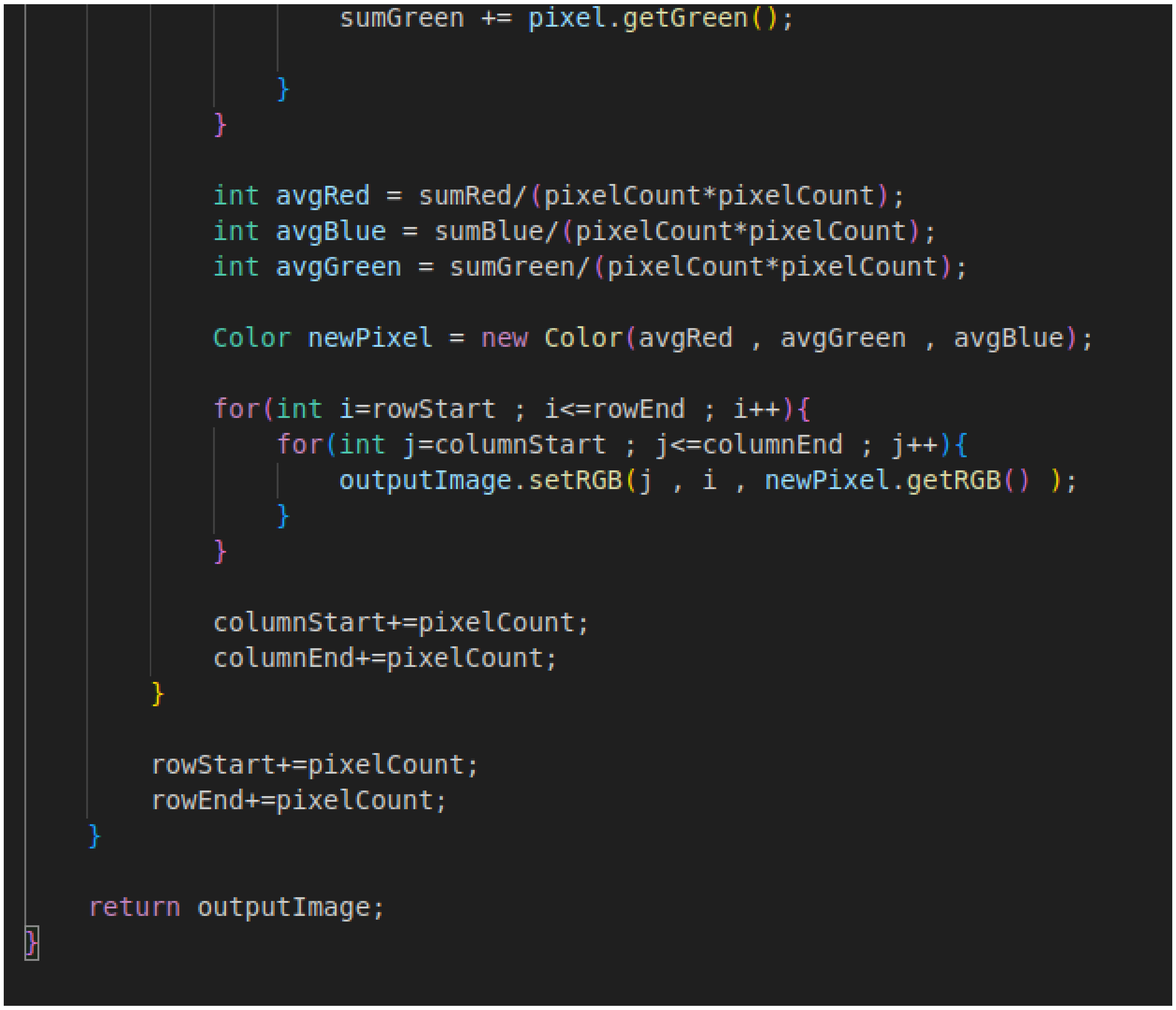
Call the function to rotate clockwise 3 times.



