Using the Java 2D ColorConvertOp and RescaleOp Filter Classes to Process Images

Learn how to write programs that use the RescaleOp and ColorConvertOp classes of the Java 2D API to perform a variety of filtering operations on images.

Published: September 11, 2007
By Richard G. Baldwin

Java Programming Notes # 462

- Preface
  - General
  - Viewing tip
    - Figures
    - Listings
  - Supplementary material
- General background information
  - Constructing images
  - The framework program named ImgMod05
  - RescaleOp examples
  - ColorConvertOp example
- Preview
- Discussion and sample code
- Run the program
- Summary
- Complete program listing
- Copyright
- Resources
- About the author

Preface

General

In an earlier lesson titled "A Framework for Experimenting with Java 2D Image-Processing Filters" *(see Resources)*, I taught you how to write a framework program that makes it easy to use the image-filtering classes of the Java 2D API to process the pixels in an image and to display the processed image.

At the close of that lesson, I told you that future lessons would teach you how to use the following image-filtering classes from the Java 2D API:

- **LookupOp**
- **AffineTransformOp**
- **BandCombineOp**
- **ConvolveOp**
- **RescaleOp**
- **ColorConvertOp**

In several of the previous lessons listed in the Resources section, I taught you how to use the **LookupOp**, **AffineTransformOp**, **BandCombineOp**, and **ConvolveOp** image-filtering classes.

In this final lesson of the series, I will teach you how to use the **RescaleOp** and **ColorConvertOp** image-filtering classes.

Viewing tip

I recommend that you open another copy of this document in a separate browser window and use the following links to easily find and view the figures and listings while you are reading about them.

Figures

- Figure 1. Image with all three colors inverted.
- Figure 2. Image with only one color inverted.
- Figure 3. Use of RescaleOp to adjust contrast and brightness.
- Figure 4. Conversion of color space to CS_GRAY.
- Figure 5. User input GUI with the ColorConvertOp tab selected.
- Figure 6. User input GUI with the RescaleOp tab selected.

Listings

- Listing 1. Beginning of the class definition.
- Listing 2. Creation of components used to construct the ColorConvertOp page.
- Listing 3. Creation of components used to construct the RescaleOp page.
- Listing 4. The primary constructor.
- Listing 5. The constructColorConvertPage method.
- Listing 6. The processColorConvertPage method.
- Listing 7. The constructColorRescalePage method.
- Listing 8. The processColorRescalePage method.
- Listing 9. The processImg method.
- Listing 10. Complete listing of the ImgMod43 class.

Supplementary material

I recommend that you also study the other lessons in my extensive collection of online Java tutorials. You will find a consolidated index at www.DickBaldwin.com.

General background information

Constructing images

Before getting into the programming details, it may be useful for you to review the concept of how images are constructed, stored, transported, and rendered in Java *(and in most modern computers for that matter).* I provided a great deal of information on these topics in the earlier lesson titled "Processing Image Pixels using Java, Getting Started" *(see Resources).* Therefore, I won't repeat that information here. Rather, I will simply refer you back to the earlier lesson.

The framework program named ImgMod05

It will also be useful for you to understand the behavior of the framework program named **ImgMod05**. Therefore, I strongly recommend that you study the earlier lesson titled "A Framework for Experimenting with Java 2D Image-Processing Filters" *(see Resources).*

However, if you don't have the time to do that, you should take a look at the earlier lesson titled "Using the Java 2D LookupOp Filter Class to Process Images" *(see Resources)*, in which I summarized the behavior of the framework program named **ImgMod05**.

RescaleOp examples

The **RescaleOp** class can be used to multiply the color value for each pixel by a user-specified scale factor, and then to add a user-specified constant to the product. Separate scale factors and additive constants are provided for each of the red, green, and blue colors.

Color values that fall outside the allowable range from 0 to 255 are simply clipped to 0 and 255.

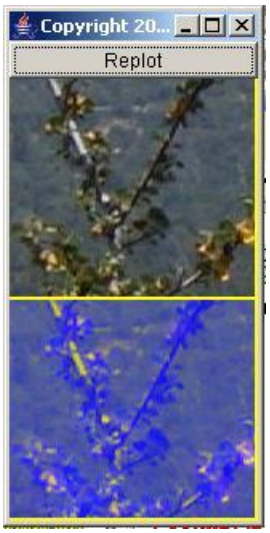
Color inversion

Several previous lessons have discussed inverting the colors in an image. This is another class that makes it easy to invert the colors. In this case, the scale factor for all three bands would be set to -1 and the additive constant would be set to 255. This would produce an inverted output image like that shown in Figure 1.

Figure 1. Image with all three colors inverted.

Just in case you have a need to do so, each color band can be inverted separately producing results like those shown in Figure 2. Figure 2 shows the result of inverting only the blue pixels in an image without modifying the red and green pixels.

Figure 2. Image with only one color inverted.



Adjusting the contrast and brightness

Perhaps more importantly, the **RescaleOp** class can also be used to adjust the contrast and brightness of an image using the concepts that I explained in the earlier lesson titled "Processing Image Pixels Using Java: Controlling Contrast and Brightness" *(see Resources)*.

