```
import java.awt.Desktop;
import java.awt.Toolkit;
import java.awt.datatransfer.*;
import java.util.*;
import java.io.*;
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import javax.swing.JOptionPane;

/*************************************************************
Tasks are the methods called for by Display
through the GUI.

@author Susanna Bradbury, James Houghton, Pranav Ramanan
@version 06/05/2014
*************************************************************/
public class Tasks
{
    /*************************************************************
    Location of where the most recent saved file was saved.
    *************************************************************/
    public static String saveLocation=null;

    /* Pranav's methods */
    /*************************************************************
    Inverts the loaded image's colors.
    *************************************************************/
    public static void task1()
    {
        System.out.println(Display.getText());
        BufferedImage image=Display.getImage();

        if(Display.getText()==null)
        {
            String message = Steg.decrypt(image);
            Display.setEncodedText(message);
        }
        image = Effects.Inverse(image,Display.getHeight(),Display.getWidth());
        Display.unloadImage();
        Display.loadBI(image, 1);
        image = Steg.encrypt(image);
        Display.loadBI(image, 2);
    }
    /*************************************************************
    Fades the colors of the image.
    *************************************************************/
    public static void task2()
    {
        BufferedImage image=Display.getImage();

        if(Display.getText()==null)
        {
            String message = Steg.decrypt(image);
            Display.setEncodedText(message);
```

```
        }
        String color = JOptionPane.showInputDialog("R | G | B | ALL");
        if ((color != null) && (color.length() > 0))
        {
            try
            {
                String tintString = JOptionPane.showInputDialog("Decrease color value by:
                (0-225)");
                if ((tintString!=null) && (tintString.length()>0))
                {
                    int tint = Integer.parseInt(tintString);
                    image = Effects.Fade(image,Display.getHeight(),Display.getWidth(),tint,color
                    );
                    Display.unloadImage();
                    Display.loadBI(image, 1);
                    image = Steg.encrypt(image);
                    Display.loadBI(image, 2);
                }
            }
            catch(Exception e){}
        }
    }
    /*************************************************************
    Tints the image's colors.
    *************************************************************/
    public static void task3()
    {
        BufferedImage image=Display.getImage();

        if(Display.getText()==null)
        {
            String message = Steg.decrypt(image);
            Display.setEncodedText(message);
        }

        String color = JOptionPane.showInputDialog("R | G | B | ALL");
        if ((color != null) && (color.length() > 0))
        {
            try
            {
                String tintString = JOptionPane.showInputDialog("Increase color value by:
                (0-225)");
                if ((tintString!=null) && (tintString.length()>0))
                {
                    int tint = Integer.parseInt(tintString);
                    image = Effects.Tint(image,Display.getHeight(),Display.getWidth(),tint,color
                    );
                    Display.unloadImage();
                    Display.loadBI(image, 1);
                    image = Steg.encrypt(image);
                    Display.loadBI(image, 2);
                }
            }
            catch(Exception e){}
```

```
    }
}
/*****************************************************************
Makes the image a mix of only black and white color.
*****************************************************************/
public static void task4()
{
    BufferedImage image=Display.getImage();

    if(Display.getText()==null)
    {
        String message = Steg.decrypt(image);
        Display.setEncodedText(message);
    }

    image = Effects.BlackWhite(image,Display.getHeight(),Display.getWidth());
    Display.unloadImage();
    Display.loadBI(image, 1);
    image = Steg.encrypt(image);
    Display.loadBI(image, 2);
}
/*****************************************************************
Removes a specified color in the loaded image.
*****************************************************************/
public static void task5()
{
    BufferedImage image=Display.getImage();

    if(Display.getText()==null)
    {
        String message = Steg.decrypt(image);
        Display.setEncodedText(message);
    }

    String color = JOptionPane.showInputDialog("R|G|B");
    if ((color != null) && (color.length() > 0))
    {
    image = Effects.Remove(image,Display.getHeight(),Display.getWidth(),color);
    Display.unloadImage();
    Display.loadBI(image, 1);
    image = Steg.encrypt(image);
    Display.loadBI(image, 2);
    }
}
/*****************************************************************
Grayscales the image.