6)BLUR

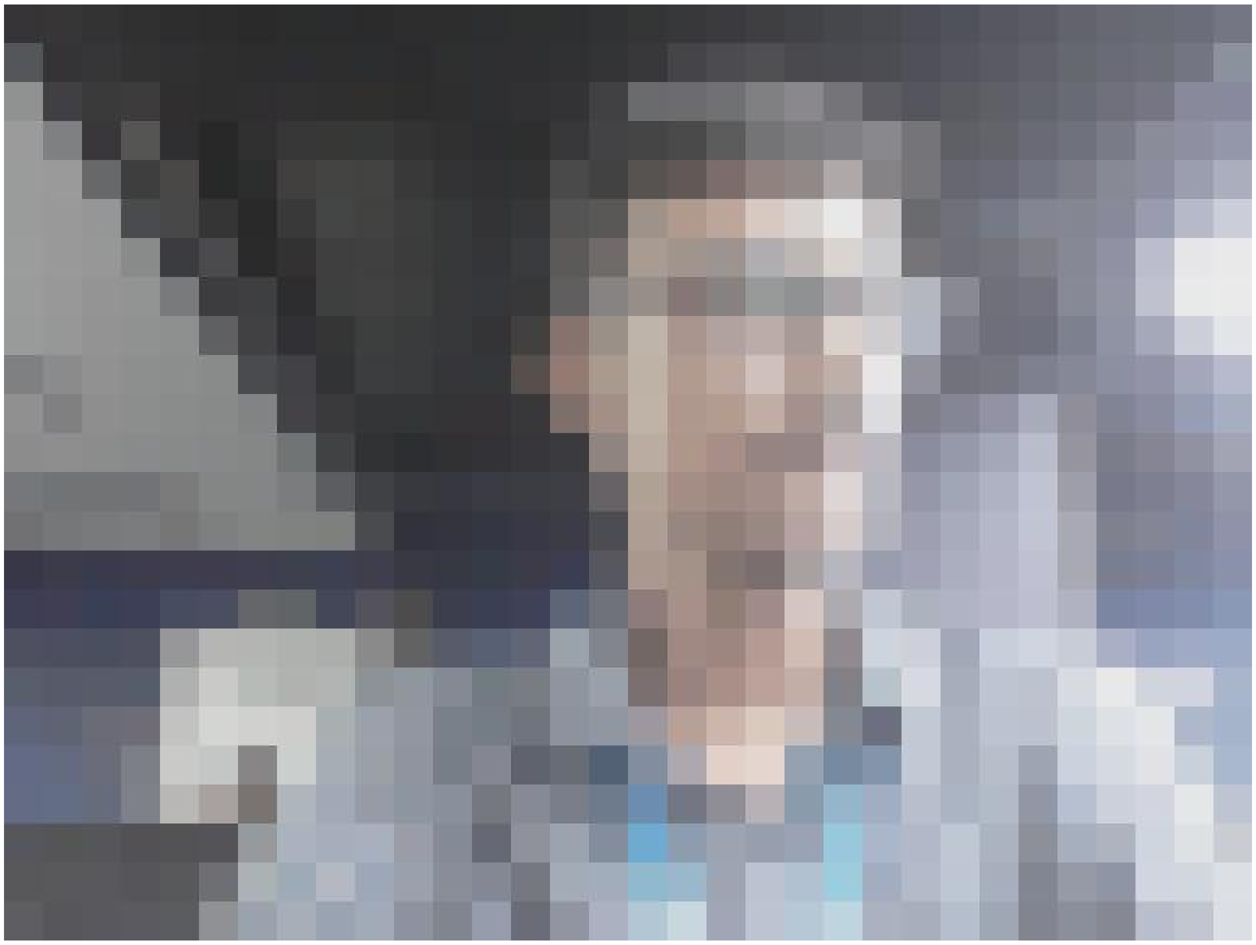


```java
static BufferedImage blurr(BufferedImage inputImage , int pixelCount){

    int height = inputImage.getHeight();
    int width = inputImage.getWidth();

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

    int rowStart=0;
    int rowEnd = pixelCount-1;

    while(rowEnd<height){

        int columnStart = 0;
        int columnEnd = pixelCount-1 ;

        while(columnEnd<width){

            int sumRed = 0;
            int sumGreen = 0;
            int sumBlue = 0;

            for(int i=rowStart ; i<=rowEnd ; i++){
                for(int j=columnStart ; j<=columnEnd ; j++){

                    Color pixel = new Color(inputImage.getRGB(j,i));

                    sumRed += pixel.getRed();
                    sumBlue += pixel.getBlue();
                    sumGreen += pixel.getGreen();

                }
```

