PSA 3 : Modifying Pictures

Read all instructions in this document before starting any coding!

Due: Sunday, October 29th, 11:59pm

In this assignment you will use **loops** to modify the colors of images (Picture objects).
Overview Video about this PSA

As described in class, **you may complete at most 4 of the quarter's programming assignment using pair programming.** If you have decided to work in pairs, please review the guidelines for pair programming. In addition, note the following details about working with a partner:

- For this assignment, if you decided to work in pairs, find a partner before you start on the assignment. Or else, you have violated the policy on pair programming.
- You will submit only ONE version between the two of you though your group can submit as many times as you want to.

Helpful Information:

Online Communication: Using Piazza, Opening Regrade Requests

Getting Help from Tutor, TA, and Professor's Hours

Lab and Office Hours (Always refer to this calendar before posting.)

Academic Integrity: What You Can and Can't Do in CSE 8A

Remote Lab Access and File Transfer Guide or **Yingjun's SSH and VNC post**

Picture Documentation and Pixel Documentation

Table of Contents:

Getting Started

Instructions for working on a B230 lab machine:

1) Open a terminal
2) Copy the starter code from the public folder.

Paste the following lines of code one after the other in the terminal i.e. paste the first line, press enter and then the second line and so on:

```
cd ~/
mkdir psa3
cd psa3
cp ../../public/psa3/* .
ls
```

(Do you recall what each of the above commands mean. If you are not able to, review PSA0, where each of the commands is explained.)

3) Verify that Picture.java, TestPictures.java, TriEffect.java, bear.jpg, 8Arocks.png, blending.png, and yellowFlowers.jpg is shown as the result of the last command.
Following these procedures, your file will be named correctly and be located in the correct place. For every assignment you need to follow correct naming assignments in the correct locations to get credit for your work.

4) Proceed to start your programming assignment.

Instructions for working on your Local Machine (Your Laptop/Computer):

1) Create a psa3 folder on your machine.
2) Download zip and unzip it inside the psa3 folder: LINK HERE

Problems:

Part A: Color Scaling (4 points)

Write a program that scales the red, blue, and green components of all pixels of a Picture by specified values. To do this, you will write a method called `scaleColor` in **Picture.java** and then in the main method of **TestPictures.java** you will test your `scaleColor` method by showing and exploring the original and changed pictures.

Method: `scaleColor` (2 pts for producing the proper effect, 1 pt for proper parameters)

- Add a method to Picture.java called scaleColor. Your method takes three double parameters which specify how much the color components of ALL PIXELS of the picture should be scaled by **(red, green and blue, in that order)**. For example, if you pass (1.12, 1.2, 0.5) as parameters to your method and the color values of one of the pixels of the picture is (150, 200, 50), then the red, green and blue color values should be scaled to (168, 240, 25), respectively, after your method is called. The order in which the color values are passed to the method should be red, green, and blue, in that order.
- The method should scale the color values of **ALL** the pixels in the picture.
- Keep in mind that pixel value should be integers, so you may need to typecast values back to int.
- Pixel values cannot go below 0, or above 255, so you should write some logic in order to guard against this. Here are some ideas on how to use if statements (we will cover it later this quarter).

```
if(variable > 255) {
     //change var = 255 (the max value)
}
// do this for values below 0 as well
```

- Test your method in TestPictures.java

```
/*
 * Scale the given values from the appropriate colors.
 * Input: rScale - the amount of red value that should be scaled
 *        gScale - the amount of blue value that should be scaled
 *        bScale - the amount of green value that should be scaled
 * Returns: nothing
 */
public void scaleColor(double rScale, double gScale, double bScale)
{
    //fill in
}
```

Method: `TestPictures.java` - main (1 pt for proper execution with `scaleColor`)

Fill in the statements (inside the main method) to test and show the results of your `scaleColor` method:

Method "scaleColor"

```
public \boxed{void} scaleColor(double rScale, double gScale,
                               double bScale)
  {
```

Method: `TestPictures.java` - main (1 pt for proper execution with `scaleColor`)

Fill in the statements (inside the main method) to test and show the results of your `scaleColor` method:

