ImageProcessor.java

```java
import java.io.*;

public class ImageProcessor {

    public static BufferedImage oImage;//Original image
    public static BufferedImage pImage;//Processed image
    public static final String FILE_NAME = "image.png";//Image file name assumed to be fixed
    public static final int IMG_DIM = 512; //Image dimension assumed to be fixed
    public static final int BTN_H = 100;//Button Height

    public static double currProgress = 0;//Current progress of the image process

    public static void main(String[] args) {
        //Frame for the program
        JFrame frame = new JFrame("Image Processor");
        frame.setLayout(null);
        frame.setBounds(0,0,IMG_DIM*2,IMG_DIM+BTN_H*2);

        loadImage(FILE_NAME);

        //Original Image
        JLabel orgImg = new JLabel(new ImageIcon(oImage));
        orgImg.setBounds(0,0,IMG_DIM,IMG_DIM);
        frame.add(orgImg);//Add image to the frame

        //Progress Bar
        JLabel progressBar = new JLabel();
        progressBar.setOpaque(true);
        progressBar.setBackground(Color.green);
        progressBar.setBounds(0,IMG_DIM+BTN_H,0,BTN_H);
        //Creates a separate thread that runs in the background; updating the progress bar based on the completed process
        Thread progressBarThread = new Thread()
        {
            public void run()
            {
                while(true)
                {
                    progressBar.setSize((int)(IMG_DIM*2*currProgress),BTN_H);
                    frame.repaint();
                }
            }
        };
        progressBarThread.start();
        frame.add(progressBar);//Add progress bar

        //Button to Process
        JButton btn = new JButton("Process Image!");
        btn.setBounds(0,IMG_DIM,frame.getWidth(),BTN_H);
        //On a button click the following happens
        btn.addActionListener(new ActionListener()
                {
                    //Not only creates an action listener but another thread to process the image
                    public void actionPerformed(ActionEvent e)
                    {
                        //Image Processing Thread
                        Thread imgThread = new Thread()
                                {
                                    public void run()
                                    {
                                        //The file processing begins
                                        processImage(FILE_NAME);
                                        //Once it has competed the new image is displayed
                                        JLabel newImg = new JLabel(new ImageIcon(pImage));
                                        newImg.setBounds(IMG_DIM,0,IMG_DIM,IMG_DIM);
                                        frame.add(newImg);//Adds the new image to the frame
                                        frame.repaint();//Redraws the newly added image
                                    }
                                };
                        imgThread.start();//Start processing!
                    }
                });
        frame.add(btn);//Add button to the frame

        //Makes sure the entire app is closed completely
        frame.addWindowListener(
                    new WindowAdapter()
                    {
                        public void windowClosing(WindowEvent we)
                        {
                            System.exit(0);
                        }
                    }
                );

        frame.setVisible(true);//Displays the frame
    }
    public static void loadImage(String name)
    {
        try
        {
            oImage = ImageIO.read(new File(name));
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
    public static void processImage(String name)
    {
        try
        {
            pImage = ImageIO.read(new File(name));

            int count = 0;
            for(int i=0;i<pImage.getHeight();i++)
            {
                for(int j=0;j<pImage.getWidth();j++)
                {
                    count++;
                    currProgress = (double)(count)/(double)(IMG_DIM*IMG_DIM);
                    System.out.println(currProgress);//Leave this in the slow down the processing
                    int rgbInt = pImage.getRGB(j, i);

                    //Invert colors
                    //int newColor = invertColor(rgbInt);

                    //Grey scale
                    //int newColor = greyScaleColor(rgbInt);

                    //Blur
                    int newColor = blurredColor(oImage,i,j,5);
                    pImage.setRGB(j, i, newColor);

                }
            }

            ImageIO.write(pImage, "png", new File("./output.png"));

        }
        catch(Exception e)
        {
            System.out.println(e);
            e.printStackTrace();
        }
    }
    public static int invertColor(int c)
    {
        int a = (c>>24)&0XFF; //Shift left 24 bits, then AND with 0xFF or 11111111 to get 
alpha
        int r = (c>>16)&0xFF; //Shift left 16 bits, then AND with 0xFF or 11111111 to get red
        int g = (c>>8)&0xFF;
        int b = (c)&0xFF;

        //Subtracting the max value of r,g,b (IE 255) flips the colors
        return (a << 24) |
                ((255-r) << 16) |
                ((255-g) << 8) |
                ((255-b));
    }
    public static int greyScaleColor(int c)
    {
        int a = (c>>24)&0XFF; //Shift left 24 bits, then AND with 0xFF or 11111111 to get 
alpha
        int r = (c>>16)&0xFF; //Shift left 16 bits, then AND with 0xFF or 11111111 to get red
        int g = (c>>8)&0xFF;
        int b = (c)&0xFF;

        int avg = (r+g+b)/3; //Take average of all 3 colors to make it greyscale

        return (a << 24) |
                ((avg) << 16) |
                ((avg) << 8) |
                ((avg));
    }
    /**
     *
     * @param img - image to be blurred
     * @param i - start pixel vert
     * @param j - start pixel hori
     * @param pRange - How far extended from i,j will it be blurred. Higher = blurrier
     * @return Average of color data thus a blurred pixel
     */
    public static int blurredColor(BufferedImage img, int i, int j, int pRange)
    {
        int rs = 0;
        int gs = 0;
        int bs = 0;
        int a = 0xFF;//Constant alpha
        int count = 0;
        for(int k = i-pRange;k<i+pRange;k++)
        {
            for(int l = j-pRange;l<j+pRange;l++)
            {
                if(!isValid(k) || !isValid(l))
                    continue;
                int c = img.getRGB(l, k);
                rs += getRed(c);
                gs += getGreen(c);
                bs += getBlue(c);
                count++;
            }
        }
        rs /= count;
        gs /= count;
        bs /= count;

        return (a << 24) |
                ((rs) << 16) |
                ((gs) << 8) |
                ((bs));
    }
    public static int getAlpha(int c)
    {
        return (c >> 24) & 0xFF;
    }
    public static int getRed(int c)
    {
        return (c >> 16) & 0xFF;
    }
    public static int getGreen(int c)
    {
        return (c >> 8) & 0xFF;
    }
    public static int getBlue(int c)
    {
        return c & 0xFF;
    }
    public static boolean isValid(int i)
    {
        return i >= 0 && i < IMG_DIM;
    }
}
```