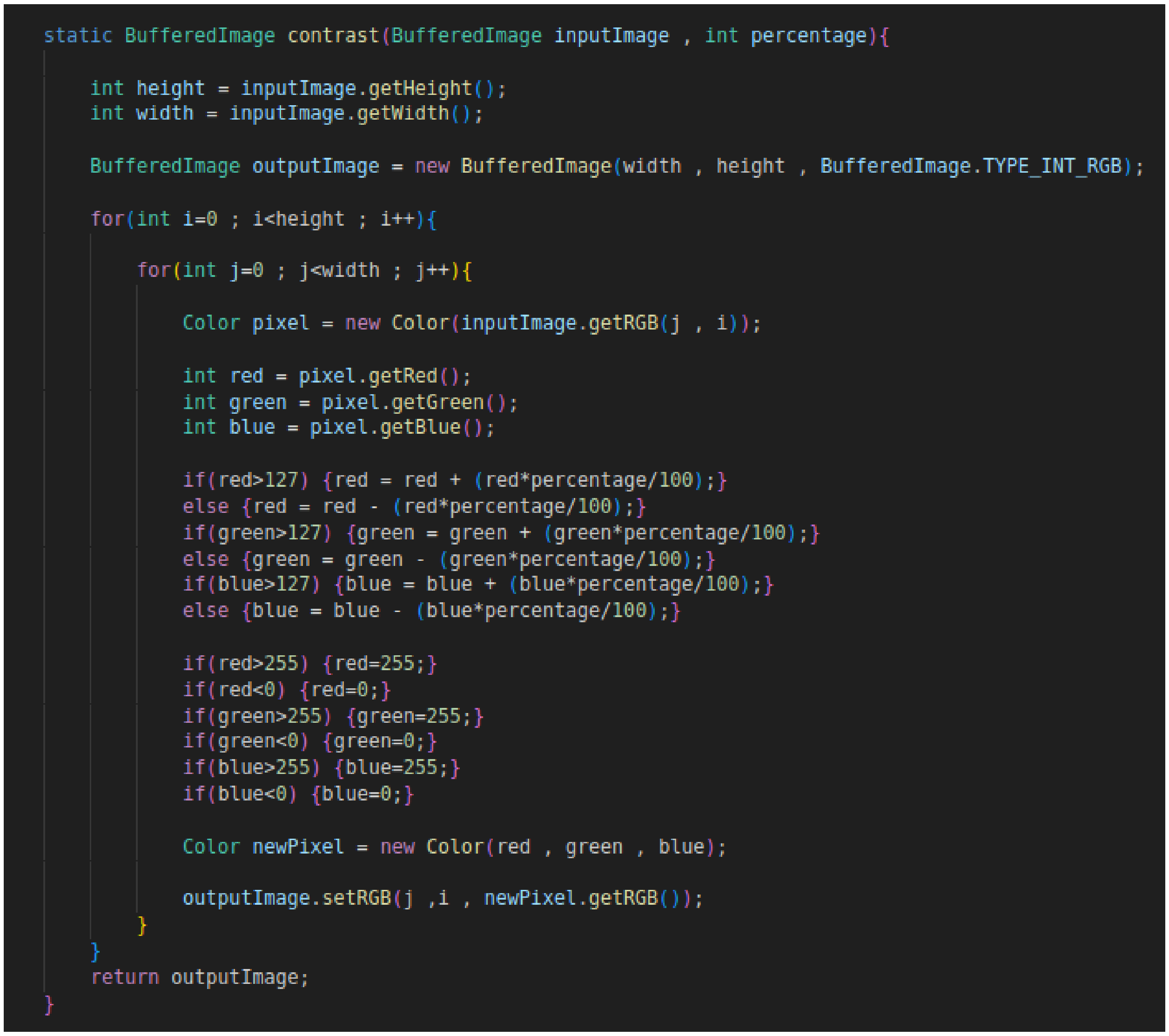


```
                    sumGreen += pixel.getGreen();

                }
            }

            int avgRed = sumRed/(pixelCount*pixelCount);
            int avgBlue = sumBlue/(pixelCount*pixelCount);
            int avgGreen = sumGreen/(pixelCount*pixelCount);

            Color newPixel = new Color(avgRed , avgGreen , avgBlue);

            for(int i=rowStart ; i<=rowEnd ; i++){
                for(int j=columnStart ; j<=columnEnd ; j++){
                    outputImage.setRGB(j , i , newPixel.getRGB() );
                }
            }

            columnStart+=pixelCount;
            columnEnd+=pixelCount;
        }

        rowStart+=pixelCount;
        rowEnd+=pixelCount;
    }

    return outputImage;
}
```

