

```

//This program takes an image and inverts its color
import java.awt.Color;
import java.io.File;
import java.net.URL;
import java.awt.image.BufferedImage;
import java.awt.image.ColorModel;
import java.awt.image.Raster;
import java.awt.image.WritableRaster;
import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;

/**
 * A picture whose pixels can be read and written.
 */
public class Picture
{
    private String source;
    private JFrame frame;
    private JLabel label;
    private BufferedImage image;

    /**
     * Constructs a picture with no image.
     */
    public Picture()
    {
        frame = new JFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        label = new JLabel("(No image)");
        frame.add(label);
        frame.pack();
        frame.setVisible(true);
    }

    /**
     * Gets the width of this picture.
     * @return the width
     */
    public int getWidth() { return image.getWidth(); }

    /**
     * Gets the height of this picture.
     * @return the height
     */
    public int getHeight() { return image.getHeight(); }

    /**
     * Loads a picture from a given source.
     * @param source the image source. If the source starts
     * with http://, it is a URL, otherwise, a filename.
     */
    public void load(String source)
    {
        try

```

```

    {
        this.source = source;
        BufferedImage img;
        if (source.startsWith("http://"))
            img = ImageIO.read(new URL(source).openStream());
        else
            img = ImageIO.read(new File(source));

        setImage(img);
    }
    catch (Exception ex)
    {
        this.source = null;
        ex.printStackTrace();
    }
}

/**
 * Reloads this picture, undoing any manipulations.
 */
public void reload()
{
    load(source);
}

/**
 * Displays a file chooser for picking a picture.
 */
public void pick()
{
    JFileChooser chooser = new JFileChooser(".");
    if (chooser.showOpenDialog(null) == JFileChooser.APPROVE_OPTION)
    {
        load(chooser.getSelectedFile().getAbsolutePath());
    }
}

private void setImage(BufferedImage image)
{
    this.image = image;
    label.setIcon(new ImageIcon(image));
    label.setText(" ");
    frame.pack();
}

/**
 * Gets the color of a pixel.
 * @param x the column index (between 0 and getWidth() - 1)
 * @param y the row index (between 0 and getHeight() - 1)
 * @return the color of the pixel at position (x, y)
 */
public Color getColorAt(int x, int y)
{
    Raster raster = image.getRaster();
    ColorModel model = image.getColorModel();
    int argb = model.getRGB(raster.getDataElements(x, y, null));
    return new Color(argb, true);
}

```

```

    }

    /**
     Sets the color of a pixel.
     @param x the column index (between 0 and getWidth() - 1)
     @param y the row index (between 0 and getHeight() - 1)
     @param c the color for the pixel at position (x, y)
    */
    public void setColorAt(int x, int y, Color c)
    {
        WritableRaster raster = image.getRaster();
        ColorModel model = image.getColorModel();
        Object colorData = model.getDataElements(c.getRGB(), null);
        raster.setDataElements(x, y, colorData);
        label.repaint();
    }
    public static void main(String[] args)
    {
        Picture pic = new Picture();
        pic.pick();

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

        for(int j = 0; j <= height - 1; j++)
        {
            for(int i = 0; i <= width - 1; i++)
            {
                Color original = pic.getColorAt(i, j);
                int red = 255 - original.getRed();
                int green = 255 - original.getGreen();
                int blue = 255 - original.getBlue();
                Color negative = new Color(red, green, blue);
                pic.setColorAt(i, j, negative);
            }
        }
    }
}
//Made by Ivan K.

```