```
 1 import java.lang.Math;
 2 import java.awt.image.BufferedImage;
 3 import java.nio.charset.Charset;
 4 import java.awt.Color;
 5
 6 /**************************************************************
 7 Steg contains methods for encrypting and decrypting messages in
 8 images, as well as other methods that aid in those tasks.
 9
10 @author Susanna Bradbury
11 @version 06/04/2014
12 **************************************************************/
13 public class Steg{
14
15    /**************************************************************
16    Encodes a message in an image.
17    **************************************************************/
18    public static BufferedImage encrypt(BufferedImage picture){
19       String message=Display.getText();
20       int width=picture.getWidth();
21       int newwidth=(width/3)*3;
22       int height=picture.getHeight();
23       BufferedImage encryptedImage = picture;
24       int x=width/2;
25       int y=height/2;
26       char[] bytes=message.toCharArray();
27       int[] bits=new int[(bytes.length*8)+3];
28       for (int c=0;c<(bytes.length);c++){
29          for (int d=7;d>=0;d--){
30             bits[(c*8)+d]=(bytes[c]&1);
31             bytes[c]>>>=1;
32          }
33       }
34       bits[bits.length-3]=0;
35       bits[bits.length-2]=0;
36       bits[bits.length-1]=0;
37       int length=(bits.length);
38       if (length>=10000000){
39          Display.eTimeErrorStart();
40       }
41       int rows=(length/3)/newwidth;
42       int extra=(length/3)%newwidth;
43       int startx=0;
44       int starty=y-(rows/2);
45       int totalSize=(newwidth*(height-2))*3;
46       int charNum=(length-totalSize)/8;
47       if (charNum>0){
48          //show popup notifying user that message is too long for image
49          //ask user to alter message or select larger image
50          Display.lengthError(charNum);
51          return picture;
52       }
53       int messloc=0;
54       for (int k=0;k<rows;k++){
55          startx=0;
56          for (int i=0;i<newwidth;i++){
57             Color c=new Color(encryptedImage.getRGB(startx,starty));
58             int red=c.getRed();
59             int green=c.getGreen();
```

```
            int blue=c.getBlue();
            int redbit=(red&1);
            int greenbit=(green&1);
            int bluebit=(blue&1);
            int messagebit=bits[messloc];
            int dif=redbit-messagebit;
            int newred=red-dif;
            messagebit=bits[messloc+1];
            dif=greenbit-messagebit;
            int newgreen=green-dif;
            messagebit=bits[messloc+2];
            dif=bluebit-messagebit;
            int newblue=blue-dif;
            Color n=new Color(newred,newgreen,newblue);
            int newcolor=n.getRGB();
            encryptedImage.setRGB(startx,starty,newcolor);
            startx++;
            messloc+=3;
         }
         starty++;
      }
      startx=0;
      for (int j=0;j<extra;j++){
         Color c=new Color(encryptedImage.getRGB(startx,starty));
         int red=c.getRed();
         int green=c.getGreen();
         int blue=c.getBlue();
         int redbit=(red&1);
         int greenbit=(green&1);
         int bluebit=(blue&1);
         int messagebit=bits[messloc];
         int dif=redbit-messagebit;
         int newred=red-dif;
         messagebit=bits[messloc+1];
         dif=greenbit-messagebit;
         int newgreen=green-dif;
         messagebit=bits[messloc+2];
         dif=bluebit-messagebit;
         int newblue=blue-dif;
         Color n=new Color(newred,newgreen,newblue);
         int newcolor=n.getRGB();
         encryptedImage.setRGB(startx,starty,newcolor);
         startx++;
         messloc+=3;
      }
      writeLength(encryptedImage,(length-3));
        if (length>=10000000){
         Display.eTimeErrorEnd();
      }
      return encryptedImage;
    }

    /*************************************************************
   Decodes a message from an image.
   *************************************************************/
   public static String decrypt(BufferedImage picture){
      String decryptedMessage = new String("");
      int width=picture.getWidth();
      int newwidth=(width/3)*3;
```

```
119       int height=picture.getHeight();
120       int x=width/2;
121       int y=height/2;
122       int length=(readLength(picture))+3;
123       if (length==0){
124          return "";
125          //show a popup stating that this image has not been encrypted
126
127       }
128       if (length>=600000){
129          if (Display.timeError()==0){
130             return null;
131          }
132       }
133       System.out.println(""+(length-3));
134       int rows=(length/3)/newwidth;
135       int extra=(length/3)%newwidth;
136       int startx=0;
137       int starty=y-(rows/2);
138       int[] bits=new int[length];
139       int messloc=0;
140       for (int k=0;k<rows;k++){
141          startx=0;
142          for (int i=0;i<newwidth;i++){
143             Color c=new Color(picture.getRGB(startx,starty));
144             int red=c.getRed();
145             int green=c.getGreen();
146             int blue=c.getBlue();
147             int redbit=(red&1);
148             int greenbit=(green&1);
149             int bluebit=(blue&1);
150             bits[messloc]=redbit;
151             bits[messloc+1]=greenbit;
152             bits[messloc+2]=bluebit;
153             startx++;
154             messloc+=3;
155          }
156          starty++;
157       }
158       if (extra>0){
159          startx=0;
160       }
161       for (int j=0;j<extra;j++){
162          Color c=new Color(picture.getRGB(startx,starty));
163             int red=c.getRed();
164             int green=c.getGreen();
165             int blue=c.getBlue();
166             int redbit=(red&1);
167             int greenbit=(green&1);
168             int bluebit=(blue&1);
169             bits[messloc]=redbit;
170             bits[messloc+1]=greenbit;
171             bits[messloc+2]=bluebit;
172             startx++;
173             messloc+=3;
174       }
175       char[] bytes=new char[length/8];
176       for (int c=0;c<bytes.length;c++){
177          bytes[c]=0;
```

```
178         for(int i=0;i<8;i++){
179            bytes[c]<<=1;
180            bytes[c]+=(char)bits[c*8+i];
181         }
182      }
183      for(int i=0;i<bytes.length;i++)
184      {
185         decryptedMessage=(decryptedMessage+bytes[i]);
186      }
187      return decryptedMessage;
188
189    }
190
191   /*************************************************************
192   Writes the length of the encrypted message to the first row of
193   the image to enable decryption.
