

CSE 8A Programming Assignment 7

Fall 2019

Due Date: November 25, 2019

Learning Goals:

- Working with objects (`Image` and `Color`)
- Getting comfortable with instance methods and static methods
- Write code to transform images

Submission:

There are two deliverables for this assignment which you will submit separately on Gradescope. See instructions on how to submit at the end of this document.

- Deliverable 1: You will submit the `image.java` file to be autograded
 - Deliverable 2: You will submit a PDF write-up using the template [here](#)
-

Overview

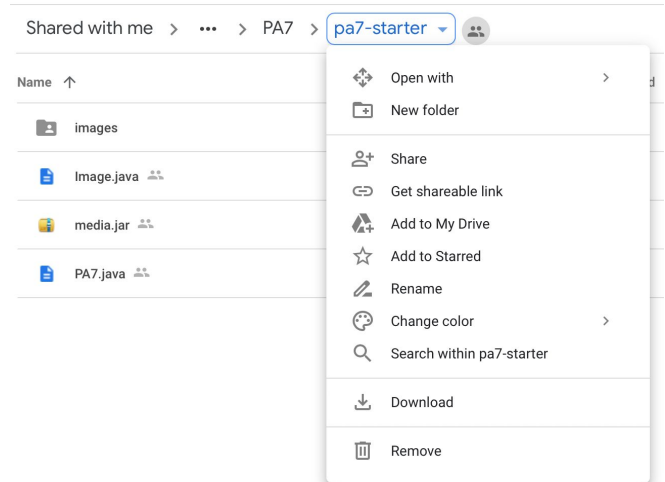
In this programming assignment, we will continue our focus on working with 2D arrays that represent images. You will perform some interesting transformations that will allow us to make a fun collage with your favorite pictures. While having the chance to get creative, you will also be learning how to work with objects and instance methods, as well as getting more practice iterating through 2D arrays.

Instance Methods vs. Static Methods

Up until this assignment, you have been creating and writing methods that have the keyword `static` in their signature. By adding this keyword, you were making methods that could be called without creating an object. A static method called `myMethod` in a class called `MyClass` could be called by `MyClass.myMethod()`, or from within the class just by calling the method from another static method. On the other hand, instance methods do not contain the keyword `static`, and they require an object to be created in order to be called. You can think of these methods as being “performed on” the object that they are called with. For the example above, you would need to create an object like `MyClass myObject = new MyClass();` and then you could call the method on that object like this: `myObject.myMethod()`. Instance methods are usually used for methods that would affect or use the object that they are called on, while static methods perform an action that does not rely on any information from an object.

Getting Started

To get started, first download the starter code [here](#). Click on the dropdown at the top that says PA7 Starter Code and then download. This will download a zip file. Unzip it and you are good to go. You will only modify the `PA7.java` and the `Image.java` file.



Note: If you are not logged into Google, you will see a different interface. In this case, there will be a button on the top-right corner reading “Download All”. Click that button.

Important: Do not change the original folder structure, that is, do not rename/move files around or the commands to compile and run below will not work.

You may add more images into the `images/` folder for testing and for Part 2 of the assignment.

Inside `PA7.java` you will find one `import` statement that is the same as in PA6. The documentation for the most relevant methods are provided below. You will also be adding some methods in `Image.java` yourself that you will be using in `PA7.java`. You may find more documentation for the `Color` class in the official java documentation website [here](#).

Before you write any code, first try to compile and run the starter code. If you encounter any problems compiling or running the starter code, please notify the staff on Piazza. (instructions on how to compile and run below)

Compiling and Running

Like in the previous PAs, you have to tell java how to find the classes that your code uses that are not built-into java. In this PA, we will be using the `CSE8ALib` class, the `ImageLibRef` class, and the `SimpleImage` class (used in the provided code for `Image.java`) inside `media.jar`. We do this by specifying the classpath with the `-cp` flag.

Make sure that you are in the correct directory on your terminal/cmd. If not, first change directory (`cd`) into the correct directory. The starter code should compile before making any modifications.

Compiling

For MacOS/Unix: `javac -cp "media.jar:." PA7.java Image.java`

For Windows: `javac -cp "media.jar;. " PA7.java Image.java`

Running

For MacOS/Unix: `java -cp "media.jar:." PA7`

For Windows: `java -cp "media.jar;. " PA7`

Relevant Documentation

Similar to the last PA, here is some documentation for the classes you will be working with.

Color documentation (java.awt.Color)

| | |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------|
| Color(int red, int green, int blue) Constructor that creates an RGB color with the specified red, green, and blue values in the range (0 - 255). | |
| int | getRed() Returns the red component of this Color object. |
| int | getGreen() Returns the green component of this Color object. |
| int | getBlue() Returns the blue component of this Color object. |

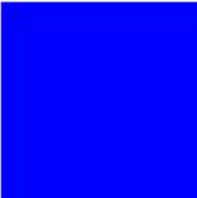

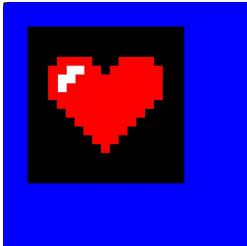
Image documentation (image.Image - defined in media.jar)

| | |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|
| Image(String path) Constructor that creates an image from the file specified by the given path. e.g. <code>new Image("images/cat.jpg")</code> will create a new image from the file <code>cat.jpg</code> | |
| Image(Color[][] pixels) Constructor that creates an image with the given 2D array of pixels. The height of the image is the size of the first dimension of <code>pixels</code> and the width is the size of the second dimension. | |
| int | getWidth() Returns the width in pixels of this Image object. |
| int | getHeight() Returns the height in pixels of this Image object. |
| Color[][] | getPixels2D() Returns the 2D array of Colors representing pixels of this Image object. |
| boolean | explore() Shows the picture in an interactive window. |

Part 1: Modifying images

In this assignment, you will be implementing 5 types of image modifications that are in the starter code. The `canvas` method is a constructor method inside the `Image` class, meaning that it will create a new `Image` object. The other 4 methods are instance methods, meaning that you call them on an object (for example, `img.scaleTo(...)`).

