

```

1 import java.util.*;
2 import java.awt.image.BufferedImage;
3 import java.io.*;
4 import java.awt.Color;
5
6 /*****
7 Effects contains methods for editing pictures in the
8 steganography GUI.
9
10 @author Pranav Ramanan
11 @version 06/05/2014
12 *****/
13 public class Effects
14 {
15     /*****
16     Creates new image that is opposite color of old image
17     *****/
18     public static BufferedImage Inverse(BufferedImage img, int h, int w)
19     {
20
21         BufferedImage newImage = new BufferedImage(w, h, BufferedImage.TYPE_INT_RGB);
22
23         for(int y=0; y<h; y++)
24         {
25             for(int x=0; x<w; x++)
26             {
27                 int rgb=img.getRGB(x, y);
28                 int r = (rgb)&0xFF;
29                 int g = (rgb>>8)&0xFF;
30                 int b = (rgb>>16)&0xFF;
31
32                 int invr= 255-r;
33                 int invg= 255-g;
34                 int invb= 255-b;
35
36                 int invColor = invr;
37                 invColor = (invColor << 8) + invg;
38                 invColor = (invColor << 8) + invb;
39                 newImage.setRGB(x, y, invColor);
40             }
41         }
42         return newImage;
43     }
44
45     /*****
46     Takes the image and slightly changes the color of the
47     pixels to give it an opaque look.
48     *****/
49     public static BufferedImage Fade(BufferedImage img, int h, int w,int fade,String RGB
50 )
51     {
52         BufferedImage newImage = new BufferedImage(w, h, BufferedImage.TYPE_INT_RGB);
53         for(int y=0; y<h; y++)
54         {
55             for(int x=0; x<w; x++)
56             {
57                 int rgb=img.getRGB(x, y);
58                 Color c = new Color(rgb);

```

```

59         Color tintColor = c;
60     if( RGB.toUpperCase().equals("R"))
61     {
62         int boostRed = c.getRed()-fade;
63         if(boostRed<0) boostRed=0;
64         if(boostRed>255) boostRed=225;
65         tintColor = new Color(boostRed,c.getGreen(),c.getBlue());
66     }
67     else if( RGB.toUpperCase().equals("G"))
68     {
69         int boostGreen = c.getGreen()-fade;
70         if(boostGreen<0) boostGreen=0;
71         if(boostGreen>225) boostGreen=225;
72         tintColor = new Color(c.getRed(),boostGreen,c.getBlue());
73     }
74     else if( RGB.toUpperCase().equals("B"))
75     {
76         int boostBlue = c.getBlue()-fade;
77         if(boostBlue<0) boostBlue=0;
78         if(boostBlue>225) boostBlue=225;
79         tintColor = new Color(c.getRed(),c.getGreen(),boostBlue);
80     }
81     else if( RGB.toUpperCase().equals("ALL"))
82     {
83         int boostRed = c.getRed()-fade;
84         if(boostRed<0) boostRed=0;
85         int boostGreen = c.getGreen()-fade;
86         if(boostGreen<0) boostGreen=0;
87         int boostBlue = c.getBlue()-fade;
88         if(boostBlue<0) boostBlue=0;
89         tintColor = new Color(boostRed,boostGreen,boostBlue);
90     }
91     int newColor = tintColor.getRGB();
92     newImage.setRGB(x, y, newColor);
93 }
94 }
95 return newImage;
96 }
97
98 /*****
99 Takes the image and adds a color to each of the pixels
100 to tint the image.
101 *****/
102 public static BufferedImage Tint(BufferedImage img, int h, int w, int tint, String RGB)
103 {
104
105     BufferedImage newImage = new BufferedImage(w, h, BufferedImage.TYPE_INT_RGB);
106
107     for(int y=0; y<h; y++)
108     {
109         for(int x=0; x<w; x++)
110         {
111             int rgb=img.getRGB(x, y);
112             Color c = new Color(rgb);
113             Color tintColor = c;
114             if( RGB.toUpperCase().equals("R"))
115             {
116                 int boostRed = c.getRed()+tint;

```

```

117         if(boostRed>255) boostRed=225;
118         if(boostRed<0) boostRed=0;
119         tintColor = new Color(boostRed,c.getGreen(),c.getBlue());
120     }
121     else if(RGB.toUpperCase().equals("G"))
122     {
123         int boostGreen = c.getGreen()+tint;
124         if(boostGreen>225) boostGreen=225;
125         if(boostGreen<0) boostGreen=0;
126         tintColor = new Color(c.getRed(),boostGreen,c.getBlue());
127     }
128     else if(RGB.toUpperCase().equals("B"))
129     {
130         int boostBlue = c.getBlue()+tint;
131         if(boostBlue>225) boostBlue=225;
132         if(boostBlue<0) boostBlue=0;
133         tintColor = new Color(c.getRed(),c.getGreen(),boostBlue);
134     }
135     else if(RGB.toUpperCase().equals("ALL"))
136     {
137         int boostRed = c.getRed()+tint;
138         if(boostRed>255) boostRed=225;
139         int boostGreen = c.getGreen()+tint;
140         if(boostGreen>225) boostGreen=225;
141         int boostBlue = c.getBlue()+tint;
142         if(boostBlue>225) boostBlue=225;
143         tintColor = new Color(boostRed,boostGreen,boostBlue);
144     }
145     int newColor = tintColor.getRGB();
146     newImage.setRGB(x, y, newColor);
147 }
148 }
149 return newImage;
150 }
151
152 /*****
153 Creates image that is only black and white.
154 *****/
155 public static BufferedImage BlackWhite(BufferedImage img, int h, int w)
156 {
157     BufferedImage newImage = new BufferedImage(w, h, BufferedImage.TYPE_INT_RGB);
158     int avg=0;
159     for(int y=0; y<h; y++)
160     {
161         for(int x=0; x<w; x++)
162         {
163             int rgb=img.getRGB(x, y);
164             int r = (rgb)&0xFF;
165             int g = (rgb>>8)&0xFF;
166             int b = (rgb>>16)&0xFF;
167             avg+=(r+g+b)/3;
168         }
169     }
170     avg=avg/(h*w);
171     for(int y=0; y<h; y++)
172     {
173         for(int x=0; x<w; x++)
174         {
175

