```
import java.lang.Math;
import java.awt.image.BufferedImage;
import java.nio.charset.Charset;
import java.awt.Color;

/**************************************************************
Steg contains methods for encrypting and decrypting messages in
images, as well as other methods that aid in those tasks.

@author Susanna Bradbury
@version 06/04/2014
**************************************************************/
public class Steg{

    /**************************************************************
    Encodes a message in an image.
    **************************************************************/
    public static BufferedImage encrypt(BufferedImage picture){
        String message=Display.getText();
        int width=picture.getWidth();
        int newwidth=(width/3)*3;
        int height=picture.getHeight();
        BufferedImage encryptedImage = picture;
        int x=width/2;
        int y=height/2;
        char[] bytes=message.toCharArray();
        int[] bits=new int[(bytes.length*8)+3];
        for (int c=0;c<(bytes.length);c++){
            for (int d=7;d>=0;d--){
                bits[(c*8)+d]=(bytes[c]&1);
                bytes[c]>>>=1;
            }
        }
        bits[bits.length-3]=0;
        bits[bits.length-2]=0;
        bits[bits.length-1]=0;
        int length=(bits.length);
        if (length>=10000000){
            Display.eTimeErrorStart();
        }
        int rows=(length/3)/newwidth;
        int extra=(length/3)%newwidth;
        int startx=0;
        int starty=y-(rows/2);
        int totalSize=(newwidth*(height-2))*3;
        int charNum=(length-totalSize)/8;
        if (charNum>0){
            //show popup notifying user that message is too long for image
            //ask user to alter message or select larger image
            Display.lengthError(charNum);
            return picture;
        }
        int messloc=0;
        for (int k=0;k<rows;k++){
```

```
            startx=0;
            for (int i=0;i<newwidth;i++){
                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;
            }
            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;
        int height=picture.getHeight();
        int x=width/2;
        int y=height/2;
        int length=(readLength(picture))+3;
        if (length==0){
            return "";
            //show a popup stating that this image has not been encrypted

        }
        if (length>=600000){
            if (Display.timeError()==0){
                return null;
            }
        }
        System.out.println(""+(length-3));
        int rows=(length/3)/newwidth;
        int extra=(length/3)%newwidth;
        int startx=0;
        int starty=y-(rows/2);
        int[] bits=new int[length];
        int messloc=0;
        for (int k=0;k<rows;k++){
            startx=0;
            for (int i=0;i<newwidth;i++){
                Color c=new Color(picture.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);
                bits[messloc]=redbit;
                bits[messloc+1]=greenbit;
                bits[messloc+2]=bluebit;
                startx++;
                messloc+=3;
            }
            starty++;
        }
        if (extra>0){
            startx=0;
        }
        for (int j=0;j<extra;j++){
            Color c=new Color(picture.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);
                bits[messloc]=redbit;
                bits[messloc+1]=greenbit;
                bits[messloc+2]=bluebit;
                startx++;
                messloc+=3;
        }
        char[] bytes=new char[length/8];
        for (int c=0;c<bytes.length;c++){
            bytes[c]=0;
            for(int i=0;i<8;i++){
                bytes[c]<<=1;
                bytes[c]+=(char)bits[c*8+i];
            }
        }
        for(int i=0;i<bytes.length;i++)
        {
            decryptedMessage=(decryptedMessage+bytes[i]);
        }
        return decryptedMessage;

    }

    /*************************************************************
    Writes the length of the encrypted message to the first row of
    the image to enable decryption.
    *************************************************************/
    public static BufferedImage writeLength(BufferedImage picture, int lengthVal){
        String value=(lengthVal+"stop");
        int size=value.length();
        int width=picture.getWidth();
        int x=0;
        int y=0;
        char[] bytes=value.toCharArray();
        int[] bits=new int[bytes.length*8];
        for (int c=0;c<(bytes.length);c++){
            for (int d=7;d>=0;d--){
                bits[(c*8)+d]=(bytes[c]&1);
                bytes[c]>>>=1;
            }
        }
        int length=bits.length;
        int iter=length/3;
        int iterPlus=length%3;
        int messloc=0;
        for (int i=0;i<(iter);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);
            int valuebit=bits[messloc];
            int dif=redbit-valuebit;
            int newred=red-dif;
            valuebit=bits[messloc+1];
            dif=greenbit-valuebit;
            int newgreen=green-dif;
            valuebit=bits[messloc+2];
            dif=bluebit-valuebit;
            int newblue=blue-dif;
            Color n=new Color(newred,newgreen,newblue);
            int newcolor=n.getRGB();
            picture.setRGB(x,y,newcolor);
            x++;
            messloc+=3;
        }
        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++){
            bytes[c]=0;
            for(int i=0;i<8;i++){
                bytes[c]<<=1;
                bytes[c]+=(char)bits[c*8+i];
            }
        }
        String firstLine="";
        String possibleStop="";
        for(int i=0;i<bytes.length;i++)
        {
            firstLine=(firstLine+bytes[i]);
            if (firstLine.contains("stop")){
                break;
            }
        }
        if (firstLine.length()>15){
            return 0;
        }
        int place=firstLine.length()-4;
        String lineMinusStop=firstLine.substring(0,place);
        decryptedLength=Integer.parseInt(lineMinusStop);
        return decryptedLength;
    }
}
```