CSE 8A Programming Assignment 8

Fall 2019

Due Date: December 04, 2019

Learning Goals:

- Inplace modification of 2D objects.
- Create a class from scratch

Submission:

There are three deliverables for this assignment (plus the survey) 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 the Slideshow.java file to be autograded
- Deliverable 3: You will submit a PDF write-up using the template here

---

Overview

- Part 1: Students add methods to modify Images in place
- Part 2: Students implement a slideshow class
- Part 3: Each student individually completes a (slightly longer) reflection and feedback survey

Getting Started

To get started, first download the starter code here . Click on the dropdown at the top that says PA8 Starter Code and then download. This will download a zip file. Unzip it and you are good to go. You will modify the PA8.java file and 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 and sounds into the images and sounds folder for testing and for Part 2 of the assignment.

Inside PA8.java you will find two import statements. These are the classes you will be working with. The documentation for the most relevant methods are provided below. 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 are given for each part of the assignment)

All the reference material can be found at the end of this document in the appendix section.

Part 1: Modifying images in place

NOTE - Comment out the line '`private static PictureFrame pf = new PictureFrame();`' and the whole 'show' function in Image.java for this part of the assignment. You will only need that for the part 2.

In this section you will write code in Image.java (your image filtering methods) and PA8.java (your tests). You will also use code we provide in the media.jar file, as in PA7.

Compiling and Running

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:." PA8.java Image.java`
For Windows: `javac -cp "media.jar;." PA8.java Image.java`

Running

For MacOS/Unix: `java -cp "media.jar:." PA8`
For Windows: `java -cp "media.jar;." PA8`

1.1 Implement two methods in the Image class

In this part of the assignment, you will implement image filters like you have in your previous assignments but with one difference, **the modifications for this assignment needs to be done in place** i.e. you should not create a new Image object for storing the modified pixel values. The pixel values should be changed for the calling Image object.

Why use In Place? - Saves Space! Since you will not create another Image object to store the modified image you use less space.

You will be implementing two filters, one will be the flip horizontal square filter and the second one can be anything of your choice. The only requirement here is to modify the images in place when using these filters.

Important: Do not change the method signature (name, return type and parameters) of the methods defined in the starter code. After implementing the methods, replace the dummy return value with the correct one.

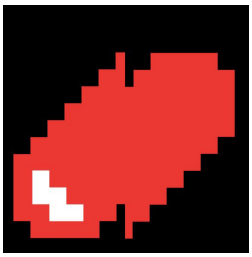
Method 1: Flip Horizontal Rect

`public void flipHorizontalRect(int topLeftX, int topLeftY, int width, int height)`

This function receives the topLeftX and TopLeftY coordinates and width and height for the rectangle which needs to be flipped in the given image on the horizontal axis. The rectangle starting from `topLeftX` and `topLeftY` that is `width` pixels wide and `height` pixels tall in the calling object will be flipped. This transformation shall be done in place in this function and a copy of the original image should not be created.

The image on the right shows the resulting image when flipHorizontalRect is called on the heart image with the parameters :

topLeftX =0, topLeftY=0, width = original image Width/2 and height = original Image Height



Method 2: Your Custom Filter

You can choose a filter that you have already implemented here or get creative again and develop a new filter.

You need to add a new method for this filter in the Image.java file. You should name this method `public void customFilter(int topLeftX, int topLeftY, int width, int height)`

Here are the requirements for your filter:

- It must modify the calling object in place. It should not return anything.
- It must include the parameters `int topLeftX, int topLeftY, int width, int height` as in the previous method that specify a region of the image to modify. (It may also take additional parameters). Only the area inside the specified box should be modified. The rest of the image should remain unchanged.
- It must still work when the bounding box goes outside the range of the calling object's legal row and column values. That is, topLeftX and/or topLeftY might be less than 0, and the width and the height of the box might cause the box to go off the right or bottom of the calling object. If any part of the bounding box is outside the calling object's dimensions, only the portion of the image in the box that is inside the dimensions should be changed.
- You may transform the image within the box using a technique you previously implemented (e.g. grayscale, or even horizontal flip).

1.2 Testing

The starter code contains one test method for flipHorizontalRect filter described above which contains one test case. Your job is to add one more test case for this filter and also create another test function for the filter of your choice. This test function shall be called from the main function and should have at least two test cases. The 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 total, you should have two test functions (one for each filter) .

- Test for flipHorizontalRect - You are given the test method for this function in the starter code along with one test case. You need to **add another test case** inside this method.
- Test for your custom filter- You need to define a test method for the filter you create. This test method should have **at least three test cases** including the tests with a bounding box fully inside the image, one with a bounding box where topLeftX or Y is < 0 and one where the bounding box starts inside the image and goes off the right or bottom. Your filter should modify the part of the rectangle within the image. You can have additional test cases in order to show the correctness of your filter. Make sure you have explanatory screenshots in your report with the image before the filter and the image after the filter was applied for each of your test cases.

In order to test your filters, 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.

Preparing to submit Part 1

You will submit your Image.java file to the Gradescope assignment **PA8 PART 1-Code**
Make sure you have commented out '`private static PictureFrame pf = new PictureFrame();`' and the whole 'show' function for this part before submitting to autograder.

In the writeup template place the following in the proper locations:

- The code for your custom filter method (including the method header and body)
- The testing code for both the flipHorizontalRect and your custom filter method
- The results of running your custom filter method (this should be screenshots showing the tests working correctly)
- A description of each test of your custom filter method including why the test was selected and how the output shows it works correctly.

