src\ImageEditorPanel.java

```java
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.io.File;
import javax.imageio.ImageIO;
import java.awt.*;
import javax.swing.*;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

public class ImageEditorPanel extends JPanel implements KeyListener{

    Color[][] pixels;

    public ImageEditorPanel() {
        BufferedImage imageIn = null;
        try {
            // the image should be in the main project folder, not in \src or \bin
            imageIn = ImageIO.read(new File("steve.jpg"));
        } catch (IOException e) {
            System.out.println(e);
            System.exit(1);
        }
        pixels = makeColorArray(imageIn);
        setPreferredSize(new Dimension(pixels[0].length, pixels.length));
        setBackground(Color.BLACK);
        addKeyListener(this);
    }

    public void paintComponent(Graphics g) {
        // paints the array pixels onto the screen
        for (int row = 0; row < pixels.length; row++) {
            for (int col = 0; col < pixels[0].length; col++) {
                g.setColor(pixels[row][col]);
                g.fillRect(col, row, 1, 1);
            }
        }
    }

    public Color[][] flipHoriz(Color[][] pixels){
        Color[][] copy = new Color[pixels.length][pixels[0].length];
        for (int r = 0; r < pixels.length; r++) {
            for (int c = 0; c < pixels[r].length; c++) {

                copy[r][(pixels[0].length - 1) - c] = pixels[r][c];
            }
        }
        return copy;
    }

    public Color[][] flipVert(Color[][] pixels){
        Color[][] copy = new Color[pixels.length][pixels[0].length];

        for (int r = 0; r < pixels.length; r++) {
```

```
54              for (int c = 0; c < pixels[r].length; c++) {
55
56                  copy[(pixels.length - 1) - r][c] = pixels[r][c];
57              }
58          }
59          return copy;
60      }
61
62      public Color[][] grey(Color[][] pixels){
63          Color[][] greyscale = new Color[pixels.length][pixels[0].length];
64
65          for (int r = 0; r < greyscale.length; r++) {
66              for (int c = 0; c < greyscale[r].length; c++) {
67                  Color clr = pixels[r][c];
68                  int red = clr.getRed();
69                  int blue = clr.getBlue();
70                  int green = clr.getGreen();
71                  int grey = (red + blue + green) / 3;
72                  greyscale[r][c] = new Color(grey, grey, grey);
73              }
74          }
75          return greyscale;
76      }
77
78      public Color[][] rotate(Color[][] pixels){
79          Color[][] copy = new Color[pixels[0].length][pixels.length];
80
81          for (int r = 0; r < pixels.length; r++) {
82              for (int c = 0; c < pixels[r].length; c++) {
83
84                  copy[(pixels[0].length - 1) - c][(pixels.length - 1) - r] = pixels[r][c];
85              }
86          }
87          return copy;
88
89      }
90
91      public Color[][] contrast(Color[][] pixels){
92          Color[][] copy = new Color[pixels.length][pixels[0].length];
93          final double SCALE = 2.1;
94          for (int r = 0; r < pixels.length; r++) {
95              for (int c = 0; c < pixels[r].length; c++) {
96                  Color clr = pixels[r][c];
97                  int blue = (int)(clr.getBlue() * SCALE);
98                  int green = (int)(clr.getGreen() * SCALE);
99                  int red = (int)(clr.getRed() * SCALE);
100
101                 if (blue > 255){
102                     blue = 255;
103                 }
104                 if (green > 255){
105                     green = 255;
106                 }
107                 if (red > 255){
108                     red = 255;
109                 }
```

```
110
111                 copy[r][c] = new Color(red, green, blue);
112             }
113         }
114         return copy;
115
116     }
117
118     public Color[][] dull(Color[][] pixels){
119         Color[][] copy = new Color[pixels.length][pixels[0].length];
120         final double SCALE = .64;
121         for (int r = 0; r < pixels.length; r++) {
122             for (int c = 0; c < pixels[r].length; c++) {
123                 Color clr = pixels[r][c];
124                 int blue = (int)(clr.getBlue() * SCALE);
125                 int green = (int)(clr.getGreen() * SCALE);
126                 int red = (int)(clr.getRed() * SCALE);
127
128                 copy[r][c] = new Color(red, green, blue);
129             }
130         }
131         return copy;
132     }
133
134     public Color[][] tan(Color[][] pixels){
135         Color[][] copy = new Color[pixels.length][pixels[0].length];
136         final double SCALEGREEN = 1.5;
137         final double SCALERED = 1.4;
138         for (int r = 0; r < pixels.length; r++) {
139             for (int c = 0; c < pixels[r].length; c++) {