1. Use the `FileChooser.pickAFile()` statement to pick a picture (you can pick a picture from the mediasources directory -- or better yet -- use one of your own!).
2. The mediasources directory can be found in the following path: `home > linux > ieng6 > cs8af > public > mediasources`
    a. Once you find the folder, **you should copy a .jpg image file of your choice to your psa3 folder and use it for this assignment**.
    b. It is recommended that you use an image that is in color and is relatively small. This will help you speed up the testing.
    c. Alternatively, you can navigate to the mediasources directory and list the files in the directory using command line:
    ```
    cd ~/../public/mediasources
    ls
    ```
    d. You are also given an example image in the starter files called yellowFlowers.jpg

3. BEFORE you modify this picture, we want you to make a **COPY** of the picture object so we can "see" the difference after the picture is changed (use the same technique shown in lab). Make **a new Picture object** which is a copy of the one you just made. Use this code statement to do so:

```
Picture copy = new Picture(original);
```

4. Next step is to **explore** (you can *explore* a picture by calling the predefined `explore()` method on that picture**)** the original picture you picked.

5. Call your `scaleColor` method to change the COPY of picture you made.

6. *Explore* modified picture object after completing Step 5.

7. Mouse around to explore the original and changed picture to make sure your method works correctly (that is, a specific pixel in the original picture has been scaled by the right amounts in its red, green, and blue components for the copied picture).

- Say you passed in (1.12, 1.2, 0.5) as your parameters. Examine the values of the two pictures at, for example (100, 100), for both the pictures. If the value you examine in the unmodified picture is (150, 200, 50) and (112, 240, 25) for the modified picture, then you know something is wrong with your code. Hmm.. only the red value seems to be incorrect. Perhaps you can go back to the code and see if you accidentally set the wrong value to your red component.

Part B - Tri-Effect (6 points)

Method: `complement` (0.5pt for proper parameters, 1pt for producing the proper effect)

- Add a method (`complement`) to Picture.java that takes in **two int parameters**, which are the first and last index of the region (part) of the pixel array to be modified (inclusive).
- The method should access the pixel array of the calling picture object (use `this.getPixels()` line to store the picture's pixels in an array) and using a while loop, invert the color attribute value of each pixel between (and including) the provided indices. Hint: it may be helpful (and easier to debug!) to create a helper method which will return the correct index given an x and y coordinate.
- For example, if a picture's RGB is 100,120, and 200, the inverted RGB of that pixel would be 155 (255-100), 135 (255-120) and 55 (255-200)
- The method header for this is:

```
/**
* Create the complement of each pixel between the provided indices
* Input: start - the index of the first pixel to be modified
* (inclusive)
* end - the index of the last pixel to be modified (inclusive)
* Returns: nothing
*/
public void complement(int start, int end)
{
     //fill in
}
```

Method: `grayscale` (0.5pt for proper parameters, 1pt for producing the proper effect)

- Add another method to Picture.java named `grayscale`**(You will find out in lab on how to convert an image to grayscale or check the textbook for an example)** similar to the complement method which also has the first and last index in the pixel array as parameters. For those pixels in range, you need to change the colors to their appropriate grayscale color using a while loop.
- The method header for this is:

```
/*
* Create the gray equivalent of each pixel between the provided indices
* Input: start - the index of the first pixel to be modified (inclusive)
*        end - the index of the last pixel to be modified (inclusive)
* Returns: nothing
*/
public void grayscale(int start, int end)
{
     //fill in
}
```

Method: `myFilter` (0.5pt for proper parameters, 1pts for producing the proper effect)

- Add yet another method(`myFilter`) to Picture.java, similar to the the complement and grayscale methods, which also takes at least two parameters: the first and last index(both inclusive) in the pixel array to be modified. Apply any effect of your choice to the pixels in range using a while loop.