Figure 3 shows the result of using the **RescaleOp** class to improve the contrast and brightness of the input image. In this case, each color value was multiplied by 3 and then a value of -160 was added to each product.

Figure 3. Use of RescaleOp to adjust contrast and brightness.

The statistical changes

Multiplying the color values by a scale factor widens the distribution as shown in the earlier lesson. This increases the contrast. Adding the constant adjusts the mean value, thus modifying the brightness. *(See the histogram in Figure 2 in the earlier lesson.)*

Note that although the methodology used here isn't exactly the same as that used in the earlier lesson, the result shown in Figure 3 above compares favorably with Figure 1 in the earlier lesson indicating that the standard deviation and the mean for the output image in Figure 3 is probably very similar to the standard deviation and the mean for the output image shown in Figure 1 in the earlier lesson.

Assessment

As demonstrated in the earlier lesson, it is not difficult to write your own program to replicate the behavior of the **RescaleOp** class. However, if the **RescaleOp** class will serve your needs, use it, don't reinvent it.

One major difference

There is, however, one aspect of my implementation in the earlier lesson that I consider to be superior to the implementation of the **RescaleOp** class, particularly when used for the purpose of adjusting the contrast and brightness of an image.

The mean value modification in my earlier implementation is specified by the user as a multiplier, such as 1.25. This would, for example, cause the new mean value to be 1.25 times greater than the old mean value.

With the **RescaleOp** class, a constant must be added or subtracted from the product in order to move the mean value. You usually won't know what the actual mean value is, so you will have to do a lot of guesswork in order to determine the proper additive value.

On the other hand, the **RescaleOp** class can be used for other purposes *(such as color inversion)* where an additive constant is more appropriate than a multiplicative factor so it is a more general implementation.

ColorConvertOp example

The apparent purpose of this class is to make it possible for you to convert an image from one **ColorSpace** to another **ColorSpace**.

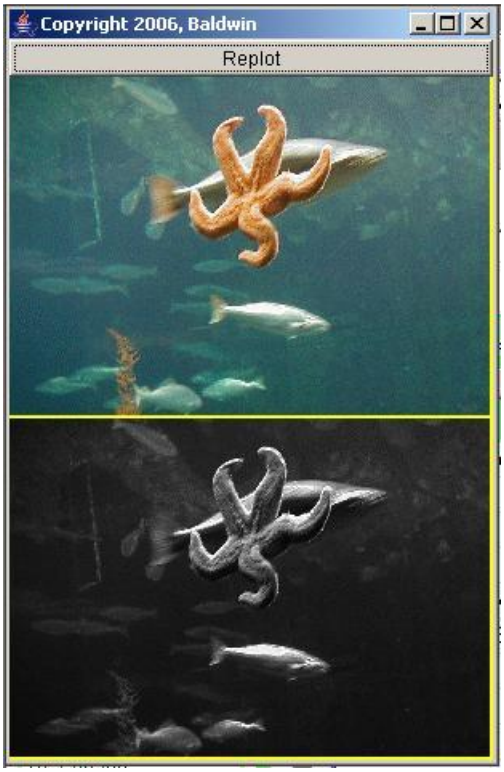
> *(I will leave it up to you to go to the Sun documentation to learn about the **ColorSpace** class.)*

In any event, when deciding upon a new color space, several possibilities are available. They are defined as constants in the **ColorSpace** class.

Conversion to grayscale

Figure 4 coverts the color space of the input image to type **ColorSpace.CS_GRAY**. As you can see, this changed the image from a color image to what would probably be called a grayscale image.

Figure 4. Conversion of color space to CS_GRAY.



Assessment

In my opinion, writing your own program to replicate the behavior of the **ColorConvertOp** class would be very difficult. If you need this capability, by all means, use the **ColorConvertOp** class and don't attempt to reinvent it.

A caveat

This is the one case that I have found where programs that use the image-filtering classes of the Java 2D API are incompatible with the framework program named **ImgMod05**. If you modify the color space of an image, the code in **ImgMod05** that attempts to write the output image into a JPEG file will throw an error.

Preview

General comments
General comments regarding the uses of classes such as this one can be found in the class named **ImgMod038** in the lesson named "Using the Java 2D LookupOp Filter Class to Process Images" *(see*

In this lesson, I will present and explain a program named **ImgMod43** that illustrates the use of the **ColorConvertOp** and **RescaleOp** classes of the Java 2D API.

Resources).

Except for problems encountered when writing the output jpeg file described below, this class is compatible with the use of the driver program named **ImgMod05**.

Writing an output jpeg file

The driver program named **ImgMod05** displays the original and the modified images in the format shown in Figure 1. It also writes the modified image into an output file in JPEG format. The name of the output file is **junk.jpg** and it is written into the current directory. Note however that changing the type of the **ColorSpace** using **ColorConvertOp** causes a failure in the attempt to write the processed image into an output jpeg file in **ImgMod05**. In two cases, that code throws an exception and doesn't write the file. In the third case, the output file doesn't contain a valid copy of the processed image.

A user input GUI

Image processing programs such as this one may provide a GUI for data input making it possible for the user to modify the behavior of the image processing method each time the **Replot** button *(shown at the top of Figure 1)* is clicked.

