Homework 3

CS 1331

Due Friday September 27th, 2013 11:59 PM

1 Introduction

Hiya everyone! Now that you should have charted how to survive haunted hotels with the Evil Hotel Sim 2013, you should have learned how implement methods inside classes. Now to follow up on that, we will be implementing more methods but this time specifically with 2D arrays. Behold your next assignment: the Picture Befuddler!

2 Problem Description

So here's the scoop. You're a museum thief, but not any museum thief! You are a museum thief with a sense of humor and thought that to confuse the police investigators, you would leave multiple fake copies of the art that you've stolen behind in your wake. But to make things even more interesting (or perhaps frustrating for your chasers), you've modified all of the fake copies to be silly alterations based on the original picture, some of which are barely recognizable. Here are the following ways you can choose to alter copies of your stolen picture:

1. **Greyscale**: Requires that you take the **average** of the RGB (red, green, and blue!) color channels together and then set each of the channels to that averaged value.
2. **Invert**: Subtract each of the RGB channel values from the maximum possible value allowed (255) and then set each channels to its respective **difference**.
3. **Watermark/Overlay**: This one's a little tricky. This alteration takes in another picture name and combines the two pictures into one. You should find where on the larger picture you want to watermark the smaller image and then average the RGB color values of the pixels of the two corresponding images and set the larger picture's equivalent pixel to the averaged amount. Think carefully about which pixels you are going to loop through, as in this method you will need the pixel values of not one but two different images!
4. **onlyRed**: Make it so that the **red** channel is the **only** channel that affects the image.
5. **onlyBlue**: Same as onlyRed only with **blue**.
6. **onlyGreen**: Once again, same only with **green**.

7. **Maximize/Posterize**: Here you will find which of the RGB color channel values is the **highest**. Based on that, you set the other two channel values to **0** and keep the highest value the same.

With that being said, create a class called **ImageProcessor** that implements all of these methods. Some things to note about ImageProcessor:

- Use the given **Pixel** and **Pic** classes to implement ImageProcessor, thus ImageProcessor is manipulating a picture of type Pic which has Pixels.
- All of these methods will return a **copy** of the input picture, they **do not** modify the input picture to return that same picture modified
- All of these methods require that you handle **2D arrays** to modify every pixel in the picture
- Your ImageProcessor class is the blueprint for an ImageProcessor object, be sure that the ImageProcessor can be created properly using a **constructor** that allows the ImageProcessor to know which picture it is manipulating
- Be sure that your ImageProcessor can remember and know which image it is manipulating, each ImageProcessor knows how to manipulate only one picture that it has been constructed with
- The **main method** should instantiate an ImageProcessor object **with an image name provided by the command-line arguments**. For Watermark, the second image will be the next command-line argument. Then, run each of the picture manipulation methods. Think of it having a similar purpose as the Driver did in the previous homework.
- **Show** all resulting pictures at the end of every method.

2.1 Useful Tips

- Think about how you might iterate through a 2D array to find all of the pixels. It takes one loop to find all the elements within a 1D array. How many might it take to do the same for a 2D array?
- How might you get the color channels of every pixel? Remember getters from the previous homework? Make sure you read the methods in both Pic and Pixel, you will need to use the methods in these classes to implement ImageProcessor.
- Since you are making copies of the picture you're manipulating, you will have to make sure you can return that copy. Be sure to read the Pic class to see if there are any way you can copy Pic object.
- Don't forget to show the resulting image at the end of the method. Once again, I emphasize reading the Pic and Pixel classes given!
- You are **not** to modify Pic.java or Pixel.java. You are given every method required to do this assignment already. Your submissions will be run with the unmodified Pic.java and Pixel.java, so if your program is relying on changes you've made then your grade will suffer.

2.2 Javadocs

We will be looking this homework for javadocs. Javadocs are a special kind of comment that makes learning someone else's code simple and easy. To write a javadoc, put this above the headers of your methods and key classes.

```
/**
 * toString creates a String representation of this object
 *
 * @return a string representation of the object
 *
 *
 */
public String toString() { ... et cetera
```

There are a number of tags (prepended with @) that you are required to use in your javadocs. They always go at the end of the javadoc, after the description of the method/class. Here are the ones you'll be required to use:

• @author Aaron Friesen	🡐 Use this javadoc on class javadocs.
• @version 1.0	🡐 Use this javadoc on class javadocs.
• @param x	🡐 Use this javadoc on method javadocs.
• @return y	🡐 Use this javadoc on method javadocs.

For examples of each of these, take a look at the javadocs inside the Pic and Pixel class.

2.3 Program Requirements

In summary, your program should…

1. Create a Pic object containing all the pixel data from an image for the ImageProcessor to manipulate
2. Create an ImageProcessor using a constructor that allows the ImageProcessor to know what picture it is manipulating, **obtaining the picture name(s) from the command-line arguments.**
3. Run the greyscale, invert, watermark, onlyRed, onlyBlue, onlyGreen, and maximize methods
4. Display the result of every method on the screen
5. Include method and class javadocs for ImageProcessor.java.

Make sure the first line is /** only and that all following lines are indented by a single space (so the * aligns with the first * on the first line. The last line of a javadoc should only contain */.

2.4 Turn-in Procedure

Turn in the following files to T-Square. When you're ready, double-check that you have submitted and not just saved a draft.

- ImageProcessor.java

Make sure you are submitting .java files and not any .class files.

3 Verify the Success of Your HW Turn-In

Practice safe submission! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.

2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.

3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.

4. Recompile and test those exact files.

5. This helps guard against a few things.

   a. It helps insure that you turn in the correct files.

   b. It helps you realize if you omit a file or files.**

      (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)

   c. Helps find last minute causes of files not compiling and/or running.

**Note: Missing files will not be given any credit, and non-compiling homework solutions will receive few to zero points. Also recall that late homework will not be accepted regardless of excuse. Treat the due date with respect. The real due date is midnight Friday. Do not wait until the last minute!

Happy Hacking!