Takes 2 inputs , inputImage and pixelCount . pixelCount is the length of the square used for blurring.
Average of all pixels inside the square is taken and is then filled inside the complete pixelCount*pixelCount square.



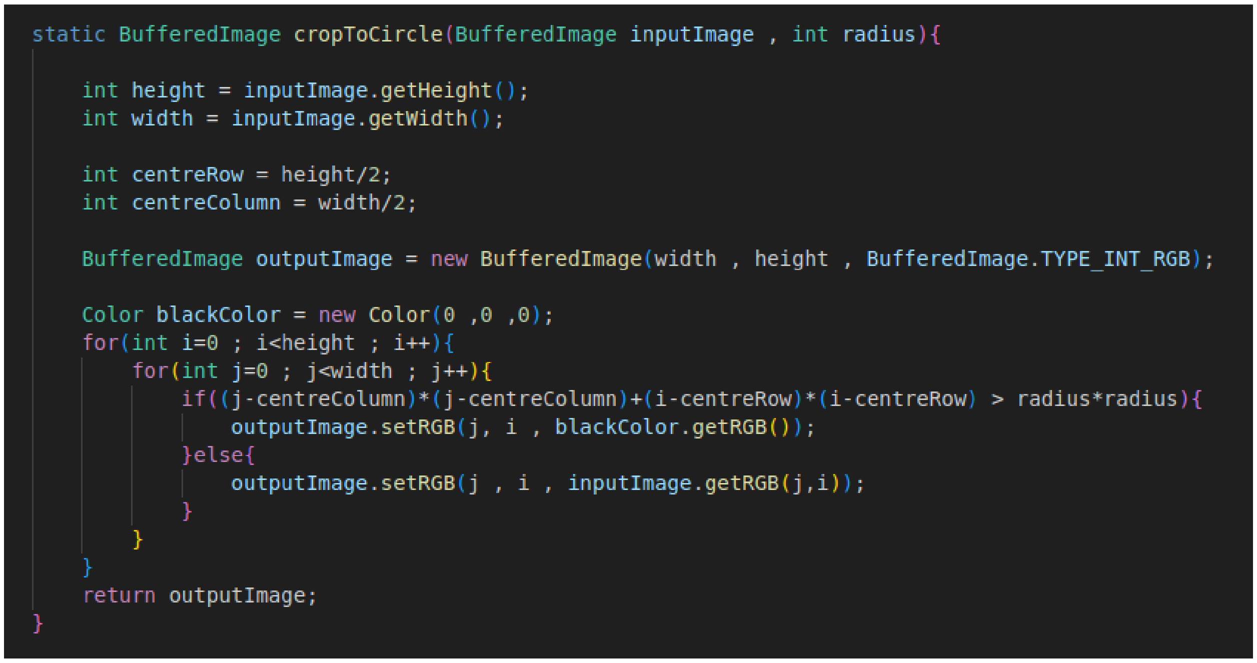
7)CONTRAST



```java
static BufferedImage contrast(BufferedImage inputImage , int percentage){

    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 ; j++){

            Color pixel = new Color(inputImage.getRGB(j , i));

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

            if(red>127) {red = red + (red*percentage/100);}
            else {red = red - (red*percentage/100);}
            if(green>127) {green = green + (green*percentage/100);}