Modifications

| | |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| <p>Canvas Constructor</p> <pre>public Image(int width, int height, Color color)</pre> <p>You will create a new constructor for the <code>Image</code> class that takes a width, height, and <code>Color</code>, and initializes an image with the dimensions provided where every pixel has the given color.</p> |  |
| <p>Crop</p> <pre>public Image crop(int topLeftCol, int topLeftRow, int bottomRightCol, int bottomRightRow)</pre> <p><code>Crop</code> takes row and column indices for the top left corner as well as the bottom right corner of the image and returns an image that contains just the pixels within the rectangular box. Cropping is inclusive of the top left and exclusive of the bottom right. The example is <code>pixel-heart.png</code> cropped with coordinates (100, 100) and (500, 500).</p> |  |
| <p>Overlay</p> <pre>public Image overlay(Image bg, int topLeftCol, int topLeftRow)</pre> <p><code>overlay</code> takes a background image and the coordinates given as a column and row. It produces a new image that has the calling image placed “on top of” the background image, with its top left corner at the given coordinate. That means the pixel at column 0 and row 0 in the foreground image (the calling image) should appear at the pixel with column index <code>topLeftCol</code> and row index <code>topLeftRow</code> in the resulting image; the pixel at column 1 and row 0 in the first image should appear at the pixel with column index <code>topLeftCol + 1</code> and row index <code>topLeftRow</code> in the resulting image, and so on. The entire resulting image should always be the same size as the background image. You may assume that the original image will always fit inside the background image. The example image is the <code>pixel-heart.png</code> image over a blue 1000x1000 canvas.</p> |  |

Chromakey

```
public Image chromakey(Image bg, Color key, double threshold)
```

chromakey takes a background image, a key, and a threshold, and creates a new image that replaces the pixels that are close enough to the key in the original image with the pixels of the background image. To be close enough, the color of the pixel has to have a distance from the key that is lower than the threshold. **The background and foreground images must be the same size.** The distance is in a function given to you in `Image.java` called `colorDistance` and is defined by:

$$\text{distance}(\text{color1}, \text{color2}) = ((\text{color1.getRed()} - \text{color2.getRed()}))^2 + (\text{color1.getGreen()} - \text{color2.getGreen()}))^2 + (\text{color1.getBlue()} - \text{color2.getBlue()}))^2 / 1000$$

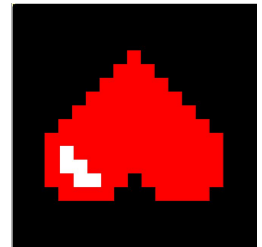
The example on the right is `crane.jpg` after calling `chromakey` with an image of a beach as the background, `Color.green` as the key, and a threshold of 30.



Flip Horizontal

```
public Image flipHorizontal()
```

Flip horizontal returns a new image that is the original image flipped on the horizontal axis. The pixels in the resulting image will be in the same column as in the original image, but the row of the pixel in the resulting image will be that amount of rows from the bottom of the image. For example, a pixel at row 0 in the original image will be in row height - 1 of the resulting image, and pixel at row 1 in the original image will be in row height - 2 of the resulting image.



Testing

As in the previous assignment, the starter code contains one test method for each of the filters above. However, in this assignment, the test methods are in `PA7.java`, and the code it is testing is inside `Image.java`. You do not need to do any testing in the `Image.java` class, and it is already being used in `PA7.java`. In each test method, we have provided you with one test case. Your job is to add two more test cases for each method. These test cases must cover a significant range of possible input values. Remember to test corner cases such as 1x1 images (only one pixel), images with maximum (255) and/or minimum (0) values for the color components, etc.

In order to test your functions, you will use the `explore()` method of the `Image` class. This will bring up an interactive window where you can see the image and inspect the value of each color component of every pixel. The strategy then is to create a small image by manually specifying the values of the 2D array, calling your filter method and then exploring the resulting image to see if each pixel has the color you expect.

Before starting part 2, submit your `PA7.java` file and `Image.java` files to Gradescope and check if you are passing all the tests. You can submit as many times as you'd like before the deadline. Also, include your test cases along with an explanation for why you chose those test cases in your PDF write-up.

Part 2: Collage

Once you've completed the methods above, you will create a collage out of images that you like (or out of the sample images). Create a compound image using at least **four different images from files**, and using at least **3** of the 5 methods you implemented in this PA **at least once**. You can use the methods from PA6, as well! Get creative – you could make a super-custom holiday card, an album cover, a poster, abstract art, and more. Make sure you have the rights to use any photos that you make a part of your collage.

Create your collage in the main method of PA7.java. Then, open it in an explore window, take a screenshot ([instructions here](#) for lab computers), and add it to your submission.

Part 3: Reflection

Once you have finished and submitted your assignment, fill out the reflection form [here](#). Don't forget that EACH STUDENT must fill out their own reflection to get credit.

Star Points

Star points will be given for collages that are particularly creative or ambitious. Feel free to implement additional functions, or do creative things in your main method.

Submission Instructions

There are two separate deliverables for this assignment: your code and the PDF write-up.

Submitting your code:

- 1) Go to [Gradescope](#)
- 2) Select **PA7-Code** and upload your Image.java file.

Submitting your PDF:

- 1) Download the template [here](#)
- 2) Go to [Gradescope](#)
- 3) Select **PA7-Writeup** and submit this PDF as you have in previous assignments