PSA 4 : Transforming Pictures

Read all instructions in this document before starting any coding!

START EARLY, START OFTEN!

Due: Sunday, November 5th, 11:59pm

This assignment is aiming to help you better understand in nested loops in the context of geometrical image processing. Moreover, we will work on some cool picture effects such as flipping part of a picture and combining several pictures together.

Disclaimer: These videos are from last year's PSA. Some content might be changed from last year. So please follow the instructions from this document only! However, the explanations in the videos are still very relevant! Overview video for 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 psa4
cd psa4
cp ../../public/psa4/* .
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 TestCollage.java, TestFlip.java, Picture.java, vertical-rectangle-200-200-100-300.bmp, horizontal-rectangle-200-200-100-300.bmp, vertical-rectangle-100-100-200-200.bmp, horizontal-rectangle-100-100-200-200.bmp, and coding.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 psa4 folder on your machine.
2) Download zip and unzip it inside the psa4 folder: LINK HERE

Problems:

Part A: Collage - (8 points)

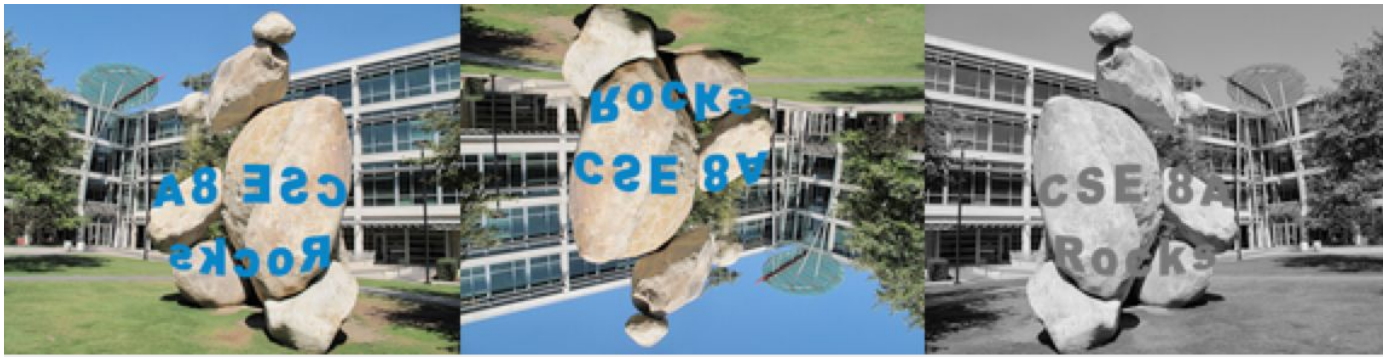
In this part of the PSA, we will make an image collage by combining three images together. Obviously we will create some cool special effects on images before combining them into a collage. To help you understand the process better, **please answer the following questions in the header comments of TestCollage.java before coding anything for part A**. TestCollage.java is the file you should create in order to see and test out the methods you implement in Picture.java. See detailed instructions below. **(1 points)**:

```
/* Filename: TestCollage.java
* Created by: CSE 8A Staff
* Date: Fall 2017
* Consider the following 3x3 Picture, each Pixel labeled 1-9.
* | 1 | 2 | 3 |
* | 4 | 5 | 6 |
* | 7 | 8 | 9 |
* 1) Place the new location for each pixel if the Picture is flipped:
*  horizontally          vertically
* |   |   |   |          |   |   |   |
* |   |   |   |          |   |   |   |
* |   |   |   |          |   |   |   |
* 2) When should you stop looping when performing a horizontal flip? How about
*    vertical?
*
* 3) If you wanted to place two 3x3 Picture side-by-side into a canvas,
*    what width and height should destination canvas have?
*/
```

Your task in Part A is to create a collage! But it won't just be any collage...We have some guidelines for you:

- The collage will consist of 3 identical images side by side
- The first image will be flipped horizontally

- The second image will be flipped vertically
- The third image will be grayscale
- See sample collage below:



To achieve this effect, you will write the following three transformation methods in Picture.java:

- `flipHorizontal()`: Will flip the calling Picture object horizontally **(1.5 points)** See image 1 of the collage.
- `flipVertical()`: Will flip the calling Picture object vertically **(1.5 points)** See image 2 of the collage.
- `grayscale()`: Will change the calling Picture object to grayscale **(1 point)** See image 3 of the collage.