*****************************************************************/
public static void task6()
{
    BufferedImage image=Display.getImage();

    if(Display.getText()==null)
    {
        String message = Steg.decrypt(image);
```

```
            Display.setEncodedText(message);
        }

        image = Effects.Grayscale(image,Display.getHeight(),Display.getWidth());
        Display.unloadImage();
        Display.loadBI(image, 1);
        image = Steg.encrypt(image);
        Display.loadBI(image, 2);
    }
    /**************************************************************
    'Colorizes' the image. Dramatically changes the colors of the image.
    **************************************************************/
    public static void task7()
    {
        BufferedImage image=Display.getImage();

        if(Display.getText()==null)
        {
            String message = Steg.decrypt(image);
            Display.setEncodedText(message);
        }
        image = Effects.Colorize(image,Display.getHeight(),Display.getWidth());
        Display.unloadImage();
        Display.loadBI(image, 1);
        image = Steg.encrypt(image);
        Display.loadBI(image, 2);
    }


    /* Susanna's methods */
    /**************************************************************
    Encodes the loaded image.
    **************************************************************/
    public static void encode()
    {
        Display.setEncodedText(Display.getText(1));
        BufferedImage eImage = Steg.encrypt(Display.getImage());
        Display.unloadImage();
        Display.loadBI(eImage, 2);

    }
    /**************************************************************
    Decodes the loaded image.
    **************************************************************/
    public static void decode()
    {
        String message = Steg.decrypt(Display.getImage());
        System.out.println(message);
        if(message.equals(""))
        {
            Display.blankError();
        }
        else
            Display.setText(message);
```

```
    }

    /* James' methods */
    /*****************************************************************
    Opens a new image.
    *****************************************************************/
    public static void open()
    {
        Display.loadImage("");
    }
    /*****************************************************************
    Saves the loaded image.
    *****************************************************************/
    public static void save()
    {
        if(saveLocation!=null)
        {
        try {
            BufferedImage image=Display.getImage(1);
            File outputFile = new File(saveLocation);
            ImageIO.write(image,"png",outputFile);
        }
        catch (IOException e){}
        }
        else
            saveAs();
    }
    /*****************************************************************
    Saves the loaded image as a specific file in a specific place.
    *****************************************************************/
    public static void saveAs()
    {
        String filename = Display.fnSave();
        if(filename!=null)
        {
            boolean success=false;
            while (success==false){
            try {
                BufferedImage image=Display.getImage(1);
                if(filename.length()>3)
             {
                   String extension=filename.substring(filename.length()-4).toLowerCase();
                if(!extension.equals(".png"))
                {
                   filename+=".png";
                }
             }
                else filename+=".png";
                saveLocation=filename;
                File outputFile = new File(filename);
                ImageIO.write(image,filename.substring(filename.length()-3),outputFile);
                success=true;
            }
```

```
            catch (IOException e){filename = JOptionPane.showInputDialog("Try again.");}}
        }
    }

    /*****************************************************************
    Displays the user manual to assist the user.
    *****************************************************************/
    public static void help()
    {
        Resources.showHelp();
    }

    /*****************************************************************
    Displays a dialog containing information about CRYPTICON
    and its developers.
    *****************************************************************/
    public static void info()
    {
        Resources.displayInfo();
    }
    /*****************************************************************
    Sets the last save location to a global variable, making it
    accessible when needed.
    *****************************************************************/
    public static void setLoadedImage(String filename)
    {
        saveLocation=filename;
    }
    /*****************************************************************
    Copies text in Display's output text field to the system's clipboard.
    *****************************************************************/
    public static void copy()
    {
        String output = Display.getOutput();
        StringSelection stringSelection = new StringSelection(output);
        Clipboard clipboard = Toolkit.getDefaultToolkit().getSystemClipboard();
        clipboard.setContents(stringSelection,null);
    }
}
```