Part 2: Creating the slideshow class

NOTE - Uncomment the line `'private static PictureFrame pf = new PictureFrame();'` and the whole 'show' function in Image.java for this part of the assignment.

In this part you will be creating a slideshow of images and sounds . The purpose is to give you a feel for how to create a class from the bottom up, starting with a completely blank file.
You will be creating a program that can display a slide show made of images and sounds.

When we run you code:

- We should see a series of images ("slides") appear one at a time, accompanied by a sound. The files you would like to use can simply be uploaded in the same directory as the source code for convenience, and then hard coded into the program.
  Each image will be accompanied by exactly 1 sound
- After the sound is done playing, **the currently displayed image will be closed**, and the next image will be displayed.
- After the slideshow is completed the program should ask the user if they want to replay the slideshow, if the user enters 'yes' the first two bullet points should be repeated. The program should stop when the user enters 'no' for this prompt.

In this part of the assignment you will be writing all your code in the Slideshow.java file, which you will create (it is not part of the starter code). You will also be working with the Image class and a new Sound class, as well as using code from media.jar as in PA7. Please look at the end of this writeup for documentation on methods you will need to use in each of these classes.

Compiling and Running

Compiling

For MacOS/Unix: `javac -cp "media.jar:." Slideshow.java Image.java Sound.java`
For Windows: `javac -cp "media.jar;." Slideshow.java Image.java Sound.java`

Running

For MacOS/Unix: `java -cp "media.jar:." Slideshow`
For Windows: `java -cp "media.jar;." Slideshow`

2.1 Create a class named Slideshow with the following fields and methods

PUBLIC Fields:

- `sounds` : An array of `Sound` objects where each object represents a sound.
- `pictures` : An array of `Image` objects where each object represents an image.

IMPORTANT - In general you must have your fields as private values, but in order to facilitate automated testing for your code in this part we ask that you make your fields public.

Public Methods:

Default constructor

This constructor should take no arguments and should initialize each field to an array of length 0. Yes, it's weird to have an array that is not capable of storing any elements, but it's possible..

Add a Slide

```
void addSlide(Image newImg, Sound newSound)
```

makes new arrays and copies everything over to accommodate the new Image and the new Sound. The size of the new Image(or Sound) array would be 1 greater than the original array of images (or sounds) .

Play the Slideshow

```
void play()
```

Shows each image while playing the corresponding sound. It should display each Image for the duration of the sound. When the sound is over, it should display the next Image and play the next sound and so on until it has shown all the images and played all the sounds.

(HINT - You must use 'blockingPlay' function given in the sound documentation to halt the program while the sound completes playing. Also, you must make use of 'show' function given in the image documentation to open an image.)

Your workflow in this function should look like -

- **Open an image Ex :** pictures[0].show(); - this makes the first Image in the array of Images visible (or opens it as a pop up)
- **Play the sound while blocking the program.**
- **Repeat for all of the images and sounds in the arrays**

2.2 The slide show program (and testing your methods)

In the main method of the **Slideshow** class, implement the functionality described at the top of this section. That is your main method should:

- Create a new Slideshow object
- Add at least 3 slides (image + sound) of your choice
- Play the slideshow

- Ask the user if they want to see the slideshow again. If they type "yes", replay the slideshow. If they type anything else, exit the program.

Test your program by running it and choosing to repeat the slideshow at least once.

Preparing to submit Part 2

You will submit your Slideshow.java file to the Gradescope assignment **PA8 PART 2-Code**
You only need to submit your constructor code and the addSlide method code to the autograder. DO NOT submit the 'play' method (you can comment this out) . We will be checking your play method as a part of the writeup.

In the writeup template place the following in the proper locations:

- Provide a brief description of the slideshow. What is the topic of the slideshow ? Describe what individual slides in you slideshow represent.
- Attach your code from your play() method in the Slideshow class
- Attach your code from your main method in the Slideshow class
- Justify why you think your program is correct and also mention any known issues with your code.

Part 3: Reflection [LINK OPEN NOW]

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 **PA8 PART 1-Code** and upload your Image.java file.
3) Select **PA8 PART 2-Code** and upload your Slideshow.java file

Submitting your PDF:

1) Download the template here
2) Go to Gradescope
3) Select **PA8-Writeup** and submit this PDF as you have in previous assignments

Appendix

Below is the documentation for the classes you will be working with. Note that constructors are what you use with the new keyword to create new objects. e.g. Color c = new **Color(100, 100, 100);**

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(**String** path) **Constructor** that creates an image from the file specified by the given path. e.g. new Image("images/cat.jpg") will create a new image from the file cat.jpg	
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 pixels 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 a copy of the 2D array of Colors representing pixels of this Image object.
void	**show()** Shows the image in a defined frame.

void	**explore()** Shows the picture in an interactive window.

Note - Use show() function in your part 2 of the assignment in order to show images. If you use explore the images will pop up in different windows and thus the essence of a slideshow will be lost.

Sound documentation

Sound(**String** path) **Constructor** that creates a sound from the file specified by the given path. e.g. new Sound("sounds/play.wav") will create a new sound from the file play.jpg	
Sound(int[] soundIn**)** **Constructor** that creates a sound from array of integers specified by soundIn..	
boolean	**play**() Plays the sound file
boolean	**blockingPlay**() Plays the sound file and blocks the program until the sound finishes playing.