Remember, use the "`this`" keyword to refer to the calling object. We would also suggest using the nested for loop with `this.getPixel()` strategy of accessing and modifying your picture's pixels as shown in the textbook, as opposed to the Pixel array strategy we used in Lab 3.

Important Note: You **must** implement these three transformations such that they will operate on the **calling object itself**. That is, you will **not** be taking a parameter in any of these 3 methods.

Example:

```
Picture sourcePicture = new Picture(<some file path>);
//This will transform sourcePicture
sourcePicture.flipHorizontal();
//This will show the transformed picture
sourcePicture.show();
```

We have attached a prototype of `flipHorizontal()` in the included Picture.java file to help you out a bit. Please refer to that method, and **read the comments we have left you inside of it carefully!** There are a few hints there!

Finally, you must also write the following method in Picture.java:

`collage(Picture[] pictures)`: (2 points)

- Will create a collage out of the pictures included in the input parameter array. **Note this is an array of Pictures. In other words, each element in this array is a Picture reference!** When writing this method, you can assume that all the pictures in the array are the same dimensions.
- The collage will be created on a canvas. The width of the canvas should be EXACTLY the sum of widths of input array of Pictures. In our case, we are transforming just one Picture into 3 different styles. Hence the width will be is 3 times the width of the Picture. In other words, you won't be putting the collage on an unnecessarily large canvas like on page 152 of your textbook (So much wasted space!). You can assume that this method will ONLY be called by a canvas Picture object of appropriate size for the pictures to be included in the collage.

We have included a prototype of `collage()` in Picture.java to get you started. It looks a little different from the collage example you saw in your textbook, but hopefully it will be easier to write too. Again, see the comments in there for some (what I hope are going to be) helpful tips!

Testing and Creating the Collage (1 point)

In TestCollage.java, you will define a main method that will do the following:

- Bring up the file chooser that allows you to choose a photo from your media resources directory
- Show the original picture
- Create a canvas that is the same height as the one you chose, but 3x as wide. You can use this method to do this: `Picture canvas = new Picture(width, height);` You'll just have to plug in the appropriate width and height (they are both ints).
- Copy the original picture twice, so you have 3 picture objects, each with an identical photo. The following Picture constructor is useful here:

  ```
  Picture copy1 = new Picture(originalPicture);
  ```
- Call each of your transformation methods on each Picture to get 3 pictures that have been transformed in 3 different ways
- Show each transformed photo **individually**
- Construct your array of transformed pictures
- You will find the following statement useful: `Picture[] picArray = new Picture[3];`

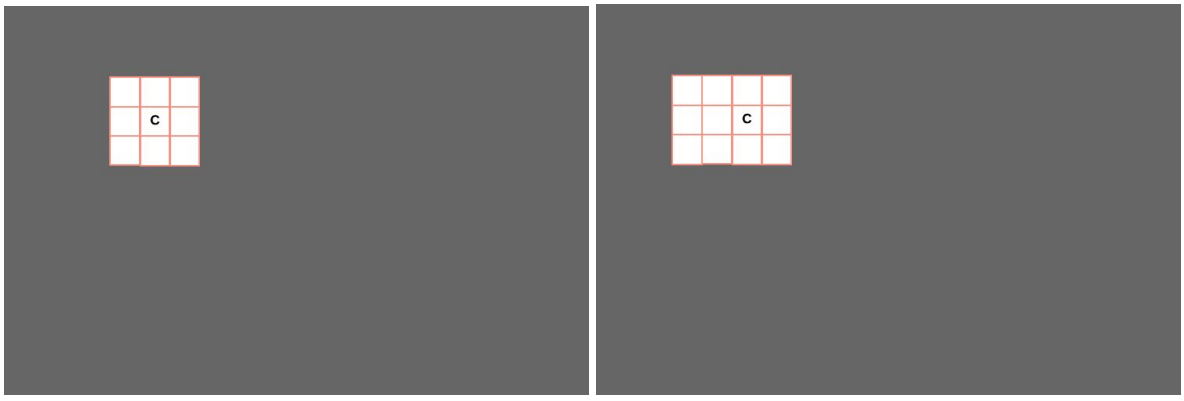
- It creates an empty array that can store references to 3 Picture objects. You will need to set each position in the array to refer to your 3 picture objects.
- Use your canvas object to call `collage(Picture [] pictures)`
- Show the canvas with the complete collage on it.