- This filter (for example : increasing the brightness of a picture) cannot be grayscale, complement , a single color filter, the example mentioned in this sentence or any filter mentioned in the book!!! Single Color Filter is where you change all the pixels of picture to a single color. For example : Setting red component, green component and blue component to 50 for all pixels in the picture.
- The method header for this is:

```
     /**
     * Apply myFilter to each pixel between and including the provided
indices
     * ADD A SHORT DESCRIPTION OF YOUR FILTER HERE
     * Input: start - the index of the first pixel to be modified (inclusive)
     *        end - the index of the last pixel to be modified (inclusive)
     * Returns: nothing
     */
     public void myFilter(int start, int end)
     {
          //fill in
     }
```

- **This filter should not be similar to the examples in the book. If you doubt whether your filter is similar to the example, they are similar in that case!**

Method: `TriEffect.java` - main (1.5 points for the proper effect on each third of the image)

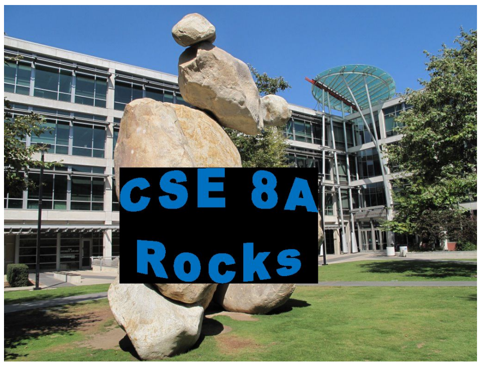
- This file will load a Picture of your choice (either a bookClasses image or a personal image) using `FileChooser.pickAFile()`.
- Display the original image.
- Call the `complement` method on the **first 1/3 of pixels i**n the image. Next call the `grayscale` method on the **middle 1/3 of pixels** in the image. Then apply the `myFilter` method to the **bottom 1/3 of the pixels** in the image**.**
- Lastly, display your new image.

Part C - Alpha Blending (6 points)

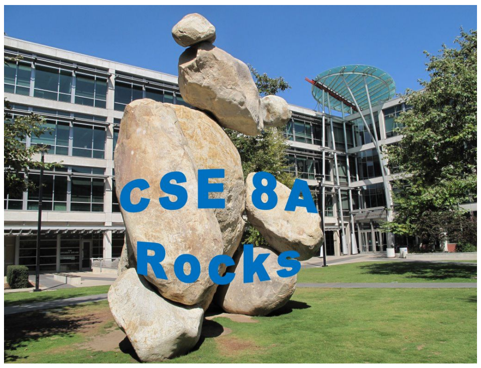
Alpha blending is an effect to “blend” one picture (source) onto another picture (target) by changing color of target pixels with the help of the source pixels’ alpha channels. This is especially true if the source picture has pixels that have some transparency variations. If we don’t consider the transparency of pixels in a picture, we might not achieve the best result. For example, the following picture shows a picture with transparent filling areas.

CSE 8A
Rocks

We have an engineering bear picture to serve as the background. The following two pictures show the different effects of simple direct copy (left) and alpha blending (right).



Simple Copy



Alpha Blending

We already know that each pixel has its red, green, and blue components. Additionally, each pixel also has a component called the Alpha channel whose range is between 0 and 255. A 0 Alpha channel means that pixel is totally transparent and 255 means the pixel is opaque. We will write a method to achieve this effect.

Method: `alphaBlending` (1pt for proper parameters, 3pts for producing the proper effect)

- Add a method (`alphaBlending`) to Picture.java that takes in **three parameters**. The first two parameters specify the (`x`,`y`) location of region that the calling object will be blended into the parameter Picture object. The last parameter is a Picture reference that serves as the background to blend the calling object into. You can assume that the size of the calling object fits in the parameter object at the specified location.
- Alpha blending works by blending two corresponding pixels (one from the parameter Picture object and one from the calling object) to create the effect that one picture is blended into the other picture. The way to create the blending effect is the following:

  Suppose that the calling Object’s pixel is pix_source and the corresponding pixel from the parameter picture is pix_target. You can use **getAlpha()** method to obtain the Alpha value from a pixel (range from 0 to 255). Alpha is the opacity of a pixel.

Red of pixel_target = alpha_source / 255 * Red_source + (1 - alpha_source / 255) * Red_target

Similarly, you can apply the formula to Green and Blue channel.

Remember to make alpha_source a double or cast it to a double to avoid loss of data in the calculation. Then cast the calculation back to an int to change the color of a Pixel.