            else {green = green - (green*percentage/100);}
            if(blue>127) {blue = blue + (blue*percentage/100);}
            else {blue = blue - (blue*percentage/100);}

            if(red>255) {red=255;}
            if(red<0) {red=0;}
            if(green>255) {green=255;}
            if(green<0) {green=0;}
            if(blue>255) {blue=255;}
            if(blue<0) {blue=0;}

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

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

Ask user for percentage. If R,B,G value is less than or equal to 127(255/2) , it decreases the values further by given percentage , else increases them.
This makes the dark darker and bright brighter.

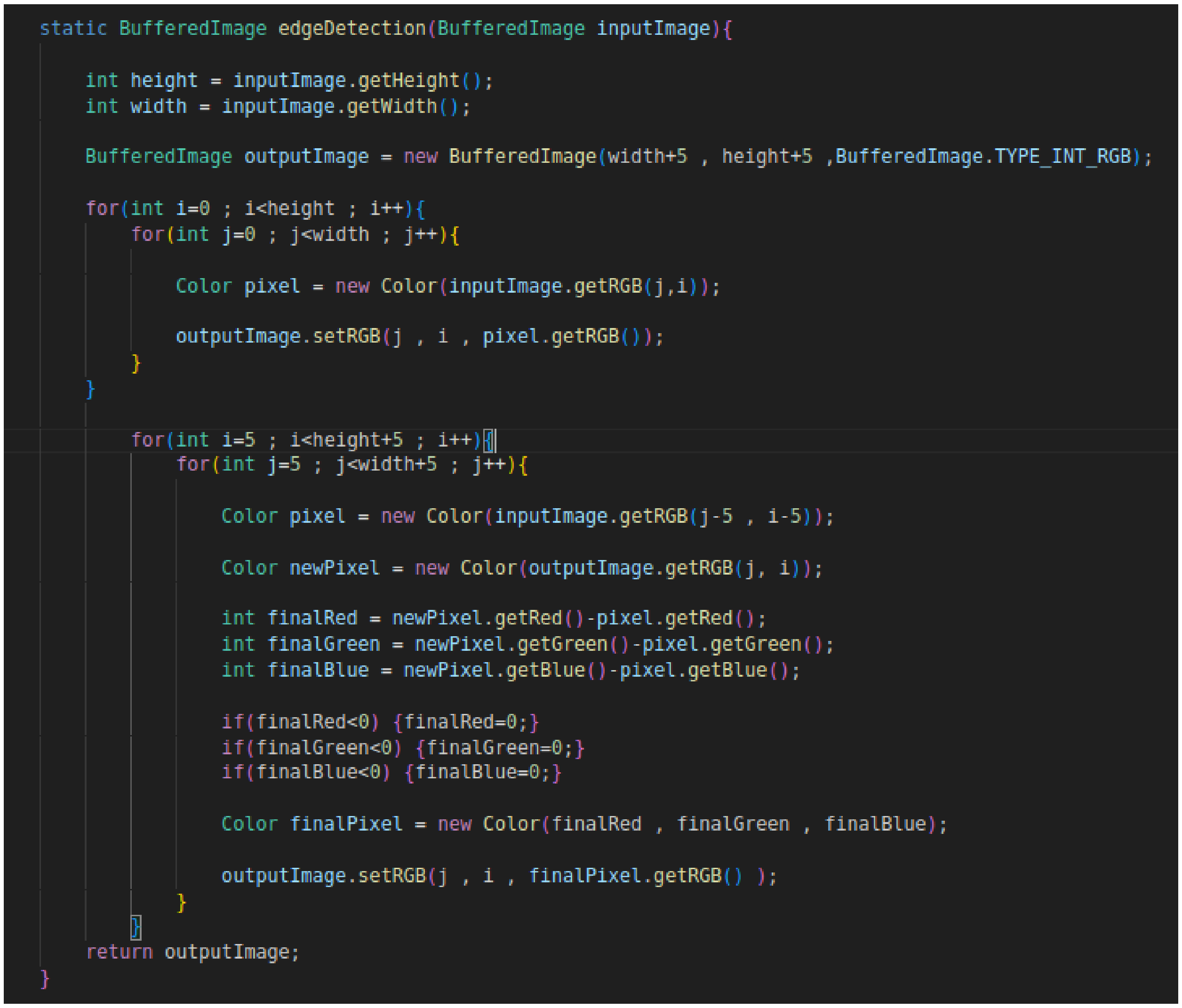
8)CROP TO CIRCLE



