CS161: Introduction to Computer Science
Homework Assignment 10
Due: 4/17 by 11:59pm

Image Processor

In this homework assignment, we'll be working with images! Images can naturally be represented using a 2-dimensional array of Pixel (short for *picture element*) values. This assignment asks you to implement various image-transformation algorithms such as inverting the pixels in the image or rotating the image clockwise.

We'll be using an image format known as PPM (portable pixel map). PPM files are not as well-known as other file formats such as jpeg, gif, etc. However, PPM is an excellent educational format because PPM files are human-readable. The tradeoff is that even small images can be very large in size. To view a PPM file, you will need a special program:

- If you're working on a lab computer in Thompson 409, then the tools are already installed for you to open a `.pgm` file. If you're on Windows, double-clicking on any `.pgm` file will open up the file in Libre OfficeDraw.

- If you're working on your own computer, you can download GIMP for free at `http://www.gimp.org/downloads`. It is compatible with Mac, Windows, and Linux. After you install GIMP, double-clicking on any `.pgm` file will open up the file in GIMP.

For this assignment, you'll be opening and reading in an image, transforming the image somehow (e.g. rotating it), and then saving the transformed image to a new file. To view the result of your transformation, you will need to double-click on the new file you created – i.e., you will not be able to view the transformed file in BlueJ but instead will manually open the file by double-clicking.

Getting Started

Download the starter code for the homework assignment. After you unzip it, you should see the following files:

- `test.pgm, test2.pgm, test3.pgm`: Three small images for debugging.

- `grumpycat.pgm`: An image of Grumpy Cat.

- `nasa.pgm`: A washed-out aerial image from NASA.

- `Image.java`: This is the Image class, which represents a greyscale image and has methods to manipulate the image. There are several methods left for you to implement.

- `Controller.java`: Put code here to test the methods in the `Image` class.

Download GIMP if you are working on your own computer. Double click on `grumpycat.pgm` and `nasa.pgm` to make sure you can open `.pgm` images.

The `Pixel` Class

We'll be working with greyscale (black and white) images where each pixel is a single value in the range $[0, 255]$. A pixel value of 0 is black and a pixel value of 255 is white. As the pixel value ranges from 0 to 255 the color changes from black, to gray, to white.

Create a new Java class called `Pixel`. The `Pixel` class should contain the following methods:

- A constructor that takes in an *integer* pixel value. You should check that the pixel value is in the range $[0, 255]$. If the user passes in a negative number, set the pixel value to 0. If the user passes in a value larger than 255, set the pixel value to 255.

- `getValue()` and `setValue()`: getter and setter methods for the pixel value. The `Image` class depends upon your getter method being named `getValue()`

- Write a method `invertIntensity()` that inverts the pixel value from black to white or from white to black. To invert a pixel, you should subtract the pixel value from 255. For example, if the pixel value is 0 (i.e. black) then the inverted pixel value would be $255 - 0 = 255$ (i.e. white). Or, if the pixel value is 150 then the inverted pixel value would be $255 - 150 = 105$.

- Write a method `logIntensity()` that takes as input a double `coeff` and sets the pixel value to the following:

$$\text{pixel} \; = \; \text{coeff} \; * log(\text{pixel} \; + 1)$$

  You can use the `log()` method in the `java.lang.Math` class. Note that the `log()` method is a static method so it is associated with the class (`Math`) as opposed to an instance of the class.

- Finally, write a method `powerLawIntensity()` that takes as input two doubles `coeff` and `gamma` and sets the pixel value to the following:

$$\text{pixel} \; = \; \text{coeff} \; * \text{pixel}^{\text{gamma}}$$

  You can use the `pow()` method in the `java.lang.Math` class to compute the pixel value raised to the power of gamma.

For each of these methods, you should make sure that the pixel value is always an integer value between $[0, 255]$. Just like the constructor, if the pixel value is less than 0, set the value to 0. If the pixel value is larger than 255, set the pixel value to 255.

Image Class

Open the Image class that has been provided. Notice that the instance variables and the constructor have already been written for you. The instance variables are the 2-dimensional array of Pixels (representing the image) along with the height of the image (i.e. the number of rows) and the width of the image (i.e. the number of columns). In addition, at the bottom you will find the following methods already implemented:

- `public void open(String filename)`: Opens the given filename and reads it into the 2D array.

