# CS-207: Programming II

# Northeastern Illinois University

# Research Lab: Image Processing

# Due: [insert date]

## Goal:

The goal of this research lab is to practice navigating the Java documentation and to practice creating a Java class. You will do this by learning about image manipulation with 2-dimensional arrays. For the research component, you will need to investigate and analyze the provided code, and write additional code to complete the task.

## The Problem:

You have just started a new job, and the programmer before you left a partially finished project, which reads in an image file and display the image. Your boss wants you to create a new class to manipulate the image by mirroring, rotating, and changing it to grayscale. You will create this new class and the methods necessary to complete your assignment.

## Instructions:

* You should work in groups of 2-3 individuals. Groups of more than 3 are not permitted.
* Each group should submit ONE lab write-up. It is the responsibility of each group member to ensure that their name is on the write-up.
* The lab write-up should be typed! Type each question (and the question number) followed by your group’s answer. Convert your lab write-up to a .pdf.
* You should use complete sentences and proper grammar in your write-up. Use spell-check! This counts as part of your grade.
* Do not copy/paste directly from your sources for your answers (this is called plagiarism). Instead, you should re-word the information in your own words.
* Place your files (1 PDF, 2 Java files, and screenshot) in a zip folder and submit to the D2L dropbox by the specified due date.
* Required: Submit a group selfie! This should not be taken in our Programming II classroom or in a hallway, but should either be a group selfie via Google hangouts (See this for an example: http://i.imgur.com/WLLxEEO.png) or a group selfie of all of you sitting in a common area and working. Your entire group must be in the picture and you should submit it as a .pdf, .jpg, or .png file.
* Each member of the group must turn in a readable digital copy of the peer assessment to an individual D2L dropbox by the assigned due date and time. The peer assessment counts as a significant part of your grade and you will receive a zero for that portion of the research lab grade if you do not turn it in.

## Part A: Getting Started

Research is often an ambiguous process, so it can be difficult to figure out how to get started on a particular problem. In this lab, we are going to imitate a scenario that happens frequently in software development: working with code that you did not write! To start, download the project files.

## Question #A.1

You have been given 3 files: an image file called animals.jpg, and two Java classes: DisplayImages.java and ManipulatedImage.java. Open the DisplayImages class, and read the class header. Is there a keyword that you do not recognize?

One of your most important tools as a developer is Google! Try searching for “java [name-of-keyword] keyword” and see what you can learn in 1-2 minutes of reading.

In Programming 2, you learn about the four fundamental concepts of Object-Oriented Programming (OOP): Encapsulation, Inheritance, Polymorphism, and Abstraction. Which of these concepts is related to the new keyword?

## Question #A.2

Locate the main method. How many lines of code does the main method contain?

This launch method is an important part of Java application. Look at the class header, and note that DisplayImages extends Application – this program will create an application, which is a different type of program than the ones we created in Programming 1. (To learn about applications, enroll in CS-319!) This application is made using JavaFX, which is a set of packages that will allow us to display and manipulate our image. Look up Application in the Java 8 documentation. Read the section called “Life-cycle” – what five steps happen whenever a JavaFX Application is launched?

## Question #A.3

Luckily, you don’t have to be a JavaFX expert to take advantage of the code that’s already been written! The programmer before you has provided some useful comments within the start method. Read through the comments. What are the different sections of this method?

Your only task is to create the methods which modify the image – therefore, which sections are most important for you? If a section doesn’t contain any references to ManipulatedImage objects, you do not need to understand it in-depth.

## Question #A.4

Take a closer look at the important sections you identified. Write a few sentences to answer the following: what do these sections have in common, and in what ways are they different? In addition, what methods are called, and what objects are created?

## Part B: Outlining the ManipulatedImage Class

## Question #B.1

You will not modify anything in the DisplayImages class, but you need to understand how it interacts with the ManipulatedImage, so that you can write the method appropriately. Look in the start method – list all the ManipulatedImage objects which are created.

## Question #B.2

Now list all of the methods which are called on the ManipulatedImage objects you found. Do any methods have arguments? What is the return type of each method? It can be tricky to figure this out, but you can make educated guesses, and correct them later if needed.

## Question #B.3

Find the ManipulatedImage constructor(s). How many different constructors are there? What type of argument(s) do they take?

## Step #B.4: coding only

Open up the ManipulatedImage file. You have only been provided with the import statements and class header. Remember that every class can contain the following:

* Variable(s) – instance or static
* Constructor(s)
* Method(s)

Good news: you have already created a list of constructors and methods!

You will need 5 instance variables: two integers called height and width, which will store the dimensions of your image. You will also have two object instance variables: an Image object called img, and a WritableImage object called writableImg. Finally, you will have a 2D array filled with objects of the Color class; call this array colors. Be sure to properly encapsulate each variable!

## Step #B.5: coding only

Create the stubs for your constructor(s) – this means that you write only the header line of the constructor, and the lines with opening and closing curly braces, but you leave the inside of the constructor blank (for now!). This is a good way to stay organized when you write a new class.

## Step #B.6: coding only

You defined some of your methods in #B.2. Write the stubs for each method – again, this means to only write the method header, and not to write any code inside the curly braces.

You will need two other methods: setter methods for colors and writableImg. You are already familiar with setter methods: what should the setter methods be called? What should the parameter(s) and return type be?