- The method header for this is:

```
/**
   * Blend the calling object pic into the parameter Picture
   * object with the upper left corner at (x, y)
   * Input: x - The background's top left corner x coordinate
   *        y - The background's top left corner y coordinate
   *        background - The background picture to blend into
   * Returns: nothing
   */
  public void alphaBlending(int x, int y, Picture background)
  {
     //fill in

  }
```

Test alphaBlending:

Method: `testAlphaBlending` (1pt for proper parameters, 1pt for producing the proper effect)

You will write a method called `testAlphaBlending` that will check if your `alphaBlending` method works. It will work as follows:

1. Loop through both the original picture and the result picture in the same loop.
2. Compare the RGB values of each index of the two pictures
3. If at a certain index, you find that the pixels of the two pictures are not the same, you will return false
4. If you get to the end of the loop and not find such pair, you will return true

Your method will return a `boolean` type, which is a `true` or `false` value.

Note: To check the inequality between to ints, you can use the `!=` operator. For example, (`3 != 4`) will return `true` and (`3 != 3`) will return `false`

```
/**
* Test the alphaBlending method by comparing two pictures
* Input: result - The picture to compare to
* Returns: true if the pictures match
*          false if they do not
*/
public boolean testAlphaBlending(Picture result)
{
     //loop through all pixels in the calling object and parameter
     //picture using nested loops on x and y
          Pixel sP = this.getPixel(x, y);
          Pixel tP = result.getPixel(x, y);
          if( sP.getRed() != tP.getRed())
               return false;
          //Similarly for blue and green.
```

```
            //outside the nested loop, we return true as if we reach here
            //it is guaranteed that all pixels between the calling obj and
            //the parameter picture are the same.

            return true;
        }
```

In the main method in TestPictures.java, create three Picture objects: name the Picture object for 8Arocks as `source`, the Picture object of bear.jpg as `target`, and the Picture object for blending.png as `correct`. Then call the `alphaBlending` method from the `source` object and pass the (`x`,`y`) parameters as (`250`,`350`) and `target` as the third parameter. Then call `testAlphaBlending` on your `target` picture and choose `correct` as the parameter. This last method call can be written as part of the print statement to print out the return value of the method call.

```
System.out.println("Compare with correct answer. Result = " +
target.testAlphaBlending(correct));
```

If you see the result is true, then your code works correctly. If it is false, then there is something incorrect in your code. You need to go back and debug.

Program Description (2 points)

Describe what your program does as if it was intended for a 5 year old or your grandmother. Do not assume your reader is a computer science major. The programs you should comment on in this segment include **TestPictures.java, TriEffect.java and Picture.java. Write this as comments at top of Picture.java file.**