140                 Color clr = pixels[r][c];
141                 int blue = clr.getBlue();
142                 int green = (int)(clr.getGreen() * SCALEGREEN);
143                 int red = (int)(clr.getRed() * SCALERED);
144
145                 if (blue > 255){
146                     blue = 255;
147                 }
148                 if (green > 255){
149                     green = 255;
150                 }
151                 if (red > 255){
152                     red = 255;
153                 }
154
155                 copy[r][c] = new Color(red, green, blue);
156             }
157         }
158         return copy;
159     }
160
161     public Color[][] bright(Color[][] pixels){
162         Color[][] copy = new Color[pixels.length][pixels[0].length];
163         for (int r = 0; r < pixels.length; r++) {
164             for (int c = 0; c < pixels[r].length; c++) {
165                 Color clr = pixels[r][c];
```

```
166                 copy[r][c] = clr.brighter();
167             }
168         }
169         return copy;
170     }
171 
172     public Color[][] dark(Color[][] pixels){
173         Color[][] copy = new Color[pixels.length][pixels[0].length];
174         for (int r = 0; r < pixels.length; r++) {
175             for (int c = 0; c < pixels[r].length; c++) {
176                 Color clr = pixels[r][c];
177                 copy[r][c] = clr.darker();
178             }
179         }
180         return copy;
181     }
182 
183     public Color[][] blur(Color[][] pixels){
184     Color[][] copy = new Color[pixels.length][pixels[0].length];
185 
186     for (int r = 0; r < pixels.length; r++) {
187         for (int c = 0; c < pixels[0].length; c++) {
188             int red = 0;
189             int green = 0;
190             int blue = 0;
191             int pixelCt = 0;
192             for (int i = -7; i < 7; i++) {
193                 for (int j = -7; j < 7; j++) {
194 
195                     if (i + r >= 0 && j + c >= 0 && i + r < pixels.length && j + c <
    pixels[0].length){
196                         pixelCt++;
197                     Color extract = pixels[i + r][j + c];
198                     green += extract.getGreen();
199                     blue += extract.getBlue();
200                     red += extract.getRed();
201                     System.out.println(red);
202                     }
203                     }
204                 }
205                 System.out.println(pixelCt);
206                 copy[r][c] = new Color(red / pixelCt, green / pixelCt, blue / pixelCt);
207             }
208         }
209         return copy;
210     }
211 
212 
213 
214     public Color[][] poster(Color[][] pixels){
215     Color[][] copy = new Color[pixels.length][pixels[0].length];
216         return copy;
217     }
218 
219     public Color[][] makeColorArray(BufferedImage image) {
220         int width = image.getWidth();
```

```
221         int height = image.getHeight();
222         Color[][] result = new Color[height][width];
223 
224         for (int row = 0; row < height; row++) {
225             for (int col = 0; col < width; col++) {
226                 Color c = new Color(image.getRGB(col, row), true);
227                 result[row][col] = c;
228             }
229         }
230         // System.out.println("Loaded image: width: " +width + " height: " + height);
231         return result;
232     }
233 
234     @Override
235     public void keyTyped(KeyEvent e) {
236         if(e.getKeyChar() == 'f'){
237             pixels = flipHoriz(pixels);
238         }
239         if(e.getKeyChar() == 'b'){
240             pixels = bright(pixels);
241         }
242         if(e.getKeyChar() == 'd'){
243             pixels = dark(pixels);
244         }
245         if(e.getKeyChar() == 'g'){
246             pixels = grey(pixels);
247         }
248         if(e.getKeyChar() == 'h'){
249             pixels = flipHoriz(pixels);
250         }
251         if(e.getKeyChar() == 'r'){
252             pixels = rotate(pixels);
253         }
254         if(e.getKeyChar() == 'c'){
255             pixels = contrast(pixels);
256         }
257         if(e.getKeyChar() == 'v'){
258             pixels = flipVert(pixels);
259         }
260         if(e.getKeyChar() == 'd'){
261             pixels = dull(pixels);
262         }
263         if(e.getKeyChar() == 't'){
264             pixels = tan(pixels);
265         }
266         if(e.getKeyChar() == 'l'){
267             pixels = blur(pixels);
268         }
269         repaint();
270     }
271 
272     @Override
273     public void keyPressed(KeyEvent e) {
274         // TODO Auto-generated method stub
275     }
276 
```

```
277     @Override
278     public void keyReleased(KeyEvent e) {
279         // TODO Auto-generated method stub
280     }
281 }
282 
```