This program creates a user input GUI consisting of a tabbed pane containing two pages. The tabs on the pages are labeled:

- **ColorConvertOp**
- **RescaleOp**

Each page contains a set of controls that make it possible to process the image in a way that illustrates the processing concept indicated by the labels on the tabs.

Figure 5 shows a screen shot of the user input GUI for this program with the **ColorConvertOp** tab selected.

Figure 5. User input GUI with the ColorConvertOp tab selected.

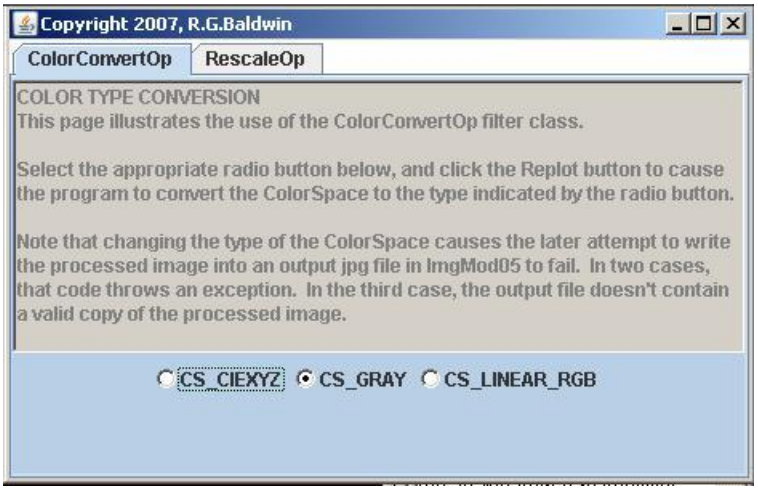
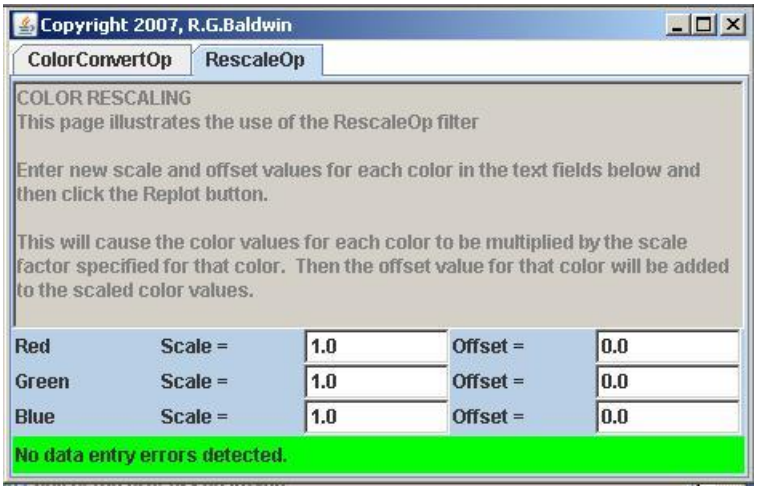


Figure 6 shows a screen shot of the user input GUI with the **RescaleOp** tab selected.

Figure 6. User input GUI with the RescaleOp tab selected.



Usage

Enter the following at the command line to run this program:

```
java ImgMod05 ImgMod43 ImageFileName
```

If the program is unable to load the image file within ten seconds, it will abort with an error message.

The program was tested using J2SE 6 under WinXP.

Discussion and sample code

Will discuss in fragments

A complete listing of this class is presented in Listing 10 near the end of the lesson. As is my custom, I will present and explain this class in fragments.

Beginning of the class definition

The class definition begins in the first program fragment shown in Listing 1. Note that it is necessary for this class to implement the interface named **ImgIntfc05** in order to be compatible with the driver program named **ImgMod05**.

Listing 1. Beginning of the class definition.

```
class ImgMod43 extends Frame implements
ImgIntfc05{

  JTabbedPane tabbedPane = new JTabbedPane();
```

Listing 1 declares and instantiates a **JTabbedPane** object, which is the primary container used to construct the GUI as shown in Figures 5 and 6.

Creation of components used to construct the ColorConvertOp page

Listing 2 creates some of the components that are used to construct the **ColorConvertOp** page shown in Figure 5. Note that other required components are created locally closer to where they are needed.

Listing 2. Creation of components used to construct the ColorConvertOp page.

```
  Panel colorConvertPage = new Panel();
  CheckboxGroup buttonGroup = new
CheckboxGroup();
  Checkbox CS_CIEXYZ =
              new
Checkbox("CS_CIEXYZ",buttonGroup,false);
  Checkbox CS_GRAY =
                 new
Checkbox("CS_GRAY",buttonGroup,true);
  Checkbox CS_LINEAR_RGB =
           new
Checkbox("CS_LINEAR_RGB",buttonGroup,false);
```

Creation of components used to construct the RescaleOp page

Listing 3 creates some of the components that are required to construct the **RescaleOp** page shown in Figure 6. Once again, other required components are created locally closer to where they are needed.