```java
static BufferedImage cropToCircle(BufferedImage inputImage , int radius){

    int height = inputImage.getHeight();
    int width = inputImage.getWidth();

    int centreRow = height/2;
    int centreColumn = width/2;

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

    Color blackColor = new Color(0 ,0 ,0);
    for(int i=0 ; i<height ; i++){
        for(int j=0 ; j<width ; j++){
            if((j-centreColumn)*(j-centreColumn)+(i-centreRow)*(i-centreRow) > radius*radius){
                outputImage.setRGB(j, i , blackColor.getRGB());
            }else{
                outputImage.setRGB(j , i , inputImage.getRGB(j,i));
            }
        }
    }
    return outputImage;
}
```

Ask user for radius .
Center pixel has co-ordinates (width/2 , height/2).
Put blak color inside the pixels which are more than radius pixels away from centre.
Keep the color in pixels inside the radius unchanged.

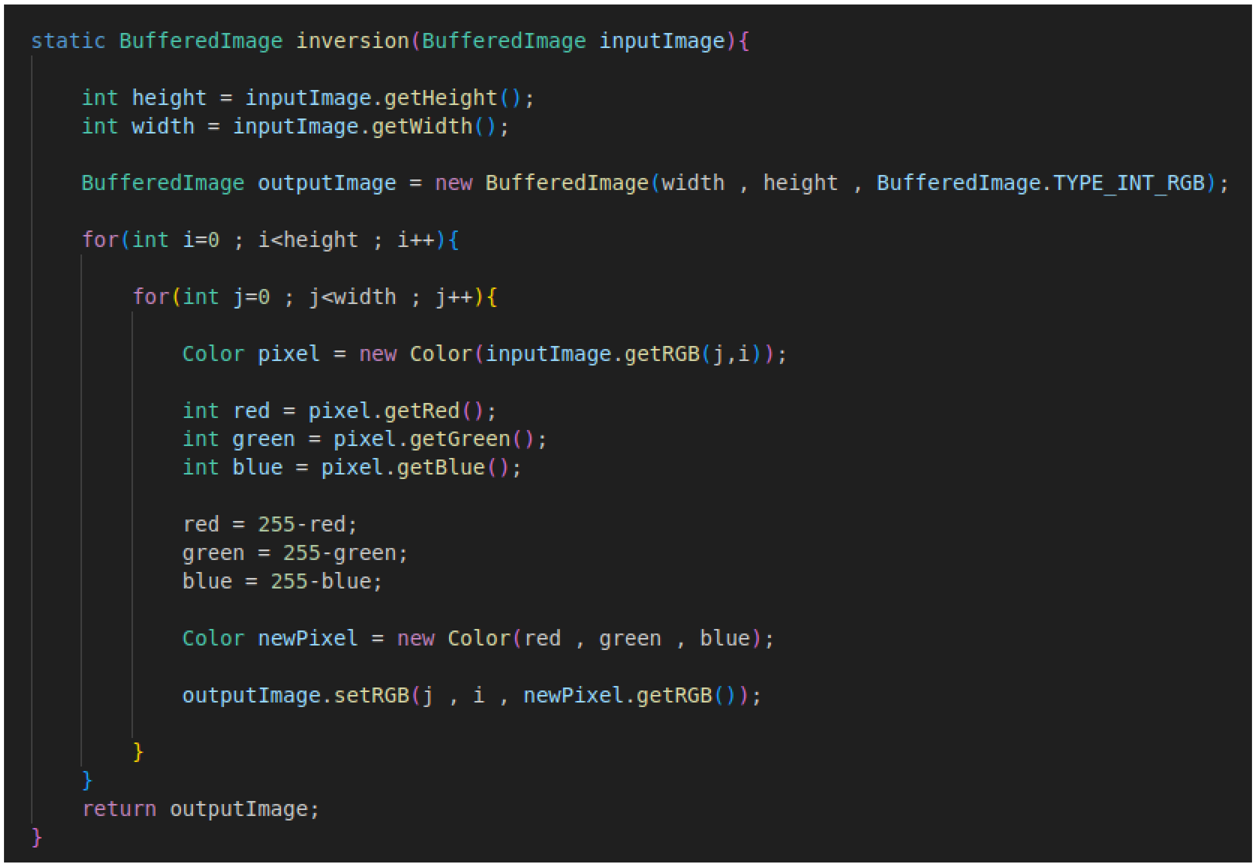
9)EDGE-DETECTION