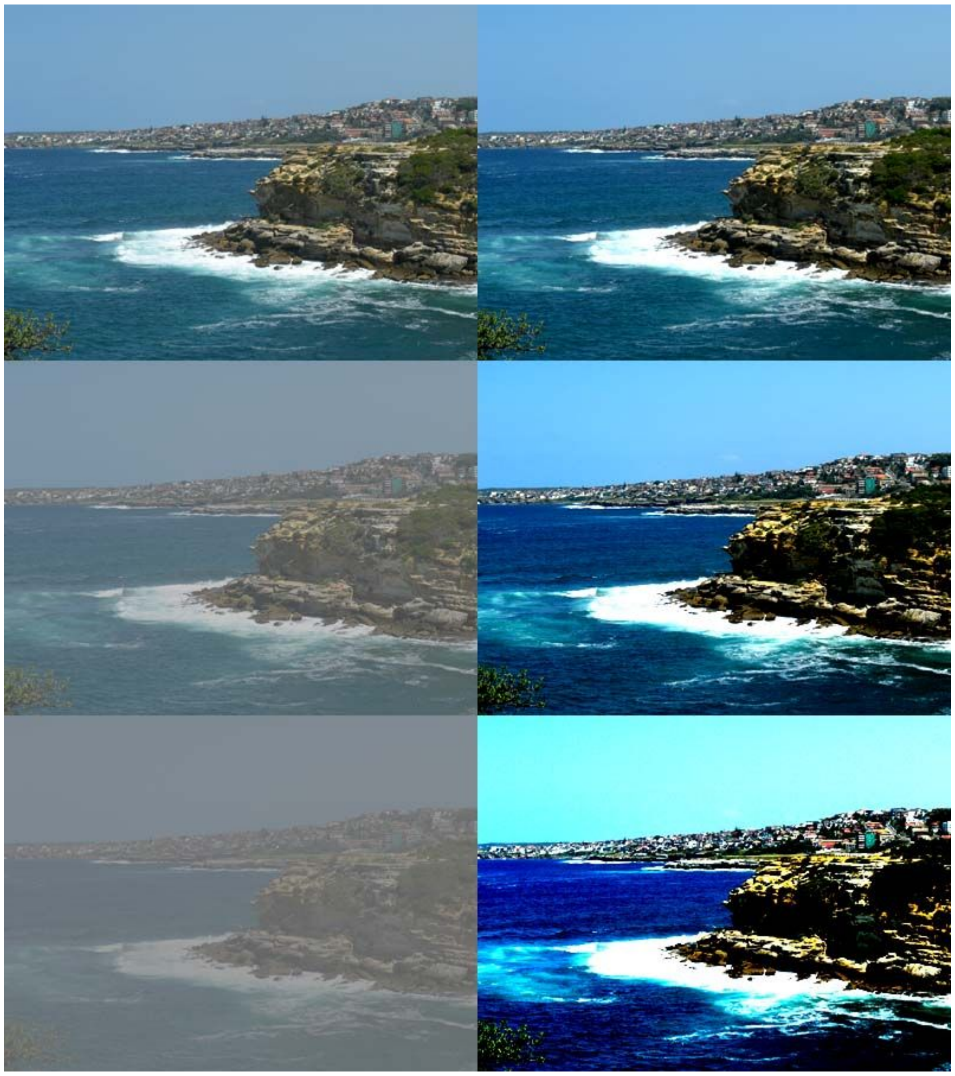
Commenting and Style (2 points)

This week we will grade you on your code style and comments for your code. We will cover this in class, but to get a head start on what we are looking for while you do your PSA, you should look at the "before" slides from week 3, which talks about style . We will be looking for:

1. Method Header Comments**:** A comment at the start of each method you write that states what the method does and what it takes as input and returns (if anything)
2. Good variable names: names that mean what the variable is for--e.g. "bob" and "s" are not a good variable names for a pixel. Try something like "currPixel"
3. Proper indentation: use Dr. Java's "indent region" command!
4. Inline comments: Intermittent comments that give a very brief explanation of your logic, usually right above or after particularly complex or possibly ambiguous snippets of code. This usually also helps programmers keep track of their thoughts and others understand the code.

Star Point

The Star point option this assignment will be to write a filter that will enhance the global contrast of an image. If you are unclear about what global contrast enhancement entails, please do a little research to clarify what this entails. An example of a set of contrast enhanced images are as follows:



The amount in which you want to increase the contrast by is up to you. When grading, we will check to see if there is a visible increase in contrast after using your filter. If we do not see any obvious changes in such a way, no star point will be given. So be sure that your contrast is truly enhanced.

Warning: This is NOT an easy Star point assignment and you should focus on the main part of the PSA before attempting this. Remember that you may complete the Star point either with your partner or on your own, but you must place a header comment at the top of your file to make it very clear whose work it is. For this Star point, you should create a NEW filter method in Picture.java that you can call in a new file called **StarPointPSA3.java**. Please name your new star point method **starPointFilter** and add a comment in both **Picture.java** and **StarPointPSA3.java** headers. **Don't forget your header comment in**

that file which includes all the usual information. If both the partners in a group are doing Star point then, create the methods as starPointFilter1(Partner 1) and starPointFilter2(Partner 2) in Picture.java and create 2 files StarPointPSA3Version1.java(Partner 1) and StarPointPSA3Version2.java(Partner 2) to call them.

How to Turn in Your Homework

1. Place your psa3 folder in the home directory of your student account (cs8afxx).
2. Ensure that the psa3 folder contains at least all the following files: **Picture.java, TestPictures.java, and TriEffect.java. If you did the Star point you will also need to turn in StarPointPSA3Version1.java (and StarPointPSA3Version2.java)**
3. Run the command: cse8aturnin psa3
4. Follow the prompts.
5. Verify that your homework was turned in with the command: cse8averify psa3

Refer back to psa0 turnin instructions for additional details.