We've left you comments in TestCollage.java as well to help guide your workflow. **READ THEM CAREFULLY.**

Part B: Picture Flip (9 points)

In this part of the PSA, we will implement methods that will allow us to flip any rectangular area in a picture. Again, to help you understand the process, we will provide you with some lead questions. **Answer the following questions in the header comments of TestFlip.java before starting (1 point):**

```
/* Filename: TestFlip.java
* Created by: CSE 8A Staff
* Date: Fall 2017
* Consider the following 3x3 and 3x4 Picture sections from a larger Picture,
* centers labeled with C whose coordinate is (x, y).
```

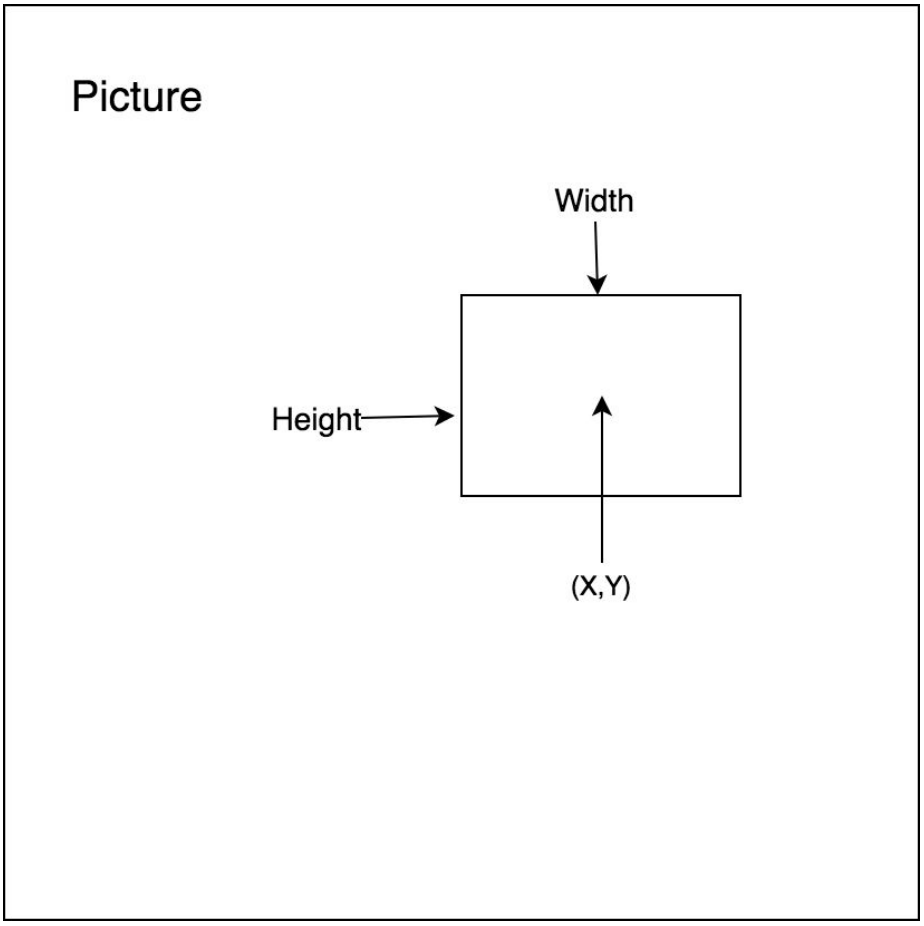



```
* 1) How would you use the height and width of the 3x3 section to find its
*    top left corner, given the coordinates of its center C?
*
* 2) What if the grid of pixels we focus on is a 3x4 region? Pay attention to where
*    we define the center of the grid if a dimension is even.
*
* 3) Can you generalize the finding if the region we focus on is a w by h region
* and the center has a coordinate of the center is (x, y). What is the coordinate
* of the upper left corner of that region?
*/
```

Description: Your task in Part B is to create methods to flip only a **piece** of a picture in either the horizontal or vertical orientation.