- `public void save(String filename)`: Saves the current Image to the given filename.

- `public String toString()`: Returns the image in String format.

You should briefly read through these methods to familiarize yourself but do not modify them! Add the following methods to the image class:

- A method `inverseTransform()` that inverts all the pixels in the image. This method should take no inputs.

- A method `logTransform()` that takes as input a double named `coeff` and converts every pixel to the appropriate log intensity value.

- A method `powerLawTransform()` that takes as input two doubles `coeff` and `gamma` and converts every pixel to the appropriate power law intensity value.

To test your methods, you can add code to the `main()` method in the `Controller` class. For example, consider the following code:

```
public static void main(String[] args){
    Image img = new Image("grumpycat.pgm");
    img.powerLawTransform(.75, 1.5);
    img.save("grumpycat_transformed.pgm");
}
```

After running the `main()` method, double click on the `grumpycat_transformed.pgm` file and it should automatically open. Your grumpy cat should now look like:
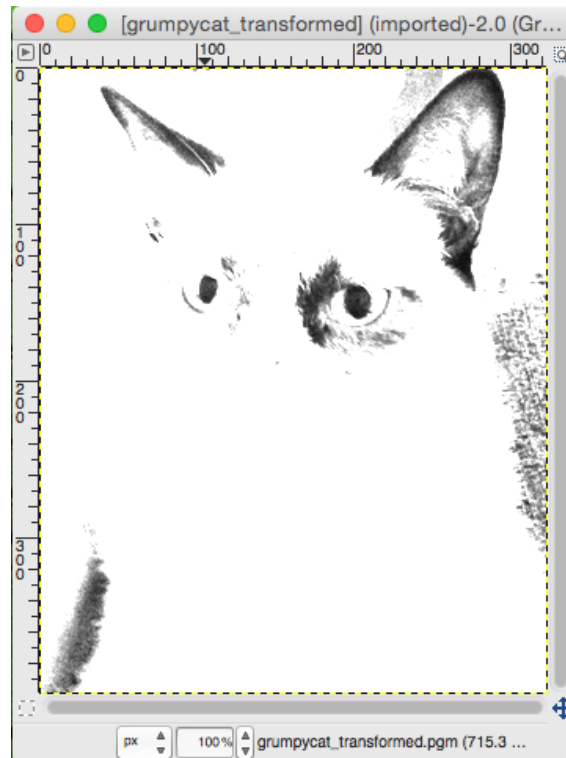
More Transformations

The following transformations should flip and rotate the image when called:

- A method `flipVertical()` that flips the image across the vertical (y) axis. In other words, the pixels on the left-hand side of the image get swapped to the right-hand side and vice versa.

- A method `flipHorizontal()` that flips the image across the horizontal (x) axis. In other words, the pixels on the top of the image get swapped with the pixels on the bottom and vice versa.

- A method `rotateClockwise()` that rotates the image clockwise. Note that after rotating the image, the dimensions of the image should be swapped. For example, if the original image has 10 rows and 3 columns then the rotated image will have 3 rows and 10 columns.

- A method `rotateCounterClockwise()` that rotates the image counter-clockwise. Again, after rotating, the dimensions of the image should be swapped.

▬▬ Style Guide ▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬

Before you submit your assignment, go through the checklist below and make sure your code conforms to the style guide.

Checklist
- ☐ All instance variables are declared private
- ☐ All instance variables are initialized in the constructor!
- ☐ All unused variables are deleted
- ☐ All instance variables are used in more than one method (if not, make them local)
- ☐ Javadoc comment for all classes
- ☐ All methods have Javadoc comments (except for the `main` method)
- ☐ All numbers have been replaced with constants (i.e. no magic numbers)
- ☐ Proper capitalization of variables, methods, and classes
- ☐ Use white space to separate different sections of your code

Read the "Style Guide" (under "Resources" on the course website) for more information.

▬▬ Submitting your homework assignment ▬▬▬▬▬▬▬▬▬

You should submit your `hw10` folder with your `Pixel`, `Image`, and `Controller` class inside. Please rename your folder with both your first and last name *before* you zip it!