Listing 3. Creation of components used to construct the RescaleOp page.

```
  Panel colorRescalePage = new Panel();
  TextField redScaleField = new
TextField("1.0");
  TextField redOffsetField = new
TextField("0.0");
  TextField greenScaleField = new
TextField("1.0");
  TextField greenOffsetField = new
TextField("0.0");
  TextField blueScaleField = new
TextField("1.0");
  TextField blueOffsetField = new
TextField("0.0");

  //The following Label is used to notify of
data entry
  // errors.
  String okMessage = "No data entry errors
detected.";
  Label errorMsg = new Label(okMessage);
```

The primary constructor

The primary constructor is shown in Listing 4. It calls other methods to separate the construction of the GUI into easily understandable units. Each method that it calls constructs one page in the tabbed pane.

Listing 4. The primary constructor.

```
  ImgMod43(){//constructor

    //Construct the pages and add them to the
tabbed pane.
    constructColorConvertPage();
    tabbedPane.add(colorConvertPage);

    constructColorRescalePage();
    tabbedPane.add(colorRescalePage);

    add(tabbedPane);//Add tabbedPane to the
Frame.

    setTitle("Copyright 2007, R.G.Baldwin");
    setBounds(555,0,470,300);
    setVisible(true);

    //Define a WindowListener to terminate the
program.
    addWindowListener(
      new WindowAdapter(){
        public void windowClosing(WindowEvent
```

```
e){
          System.exit(1);
        }//end windowClosing
      }//end windowAdapter
    );//end addWindowListener
  }//end constructor
```

The constructColorConvertPage method

Listing 5 shows the method that is used to construct the page in the tabbed pane shown in Figure 5. This method is called from the primary constructor to actually construct the page.

Listing 5. The constructColorConvertPage method.

```
  void constructColorConvertPage(){

colorConvertPage.setName("ColorConvertOp");//Tab
label
    colorConvertPage.setLayout(new
BorderLayout());

    //Create and add the instructional text to the
page.
    // This text appears in a disabled text area
at the
    // top of the page in the tabbed pane.
    String text ="COLOR TYPE CONVERSION\n"
      + "This page illustrates the use of the "
      + "ColorConvertOp filter class.\n\n"
      + "Select the appropriate radio button
below, and "
      + "click the Replot button to cause the
program to "
      + "convert the ColorSpace to the type
indicated by "
      + "the radio button.\n\n"
      + "Note that changing the type of the
ColorSpace "
      + "causes the later attempt to write the
processed "
      + "image into an output jpg file in ImgMod05
to "
      + "fail.  In two cases, that code throws an
"
      + "exception.  In the third case, the output
file "
      + "doesn't contain a valid copy of the
processed "
      + "image.";

    //Note:  The number of columns specified for
the
    // following TextArea is immaterial because
the
```

```
    // TextArea object is placed in the NORTH
location of
    // a BorderLayout.
    TextArea textArea = new TextArea(text,10,1,

TextArea.SCROLLBARS_NONE);

colorConvertPage.add(textArea,BorderLayout.NORTH);
    textArea.setEnabled(false);

    //Construct the control panel and add it to
the page.
    Panel controlPanel = new Panel();
    controlPanel.add(CS_CIEXYZ);
    controlPanel.add(CS_GRAY);
    controlPanel.add(CS_LINEAR_RGB);
    colorConvertPage.add(

controlPanel,BorderLayout.CENTER);
  }//end constructColorConvertPage
```

The code in Listing 5 is straightforward and shouldn't require further explanation.

The processColorConvertPage method

The method named **processColorConvertPage** is shown in Listing 6. This method processes the image according to the radio buttons shown in Figure 5.

This method uses the **ColorConvertOp** filter class to process the image and to convert it to the **ColorSpace** type indicated by a radio button selected by the user.

This method illustrates only three of the many **ColorSpace** types defined in the **ColorSpace** class. It is called from within a **switch** statement in the method named **processImg**, which is the primary image processing method in this program. The **processImg** method is called by the driver program named **ImgMod05**.

Listing 6. The processColorConvertPage method.

```
  BufferedImage processColorConvertPage(

BufferedImage theImage){
    //Examine the radio buttons.  Cause the
ColorSpace of
    // the image to be converted to the type
indicated
    // by the radio button.
    int colorSpaceType;
    if(CS_CIEXYZ.getState() == true){
      colorSpaceType = ColorSpace.CS_CIEXYZ;
    }else if(CS_GRAY.getState() == true){
      colorSpaceType = ColorSpace.CS_GRAY;
```

```
    }else{//CS_LINEAR_RGB must be selected
      colorSpaceType =
ColorSpace.CS_LINEAR_RGB;
    }//end else

    //Create the filter object.
    ColorConvertOp filterObj = new
ColorConvertOp(

ColorSpace.getInstance(colorSpaceType),null);

    //Apply the filter and return the result.
    return filterObj.filter(theImage,null);

  }//end processColorConvertPage
```

Except for the instantiation of the **ColorConvertOp** object *(as opposed to instantiating an object of one of the other filter classes defined in the Java 2D API)*, there is nothing in Listing 6 that is new to this lesson. Therefore, the code in Listing 6 shouldn't require further explanation provided that you take a look at the constructor for the **ColorConvertOp** class in the Sun documentation.

The constructColorRescalePage method

Listing 7 shows the method that is used to construct the page shown in Figure 6. This method is called from the primary constructor.