194   *************************************************************/
195   public static BufferedImage writeLength(BufferedImage picture, int lengthVal){
196      String value=(lengthVal+"stop");
197      int size=value.length();
198      int width=picture.getWidth();
199      int x=0;
200      int y=0;
201      char[] bytes=value.toCharArray();
202      int[] bits=new int[bytes.length*8];
203      for (int c=0;c<(bytes.length);c++){
204         for (int d=7;d>=0;d--){
205            bits[(c*8)+d]=(bytes[c]&1);
206            bytes[c]>>>=1;
207         }
208      }
209      int length=bits.length;
210      int iter=length/3;
211      int iterPlus=length%3;
212      int messloc=0;
213      for (int i=0;i<(iter);i++){
214         Color c=new Color(picture.getRGB(x,y));
215         int red=c.getRed();
216         int green=c.getGreen();
217         int blue=c.getBlue();
218         int redbit=(red&1);
219         int greenbit=(green&1);
220         int bluebit=(blue&1);
221         int valuebit=bits[messloc];
222         int dif=redbit-valuebit;
223         int newred=red-dif;
224         valuebit=bits[messloc+1];
225         dif=greenbit-valuebit;
226         int newgreen=green-dif;
227         valuebit=bits[messloc+2];
228         dif=bluebit-valuebit;
229         int newblue=blue-dif;
230         Color n=new Color(newred,newgreen,newblue);
231         int newcolor=n.getRGB();
232         picture.setRGB(x,y,newcolor);
233         x++;
234         messloc+=3;
235      }
236      if (iterPlus==1){
```

```
         Color c=new Color(picture.getRGB(x,y));
         int red=c.getRed();
         int green=c.getGreen();
         int blue=c.getBlue();
         int redbit=(red&1);
         int valuebit=bits[messloc];
         int dif=redbit-valuebit;
         int newred=red-dif;
         Color n=new Color(newred,green,blue);
         int newcolor=n.getRGB();
         picture.setRGB(x,y,newcolor);
      }
      if (iterPlus==2){
         Color c=new Color(picture.getRGB(x,y));
         int red=c.getRed();
         int green=c.getGreen();
         int blue=c.getBlue();
         int redbit=(red&1);
         int greenbit=(green&1);
         int valuebit=bits[messloc];
         int dif=redbit-valuebit;
         int newred=red-dif;
         valuebit=bits[messloc+1];
         dif=greenbit-valuebit;
         int newgreen=green-dif;
         Color n=new Color(newred,newgreen,blue);
         int newcolor=n.getRGB();
         picture.setRGB(x,y,newcolor);
      }
   return picture;
   }

    /*************************************************************
   Scans the first row of the image to determine the length of the
   encrypted message.
   *************************************************************/
   public static int readLength(BufferedImage picture){
      int decryptedLength=0;
      int width=picture.getWidth();
      int x=0;
      int y=0;
      int[] bits=new int[width*3];
      int messloc=0;
      for (int i=0;i<(width);i++){
         Color c=new Color(picture.getRGB(x,y));
         int red=c.getRed();
         int green=c.getGreen();
         int blue=c.getBlue();
         int redbit=(red&1);
         int greenbit=(green&1);
         int bluebit=(blue&1);
         bits[messloc]=redbit;
         bits[messloc+1]=greenbit;
         bits[messloc+2]=bluebit;
         x++;
         messloc+=3;
      }
      char[] bytes=new char[(width*3)/8];
      for (int c=0;c<bytes.length;c++){
```

```
296          bytes[c]=0;
297          for(int i=0;i<8;i++){
298             bytes[c]<<=1;
299             bytes[c]+=(char)bits[c*8+i];
300          }
301       }
302       String firstLine="";
303       String possibleStop="";
304       for(int i=0;i<bytes.length;i++)
305       {
306          firstLine=(firstLine+bytes[i]);
307          if (firstLine.contains("stop")){
308             break;
309          }
310       }
311       if (firstLine.length()>15){
312          return 0;
313       }
314       int place=firstLine.length()-4;
315       String lineMinusStop=firstLine.substring(0,place);
316       decryptedLength=Integer.parseInt(lineMinusStop);
317       return decryptedLength;
318    }
319 }
```