```java
static BufferedImage edgeDetection(BufferedImage inputImage){

    int height = inputImage.getHeight();
    int width = inputImage.getWidth();

    BufferedImage outputImage = new BufferedImage(width+5 , height+5 ,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));

            outputImage.setRGB(j , i , pixel.getRGB());
        }
    }

        for(int i=5 ; i<height+5 ; i++){
            for(int j=5 ; j<width+5 ; j++){

                Color pixel = new Color(inputImage.getRGB(j-5 , i-5));

                Color newPixel = new Color(outputImage.getRGB(j, i));

                int finalRed = newPixel.getRed()-pixel.getRed();
                int finalGreen = newPixel.getGreen()-pixel.getGreen();
                int finalBlue = newPixel.getBlue()-pixel.getBlue();

                if(finalRed<0) {finalRed=0;}
                if(finalGreen<0) {finalGreen=0;}
                if(finalBlue<0) {finalBlue=0;}

                Color finalPixel = new Color(finalRed , finalGreen , finalBlue);

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

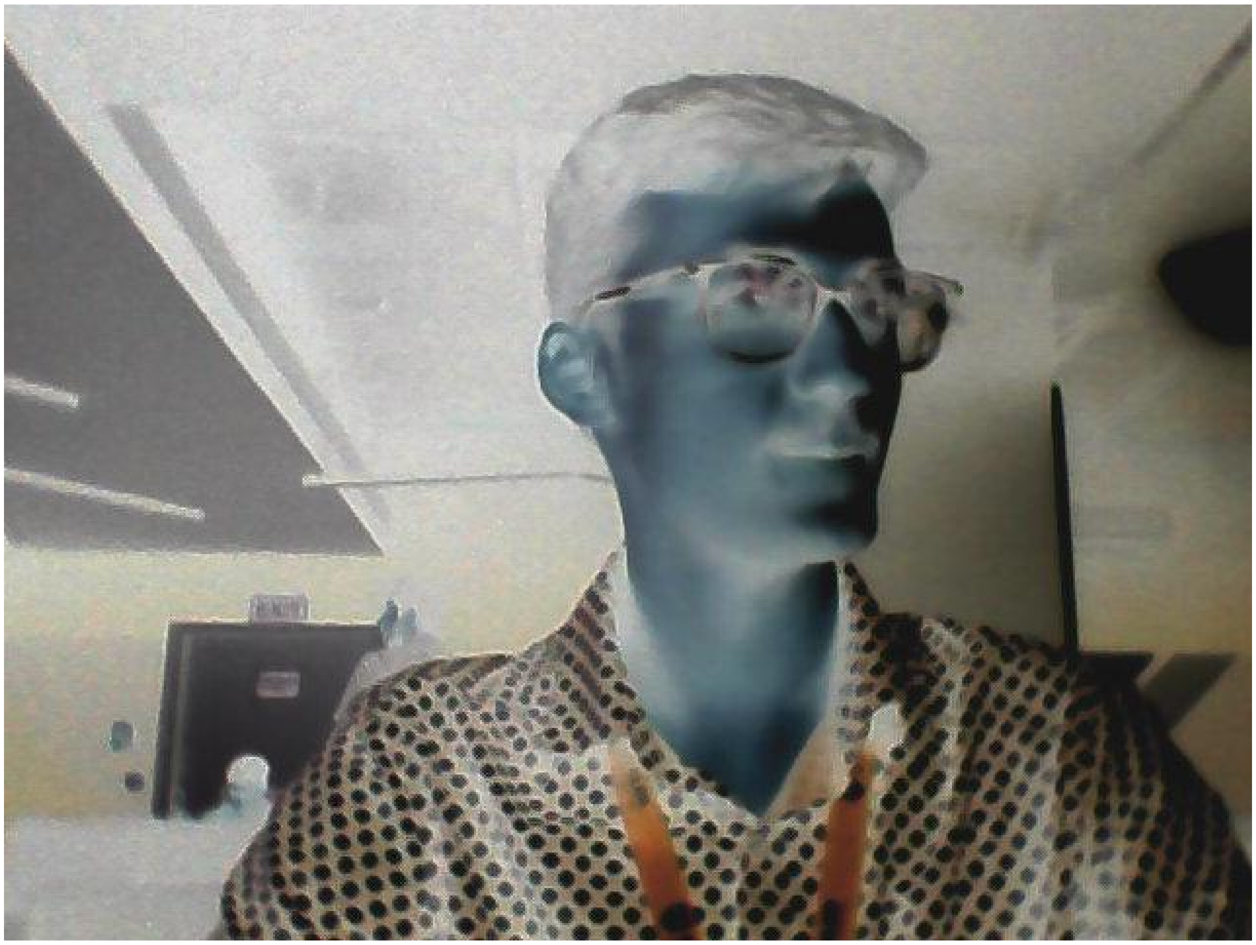
Keep the dimensions of output image (width+5 , height+5). Put the colors of original image inside output image. This leaves a margin of 5 pixels at right and at bottom.
Shift the input image by 5pixels from top and left and subtract its R,G,B values from the output image.