Listing 7. The constructColorRescalePage method.

```
  void constructColorRescalePage(){
    colorRescalePage.setName("RescaleOp");//Tab
label.
    colorRescalePage.setLayout(new
BorderLayout());

    //Create and add the instructional text to the
page.
    String text = "COLOR RESCALING\n"
      + "This page illustrates the use of the
RescaleOp "
      + "filter\n\n"
      + "Enter new scale and offset values for
each "
      + "color in the text fields below and then
click "
      + "the Replot button.\n\n"
      + "This will cause the color values for each
color "
      + "to be multiplied by the scale factor
specified "
      + "for that color.  Then the offset value
for that "
```

```
      + "color will be added to the scaled color
values.";

    //Note:  The number of columns specified for
the
    // following TextArea is immaterial because
the
    // TextArea object is placed in the NORTH
location of
    // a BorderLayout.
    TextArea textArea = new TextArea(text,9,1,

TextArea.SCROLLBARS_NONE);

colorRescalePage.add(textArea,BorderLayout.NORTH);
    textArea.setEnabled(false);

    //Construct the control panel and add it to
the page.
    Panel controlPanel = new Panel();
    controlPanel.setLayout(new GridLayout(3,5));
    controlPanel.add(new Label("Red"));
    controlPanel.add(new Label("Scale = "));
    controlPanel.add(redScaleField);
    controlPanel.add(new Label("Offset = "));
    controlPanel.add(redOffsetField);

    controlPanel.add(new Label("Green"));
    controlPanel.add(new Label("Scale = "));
    controlPanel.add(greenScaleField);
    controlPanel.add(new Label("Offset = "));
    controlPanel.add(greenOffsetField);

    controlPanel.add(new Label("Blue"));
    controlPanel.add(new Label("Scale = "));
    controlPanel.add(blueScaleField);
    controlPanel.add(new Label("Offset = "));
    controlPanel.add(blueOffsetField);

    colorRescalePage.add(

controlPanel,BorderLayout.CENTER);

    //Add the errorMsg label.

colorRescalePage.add(errorMsg,BorderLayout.SOUTH);
    errorMsg.setBackground(Color.GREEN);

  }//end constructColorRescalePage
```

Although Listing 7 is rather long and tedious, there is nothing in Listing 7 that is new to this lesson. Therefore, a further explanation of Listing 7 should not be required.

The processColorRescalePage method

This method processes the image according to the controls shown on the page in Figure 6. This method is called from within the **switch** statement in the method named **processImg**.

This method uses the scale and offset values specified by the user to modify the red, green, and blue color values. The new color value for each pixel is the old value multiplied by the scale factor for that color plus the offset value for that color.

Listing 8. The processColorRescalePage method.

```
  BufferedImage processColorRescalePage(

BufferedImage theImage){

    //Reset the error message to the default.
    errorMsg.setText(okMessage);
    errorMsg.setBackground(Color.GREEN);

    //Create the arrays required to contain
the scale and
    // offset values.
    float[] scale = new float[3];
    float[] offset = new float[3];

    //Populate the scale and offset arrays
using data from
    // the text fields.
    try{//Get input value from the text field.
      scale[0] =

Float.parseFloat(redScaleField.getText());
      offset[0] =

Float.parseFloat(redOffsetField.getText());
      scale[1] =

Float.parseFloat(greenScaleField.getText());
      offset[1] =

Float.parseFloat(greenOffsetField.getText());
      scale[2] =

Float.parseFloat(blueScaleField.getText());
      offset[2] =

Float.parseFloat(blueOffsetField.getText());
    }catch(java.lang.NumberFormatException e){
      //Set the error message.
      errorMsg.setText("Bad input data.");
      errorMsg.setBackground(Color.RED);

      //Make the output image black.
      scale[0] = 0;
      offset[0] = 0;
      scale[1] = 0;
```

```
      offset[1] = 0;
      scale[2] = 0;
      offset[2] = 0;
    }//end catch

    //Create the filter object.
    RescaleOp filterObj =
                           new
RescaleOp(scale,offset,null);

    //Apply the filter and return the result.
    return filterObj.filter(theImage,null);

  }//end processColorRescalePage
```

Once again, although this method is rather long and tedious, that is nothing in Listing 8 that should cause you any difficulty if you take a look at the constructor requirement for the **RescaleOp** class in the Sun documentation. Therefore, a further explanation of the code in Listing 8 should not be necessary.

The processImg method

This method must be defined to implement the **ImgIntfc05** interface and to make the class compatible with the driver program named **ImgMod05**. This method is called by the driver program named **ImgMod05**.

Listing 9. The processImg method.

```
  public BufferedImage
processImg(BufferedImage theImage){

    BufferedImage outputImage = null;

    //Process the page in the tabbed pane that
has been
    // selected by the user.
    switch(tabbedPane.getSelectedIndex()){
      case 0:outputImage =

processColorConvertPage(theImage);
             break;
      case 1:outputImage =

processColorRescalePage(theImage);
             break;
    }//end switch

    return outputImage;
  }//end processImg

}//end class ImgMod43
```

This method simply calls the appropriate method to process the page shown in Figure 5 and Figure 6 that has been selected by the user.