## Part C: Writing methods

## Step #C.1: coding only

The constructor will have 6 lines of code – first, create your img instance variable by calling the Image constructor – for the parameter, pass in the parameter of your ManipulatedImage constructor.

Set the values of width and height by calling this.img.getWidth() and this.img.getHeight(), respectively. Be careful, because these methods return double values – how can you change a double into an integer?

Next, call the setColors method, which you will write later. Initialize the writableImg object by calling the WritableImage constructor. Look up WritableImage in the JavaFX 8 documentation; use the constructor that takes the height and width variables, and be sure to enter them in the correct order!

Finally, call the setWritableImg method, which you will write later.

## Step #C.2: coding only

getWritableImg, like any getter method, only needs one line of code – a return statement. Write this method.

## Step #C.3: coding only

Images on a computer can be thought of as a grid of pixels – each pixel contains a single color, and an image can contain hundreds or thousands of pixels. You can represent the grid of pixels using a 2D array, which is what you will do with the colors array! Each cell of colors contains one Color object.

Images and arrays are laid out in the same way, where cell [0][0] and pixel (0, 0) are both in the upper leftmost cell. To write the method, setColors, you will use the PixelReader object to read the Color of each pixel in img.

First, initialize colors; you will need to use the values of width and height to set the size of the array. Next, create the PixelReader object: PixelReader reader = this.img.getPixelReader();

Finally, you will set the value of each cell of colors by calling reader.getColor(column, row). This method call returns a Color object. You will need to do this for every pixel in the image – so you should use nested for-loops!

## Step #C.4: coding only

Now, write the setWritableImg method. You can’t modify the pixels of an Image object, but you can modify the contents of a very similar class: WritableImage. For this method, you will need to create a PixelWriter object, which is the complement of the PixelReader you created in #C.3. In fact, this method will look very similar to the setColors method you just wrote!

First, create your PixelWriter by writing the following line:

PixelWriter writer = this.writableImg.getPixelWriter();

Now, you will use writer to set the value of each pixel of writableImg. You can do this by calling writer.setColor(column, row, this.color[row][column]). Again, use nested for-loops so you repeat this action for each pixel.

## Part D: Image Manipulation

## Question #D.1

Below is the code for each of the last 2 methods, rotate90 and mirrorY. rotate90 rotates the image 90 degrees counter-clockwise, and mirror should mirror it across the Y-axis. Take a look at each snippet of code – how can you figure out which method is which? Write 1-2 sentences to describe what is happening in each code snippet. If you’re unsure, try to trace through a few iterations of the loops.

//Mystery Method A

Color[][] newColors = **new** Color[colors[0].length][colors.length];   
**for**(**int** i = 0; i < **this**.colors.length; i++)  
{  
 **for**(**int** j = 0; j < **this**.colors[i].length; j++)  
 {  
 newColors[j][i] = **this**.colors[i][j];  
 }  
}  
**this**.colors = newColors;  
**this**.writableImg = **new** WritableImage(colors[0].length, colors.length);  
**this**.setWritableImg();

//Mystery Method B

Color[][] newColors = **new** Color[colors.length][colors[0].length];  
**for**(**int** i = 0; i < **this**.colors.length; i++)  
{  
 **for**(**int** j = 0; j < **this**.colors[i].length; j++)  
 {  
 newColors[i][j] = **this**.colors[i][**this**.width - 1 - j];  
 }  
}  
**this**.colors = newColors;  
**this**.setWritableImg();

## Question #D.2: coding only

Copy the code snippets into the correct methods in your ManipulatedImage class.

## Question #D.3:

In order to write the grayscale method, investigate the Color class to see how it works: look up Color in the JavaFX 8 documentation. Scroll down to look at the constructors – notice that there is only one constructor. What are the parameters, and what is the valid range of values for each?

## Question #D.4: coding only

For our purposes, the value of opacity will always be 1.0. The other parameters are color values –various combinations of these parameters will create any perceptible color. When all of the color values are equal to each other, the color is gray.

A quick way to transform a color into a gray is to find the average of the values, then set each to the average. For example, the average of 0.1, 0.4, and 0.1, is 0.2, so we could change the values to be 0.2, 0.2, and 0.2.

Create an object of the Color class and call it temp. Set temp equal to the Color object located at colors[0][0].

Search the Color documentation for 3 getter methods you can use to obtain the 3 color values.

Write the code necessary to get these values, and then find their average. Then, create a new Color object, using the average for all three values. Set colors[0][0] to point to this new Color object.

## Question #D.5: coding only

Now, put this code inside for-loops, and modify it slightly so that it will execute on each individual pixel in colors. Finally, call setWritableImg.

## Question #D.6:

Compile both classes, and run DisplayImages. Take a screenshot of your results! Place your ManipulatedImage.java file, screenshot, and lab write-up in a ZIP file, and submit it to D2L.

## Part E: Final Summary

Congratulations; your hard work paid off! Whenever you complete a project, it is important to assess what you think went well and what you need to improve on.

## Question #E.1

What was the most challenging part of this research lab for your group?

## Question #E.2

What did your group learn/find the most useful by doing this research lab?

## Question #E.3

What topics covered in your Programming II class (either in lectures or textbook reading) did you use in this lab?

## Question #E.4

What was the most fun aspect of doing this research lab?