10)INVERSION

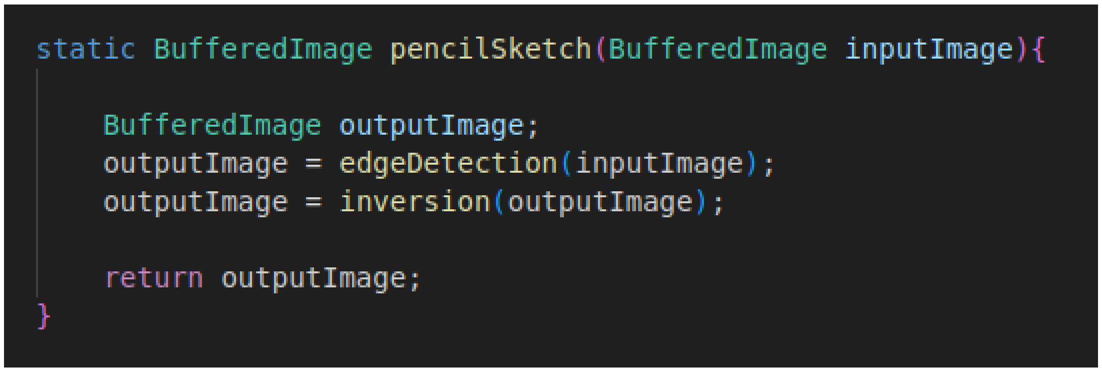


```java
static BufferedImage inversion(BufferedImage inputImage){

    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 ; j++){

            Color pixel = new Color(inputImage.getRGB(j,i));

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

            red = 255-red;
            green = 255-green;
            blue = 255-blue;

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

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

        }
    }
    return outputImage;
}
```

Replace all R,G,B values in all pixels by 255-(R,G,B).



11)PENCIL SKETCH



```
static BufferedImage pencilSketch(BufferedImage inputImage){

    BufferedImage outputImage;
    outputImage = edgeDetection(inputImage);
    outputImage = inversion(outputImage);

    return outputImage;
}
```

First call edge detection , then the inversion function.

THANK YOU !!!