Run the program

I encourage you to copy the code from Listing 10 into your text editor, compile it, and execute it. Experiment with it, making changes, and observing the results of your changes.

Keep in mind that you will also need to compile and use the program named **ImgMod05**. You will find the source code for **ImgMod05** in the earlier lesson titled "A Framework for Experimenting with Java 2D Image-Processing Filters" *(see Resources)*.

Summary

This series of lessons began with a promise that I would teach you how to use the following image-filtering classes from the Java 2D API:

- **LookupOp**
- **AffineTransformOp**
- **BandCombineOp**
- **ConvolveOp**
- **RescaleOp**
- **ColorConvertOp**

In previous lessons listed in the Resources section, I taught you how to use the **LookupOp**, **AffineTransformOp**, **BandCombineOp**, and **ConvolveOp** image-filtering classes.

In this lesson, I taught you how to use the **RescaleOp** and **ColorConvertOp** image-filtering classes.

Thus, my promise is fulfilled, and that wraps up this series on the image-filtering classes from the Java 2D API.

Complete program listing

A complete listing of the class discussed in this lesson is provided in Listing 10 below.

Listing 10. Complete listing of the ImgMod43 class.

```
/*File ImgMod43.java
Copyright 2007, R.G.Baldwin

The purpose of this program is to illustrate the use of
the ColorConvertOp and RescaleOp classes of the Java 2D
API.
```

```
See general comments in the class named ImgMod038.

Except for problems encountered when writing the output
jpg file described below, this class is compatible with
the use of the driver program named ImgMod05.

The driver program named ImgMod05 displays the original
and the modified images.  It also writes the modified
image into an output file in JPEG format.  The name of
the output file is junk.jpg and it is written into the
current directory.

Note however that changing the type of the ColorSpace
using ColorConvertOp causes a failure in the attempt to
write the processed image into an output jpg file in
ImgMod05.  In two cases, that code throws an exception
and doesn't write the file.  In the third case, the
output file doesn't contain a valid copy of the
processed image.

Image processing programs such as this one may provide a
GUI for data input making it possible for the user to
modify the behavior of the image processing method each
time the Replot button is clicked.  Such a GUI is provided
for this program.

Enter the following at the command line to run this
program:

java ImgMod05 ImgMod43 ImageFileName

If the program is unable to load the image file within ten
seconds, it will abort with an error message.

This program creates a GUI consisting of a tabbed pane
containing two pages.  The tabs on the pages are labeled:

ColorConvertOp
RescaleOp

Each page contains a set of controls that make it possible
to process the image in a way that illustrates the
processing concept indicated by the labels on the tabs.
Processing details for each page are provided in the
comments in the code used to construct and process the
individual pages.

Tested using J2SE 6 under WinXP.
**********************************************************/

import java.awt.image.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.awt.color.ColorSpace;
```

```
class ImgMod43 extends Frame implements ImgIntfc05{
  //Primary container used to construct the GUI.
  JTabbedPane tabbedPane = new JTabbedPane();

  //Components used to construct the ColorConvertOp page
  // in the JTabbedPane.
  // Components that require local access only are defined
  // locally.  Others are defined here as instance
  // variables.
  Panel colorConvertPage = new Panel();
  CheckboxGroup buttonGroup = new CheckboxGroup();
  Checkbox CS_CIEXYZ =
              new Checkbox("CS_CIEXYZ",buttonGroup,false);
  Checkbox CS_GRAY =
                 new Checkbox("CS_GRAY",buttonGroup,true);
  Checkbox CS_LINEAR_RGB =
          new Checkbox("CS_LINEAR_RGB",buttonGroup,false);

  //Components used to construct the RescaleOp page in
  // the JTabbedPane. Components that require local access
  // only are defined locally.  Others are defined here as
  // instance variables.
  Panel colorRescalePage = new Panel();
  TextField redScaleField = new TextField("1.0");
  TextField redOffsetField = new TextField("0.0");
  TextField greenScaleField = new TextField("1.0");
  TextField greenOffsetField = new TextField("0.0");
  TextField blueScaleField = new TextField("1.0");
  TextField blueOffsetField = new TextField("0.0");

  //The following Label is used to notify of data entry
  // errors.
  String okMessage = "No data entry errors detected.";
  Label errorMsg = new Label(okMessage);
  //-----------------------------------------------------//

  //This is the primary constructor.  It calls other
  // methods to separate the construction of the GUI into
  // easily understandable units.  Each method that it
  // calls constructs one page in the tabbed pane.
  ImgMod43(){//constructor

    //Construct the pages and add them to the tabbed pane.
    constructColorConvertPage();
    tabbedPane.add(colorConvertPage);

    constructColorRescalePage();
    tabbedPane.add(colorRescalePage);

    add(tabbedPane);//Add tabbedPane to the Frame.

    setTitle("Copyright 2007, R.G.Baldwin");
    setBounds(555,0,470,300);
    setVisible(true);
```

```
    //Define a WindowListener to terminate the program.
    addWindowListener(
      new WindowAdapter(){
        public void windowClosing(WindowEvent e){
          System.exit(1);
        }//end windowClosing
      }//end windowAdapter
    );//end addWindowListener
  }//end constructor
  //-----------------------------------------------------//

  //This method constructs the page in the tabbed pane
  // that shows ColorConvertOp on the tab.  This method is
  // called from the primary constructor.
  void constructColorConvertPage(){
    colorConvertPage.setName("ColorConvertOp");//Tab label
    colorConvertPage.setLayout(new BorderLayout());

    //Create and add the instructional text to the page.
    // This text appears in a disabled text area at the
    // top of the page in the tabbed pane.
    String text ="COLOR TYPE CONVERSION\n"
      + "This page illustrates the use of the "
      + "ColorConvertOp filter class.\n\n"
      + "Select the appropriate radio button below, and "
      + "click the Replot button to cause the program to "
      + "convert the ColorSpace to the type indicated by "
      + "the radio button.\n\n"
      + "Note that changing the type of the ColorSpace "
      + "causes the later attempt to write the processed "
      + "image into an output jpg file in ImgMod05 to "
      + "fail.  In two cases, that code throws an "
      + "exception.  In the third case, the output file "
      + "doesn't contain a valid copy of the processed "
      + "image.";

    //Note:  The number of columns specified for the
    // following TextArea is immaterial because the
    // TextArea object is placed in the NORTH location of
    // a BorderLayout.
    TextArea textArea = new TextArea(text,10,1,