```

```

176         int rgb=img.getRGB(x, y);
177         int r = (rgb)&0xFF;
178         int g = (rgb>>8)&0xFF;
179         int b = (rgb>>16)&0xFF;
180
181         if (((r+g+b)/3)>avg)
182         {
183             r=225;
184             g=225;
185             b=225;
186         }
187         else
188         {
189             r=0;
190             g=0;
191             b=0;
192         }
193
194         int newColor = r;
195         newColor = (newColor << 8) + g;
196         newColor = (newColor << 8) + b;
197
198         newImage.setRGB(x, y, newColor);
199     }
200 }
201
202 return newImage;
203 }
204
205 /*****
206 Creates image that is only black and white.
207 *****/
208 public static BufferedImage Grayscale(BufferedImage img, int h, int w)
209 {
210     BufferedImage newImage = new BufferedImage(w, h, BufferedImage.TYPE_INT_RGB);
211
212     for(int y=0; y<h; y++)
213     {
214         for(int x=0; x<w; x++)
215         {
216
217             int rgb=img.getRGB(x, y);
218             int r = (rgb)&0xFF;
219             int g = (rgb>>8)&0xFF;
220             int b = (rgb>>16)&0xFF;
221
222             int combColor = (r+b+g)/3;
223
224             int newColor = combColor;
225             newColor = (newColor << 8) + combColor;
226             newColor = (newColor << 8) + combColor;
227
228             newImage.setRGB(x, y, newColor);
229         }
230     }
231
232     return newImage;
233 }
234 /*****

```

```

235 Takes the image removes all the red, green, or blue.
236 *****/
237 public static BufferedImage Remove(BufferedImage img, int h, int w, String RGB)
238 {
239
240     BufferedImage newImage = new BufferedImage(w, h, BufferedImage.TYPE_INT_RGB);
241
242     for(int y=0; y<h; y++)
243     {
244         for(int x=0; x<w; x++)
245         {
246
247             int rgb=img.getRGB(x, y);
248             Color c = new Color(rgb);
249             Color tintColor = c;
250             if(RGB.toUpperCase().equals("R"))
251             {
252                 tintColor = new Color(000,c.getGreen(),c.getBlue());
253             }
254             else if(RGB.toUpperCase().equals("G"))
255             {
256                 tintColor = new Color(c.getRed(),000,c.getBlue());
257             }
258             else if(RGB.toUpperCase().equals("B"))
259             {
260                 tintColor = new Color(c.getRed(),c.getGreen(),000);
261             }
262             int newColor = tintColor.getRGB();
263             newImage.setRGB(x, y, newColor);
264         }
265     }
266     return newImage;
267 }
268
269 *****/
270 Creates new image that is opposite color of old image
271 *****/
272 public static BufferedImage Inverse(BufferedImage img, int h, int w, String BD)
273 {
274
275     BufferedImage newImage = new BufferedImage(w, h, BufferedImage.TYPE_INT_RGB);
276
277     for(int y=0; y<h; y++)
278     {
279         for(int x=0; x<w; x++)
280         {
281
282
283             int rgb=img.getRGB(x, y);
284             int r = (rgb)&0xFF;
285             int g = (rgb>>8)&0xFF;
286             int b = (rgb>>16)&0xFF;
287
288             if (BD=="B")
289             {
290                 int brightR=r+10;
291                 int brightG=g+10;
292                 int brightB=b+10;
293

```

```

294         int Bcolor = brightR;
295         Bcolor = (Bcolor << 8) + brightG;
296         Bcolor = (Bcolor << 8) + brightB;
297
298         newImage.setRGB(h, w, Bcolor);
299     }
300
301     else
302     {
303         int darkR=r-10;
304         int darkG=g-10;
305         int darkB=b-10;
306
307         int Bcolor = darkR;
308         Bcolor = (Bcolor << 8) + darkG;
309         Bcolor = (Bcolor << 8) + darkB;
310
311         newImage.setRGB(x, y, Bcolor);
312     }
313
314 }
315 }
316 }
317 return newImage;
318 }
319
320 /*****
321 Creates new image with wildly changed colors.
322 *****/
323 public static BufferedImage Colorize(BufferedImage img, int h, int w)
324 {
325     BufferedImage newImage = new BufferedImage(w, h, BufferedImage.TYPE_INT_RGB);
326     for(int y=0; y<h; y++)
327     {
328         for(int x=0; x<w; x++)
329         {
330             int rgb=img.getRGB(x, y);
331             int r = (rgb&0xFF);
332             int g = (rgb>>8)&0xFF;
333             int b = (rgb>>16)&0xFF;
334
335             int newR= ((0-r)*-1)/2+r;
336             int newG= ((0-g)*-1)/2+g;
337             int newB= ((0-b)*-1)/2+b;
338
339             int fadeColor = newR;
340             fadeColor = (fadeColor << 8) + newG;
341             fadeColor = (fadeColor << 8) + newB;
342
343             newImage.setRGB(x, y, fadeColor);
344         }
345     }
346 }
347 return newImage;
348 }
349 }

```