CSCE 145: Lab 12

Image Processing

Objective: In doing this lab you will learn about nested loops and the processing of images.

Specification: Do problem #19 on page 192 of our text, which is as follows: Add a *subtract* method to the **Compositor** class on p. 150 and shown below. This method takes two images of type **BufferedImage** as arguments and returns a new image, which is also of type **BufferedImage**. The pixels in the returned image are the *difference* of the pixels in the input images. Suppose that *(r1, g1, b1)* and *(r2, g2, b2)* are the red, green, and blue components of a pixel with coordinates *(x, y)* in the imput images. Then a pixel in the output image at the same location has component values *(r1-r2, g1-g2, b1-b2)*. Any negative component values should be set to 0. Write a class with a main method that tests this method for any two images of your choice. Is the result of subtracting the first image from the second the same as the result of subtracting the second from the first? Test this as well.

The **Compositor** class:

```
import java.awt.*;
import java.awt.image.BufferedImage;
public class Compositor {
  static final int MAX_VALUE = 255;
  // this method returns a new image created by adding
  // the source image image2 from the source image image1
  public BufferedImage add(BufferedImage image1, BufferedImage image2) {
    int width = Math.min(image1.getWidth(),  image2.getWidth());
    int height = Math.min(image1.getHeight(),  image2.getHeight());

    // create a new BufferedImage called image3
    BufferedImage image3 = new BufferedImage(width,  height, BufferedImage.TYPE_INT_ARGB);

    for (int x = 0; x < width; x++) {
      for (int y = 0; y < height; y++) {
        // get the samples of the pixel at (x, y) in image1
        int colorValue1 = image1.getRGB(x,  y);
        Color pixelColor1 = new Color(colorValue1);
        int red1 =  pixelColor1.getRed()  ;
        int green1 =  pixelColor1.getGreen()  ;
        int blue1 =  pixelColor1.getBlue();

        // get the samples of the pixel at (x, y) in image2
        int colorValue2 = image2.getRGB(x,  y);
        Color pixelColor2 = new Color(colorValue2);
        int red2 =  pixelColor2.getRed()  ;
        int green2 =  pixelColor2.getGreen()  ;
        int blue2 =  pixelColor2.getBlue();

        // find the sum of the samples to create a new color
        int red3 =  Math.min(red2 + red1, MAX_VALUE);
        int green3 =  Math.min(green2 + green1, MAX_VALUE);
        int blue3 =  Math.min(blue2 + blue1, MAX_VALUE);
```

```
        // set the color of a pixel in image3
        Color newPixelColor = new Color(red3, green3, blue3);
        int newRgbvalue = newPixelColor.getRGB();
        image3.setRGB(x, y, newRgbvalue);
      }
    }
   // returns a reference to the new image
   return image3;
  }

// main method to test the add method
  public static void main(String[] args) {
    DrawingKit dk = new DrawingKit("Compositor", 1000, 1000);
    BufferedImage p1 = dk.loadPicture("image1.jpg");
    BufferedImage p2 = dk.loadPicture("image2.jpg");
    Compositor c = new Compositor();
    BufferedImage p3 = c.add(p1,p2);
    dk.drawPicture(p3, 0, 100);
  }
```

Upload your program to the CSE Dropbox, located at https://dropbox.cse.sc.edu.