                                TextArea.SCROLLBARS_NONE);
    colorConvertPage.add(textArea,BorderLayout.NORTH);
    textArea.setEnabled(false);

    //Construct the control panel and add it to the page.
    Panel controlPanel = new Panel();
    controlPanel.add(CS_CIEXYZ);
    controlPanel.add(CS_GRAY);
    controlPanel.add(CS_LINEAR_RGB);
    colorConvertPage.add(
                         controlPanel,BorderLayout.CENTER);
  }//end constructColorConvertPage
  //-----------------------------------------------------//

  //This method processes the image according to the
```

```
  // controls located on the page in the tabbed pane that
  // shows ColorConvertOp on the tab.
  //This method uses the ColorConvertOp filter class to
  // process the image and to convert it to the
  // ColorSpace type indicated by a radio button selected
  // by the user.  This method illustrates only three of
  // the many ColorSpace types defined in the ColorSpace
  // class.
  //This method is called from within the switch statement
  // in the method named processImg, which is the primary
  // image processing method in this program.
  BufferedImage processColorConvertPage(
                                BufferedImage theImage){
    //Examine the radio buttons.  Cause the ColorSpace of
    // the image to be converted to the type indicated
    // by the radio button.
    int colorSpaceType;
    if(CS_CIEXYZ.getState() == true){
      colorSpaceType = ColorSpace.CS_CIEXYZ;
    }else if(CS_GRAY.getState() == true){
      colorSpaceType = ColorSpace.CS_GRAY;
    }else{//CS_LINEAR_RGB must be selected
      colorSpaceType = ColorSpace.CS_LINEAR_RGB;
    }//end else

    //Create the filter object.
    ColorConvertOp filterObj = new ColorConvertOp(
             ColorSpace.getInstance(colorSpaceType),null);

    //Apply the filter and return the result.
    return filterObj.filter(theImage,null);

  }//end processColorConvertPage
  //-------------------------------------------------------//

  //This method constructs the page in the tabbed pane
  // that shows RescaleOp on the tab.  This method is
  // called from the primary constructor.
  void constructColorRescalePage(){
    colorRescalePage.setName("RescaleOp");//Tab label.
    colorRescalePage.setLayout(new BorderLayout());

    //Create and add the instructional text to the page.
    String text = "COLOR RESCALING\n"
      + "This page illustrates the use of the RescaleOp "
      + "filter\n\n"
      + "Enter new scale and offset values for each "
      + "color in the text fields below and then click "
      + "the Replot button.\n\n"
      + "This will cause the color values for each color "
      + "to be multiplied by the scale factor specified "
      + "for that color.  Then the offset value for that "
      + "color will be added to the scaled color values.";

    //Note:  The number of columns specified for the
    // following TextArea is immaterial because the
```

```
    // TextArea object is placed in the NORTH location of
    // a BorderLayout.
    TextArea textArea = new TextArea(text,9,1,
                                TextArea.SCROLLBARS_NONE);
    colorRescalePage.add(textArea,BorderLayout.NORTH);
    textArea.setEnabled(false);

    //Construct the control panel and add it to the page.
    Panel controlPanel = new Panel();
    controlPanel.setLayout(new GridLayout(3,5));
    controlPanel.add(new Label("Red"));
    controlPanel.add(new Label("Scale = "));
    controlPanel.add(redScaleField);
    controlPanel.add(new Label("Offset = "));
    controlPanel.add(redOffsetField);

    controlPanel.add(new Label("Green"));
    controlPanel.add(new Label("Scale = "));
    controlPanel.add(greenScaleField);
    controlPanel.add(new Label("Offset = "));
    controlPanel.add(greenOffsetField);

    controlPanel.add(new Label("Blue"));
    controlPanel.add(new Label("Scale = "));
    controlPanel.add(blueScaleField);
    controlPanel.add(new Label("Offset = "));
    controlPanel.add(blueOffsetField);

    colorRescalePage.add(
                        controlPanel,BorderLayout.CENTER);

    //Add the errorMsg label.
    colorRescalePage.add(errorMsg,BorderLayout.SOUTH);
    errorMsg.setBackground(Color.GREEN);

  }//end constructColorRescalePage
  //-----------------------------------------------------//

  //This method processes the image according to the
  // controls located on the page in the tabbed pane that
  // shows RescaleOp on the tab.  This method is called
  // from within the switch statement in the method named
  // processImg.  This method uses the scale and offset
  // values specified by the user to modify the red,
  // green, and blue color values.  The new color value
  // for each pixel is the old value multiplied by the
  // scale factor for that color plus the offset value for
  // that color.
  BufferedImage processColorRescalePage(
                                  BufferedImage theImage){

    //Reset the error message to the default.
    errorMsg.setText(okMessage);
    errorMsg.setBackground(Color.GREEN);

    //Create the arrays required to contain the scale and
```

```
    // offset values.
    float[] scale = new float[3];
    float[] offset = new float[3];

    //Populate the scale and offset arrays using data from
    // the text fields.
    try{//Get input value from the text field.
      scale[0] =
                Float.parseFloat(redScaleField.getText());
      offset[0] =
               Float.parseFloat(redOffsetField.getText());
      scale[1] =
              Float.parseFloat(greenScaleField.getText());
      offset[1] =
             Float.parseFloat(greenOffsetField.getText());
      scale[2] =
               Float.parseFloat(blueScaleField.getText());
      offset[2] =
              Float.parseFloat(blueOffsetField.getText());
    }catch(java.lang.NumberFormatException e){
      //Set the error message.
      errorMsg.setText("Bad input data.");
      errorMsg.setBackground(Color.RED);

      //Make the output image black.
      scale[0] = 0;
      offset[0] = 0;
      scale[1] = 0;
      offset[1] = 0;
      scale[2] = 0;
      offset[2] = 0;
    }//end catch

    //Create the filter object.
    RescaleOp filterObj =
                         new RescaleOp(scale,offset,null);

    //Apply the filter and return the result.
    return filterObj.filter(theImage,null);

  }//end processColorRescalePage
  //-----------------------------------------------------//

  //The following method must be defined to implement the
  // ImgIntfc05 interface.  It is called by the driver
  // program named ImgMod05.
  public BufferedImage processImg(BufferedImage theImage){

    BufferedImage outputImage = null;

    //Process the page in the tabbed pane that has been
    // selected by the user.
    switch(tabbedPane.getSelectedIndex()){
      case 0:outputImage =
                        processColorConvertPage(theImage);
             break;
```

```
      case 1:outputImage =
                          processColorRescalePage(theImage);
               break;
    }//end switch

    return outputImage;
  }//end processImg

}//end class ImgMod43
```

Copyright

Resources

- 400 Processing Image Pixels using Java, Getting Started
- 402 Processing Image Pixels using Java, Creating a Spotlight
- 404 Processing Image Pixels Using Java: Controlling Contrast and Brightness
- 406 Processing Image Pixels, Color Intensity, Color Filtering, and Color Inversion
- 408 Processing Image Pixels, Performing Convolution on Images
- 410 Processing Image Pixels, Understanding Image Convolution in Java
- 412 Processing Image Pixels, Applying Image Convolution in Java, Part 1
- 414 Processing Image Pixels, Applying Image Convolution in Java, Part 2
- 416 Processing Image Pixels, An Improved Image-Processing Framework in Java
- 450 A Framework for Experimenting with Java 2D Image-Processing Filters
- 452 Using the Java 2D LookupOp Filter Class to Process Images
- 454 Using the Java 2D AffineTransformOp Filter Class to Process Images
- 456 Using the Java 2D LookupOp Filter Class to Scramble and Unscramble Images
- 458 Using the Java 2D BandCombineOp Filter Class to Process Images
- 460 Using the Java 2D ConvolveOp Filter Class to Process Images

About the author

Richard Baldwin *is a college professor (at Austin Community College in Austin, TX) and private consultant whose primary focus is a combination of Java, C#, and XML. In addition to the many platform and/or language independent benefits of Java and C# applications, he believes that a combination of Java, C#, and XML will become the primary driving force in the delivery of structured information on the Web.*

Richard has participated in numerous consulting projects and he frequently provides onsite training at the high-tech companies located in and around Austin, Texas. He is the author of Baldwin's Programming Tutorials, which have gained a worldwide following among experienced and aspiring programmers. He has also published articles in JavaPro magazine.

In addition to his programming expertise, Richard has many years of practical experience in Digital Signal Processing (DSP). His first job after he earned his Bachelor's degree was doing DSP in the Seismic Research Department of Texas Instruments. (TI is still a world leader in DSP.) In the following years, he applied his programming and DSP expertise to other interesting areas including sonar and underwater acoustics.

Richard holds an MSEE degree from Southern Methodist University and has many years of experience in the application of computer technology to real-world problems.

Baldwin@DickBaldwin.com

Keywords
java 2D image pixel framework filter ColorConvertOp RescaleOp

-end-