You will write the following two methods inside of Picture.java:

- `public void flipVerticalRectangle(int x, int y, int width, int height):` Will flip a rectangle of dimensions length by width and height in the vertical orientation in the calling object **(x value of the pixel remains the same)**. The position of this rectangle is defined by x and y, where (x, y) denotes the coordinates of the center of the rectangle. See the following diagram for clarification on this point.
- `public void flipHorizontalRectangle(int x, int y, int width, int height)`**:** Will flip a rectangle of dimensions length by width and height in the horizontal orientation in the calling object **(y value of the pixel remains the same)**.The position of this rectangle is defined by x and y, where (x, y) denotes the coordinates of the center of the rectangle. See the following diagram for clarification on this point.



Method description: `flipVerticalRectangle()` (3 points)

- Add a method called `flipVerticalRectangle()` to Picture.java
- This method will take in 4 parameters, the x and y coordinates of the center of the box to flip, and the width and height of the rectangle box to flip.
- You can then use the same algorithm you used for flipVertical() in Part A to accomplish this effect, except instead of iterating over the entire image, iterate ONLY within the rectangle that you've defined with x, y, width and height.

Method description: `flipHorizontalRectangle()` (3 points)

- Add a method called `flipHorizontalRectangle()` to Picture.java
- This method will take in the same parameters as `flipVerticalRectangle()`, except the method should flip the image horizontally within that bounding rectangle.
- Therefore, you can use the same algorithm you used for `flipHorizontal()` in Part A, but iterate only within the bounding rectangle.

Note: Be aware of the about the scenario in which the **rectangle to be flipped is partially outside of the picture**. In that case, proceed to **stop the flip operation** without doing any modification to the original picture using a **conditional statement**. You should do this check before modifying any Pixels so that the original picture does not change. Normally this would give an index out of bounds exception. It's our job as programmers to make code that works given any scenario.

Also consider the corner case regarding the box's width and length. When the box has even length or even width, where should the center be? What happens if the length is even and odd? Here is a simple example. If the box has height 4 and width 4, then the center should be at (2, 2).

		(x, y)	

Testing rectangle flipping (2 points):

To see your program in action, create the following program to test your methods in the TestFlip.java file. Create a main method in TestFlip.java and do the following:

- Create a new Picture using FileChooser.pickAFile().
- Show the original Picture.
- Create two copies of the Picture
- Use one copied Picture object to call **flipVerticalRectangle**() with some arbitrary coordinate for the flipping rectangle (it should flip the portion as long as the coordinates are within the image, otherwise leave the image alone)
- Use the other copied Picture object to call **flipHorizontalRectangle**()
- Show both transformed, copied images

However, testing through observation does not mean that a program is 100% correct. Writing a test program that checks your program pixel-by-pixel is more reassuring than visual examination of pictures. Recall that we created a tester method called testAlphaBlending in PSA3. You should now create one for our flip methods. **Create the following method in Picture.java.**

- **`public boolean testFlipRectangle(Picture result)`**. This method will play a very similar role as the `testAlphaBlending` method we did in PSA3. It basically compares the calling object with the parameter picture. It returns false if any pixel between these two pictures is different. It returns true otherwise. You can read on how to write the method from the PSA3 write-up.
- Note, `testFlipRectangle` will be able to test for both the `flipVerticalRectangle` and the `flipHorizontalRectangle` methods as long as you pass in the correct corresponding flipped pictures.
- You can comment out what you already wrote in your main method in **TestFlip.java** when you were testing your flip methods before. Then in your main method, make a new Picture object that corresponds to the coding.jpg picture.
- Create four new **copies** of that object. Name these objects `copy1`, `copy2`, `copy3`, and `copy4`.
- Call `flipVeriticalRectangle()` on copy1 with parameters (`200, 200, 100, 300`) and on copy2 with parameters (`100, 100, 200, 200`).
- Then call `flipHorizontalRectangle()` on copy3 with parameters (`200, 200, 100, 300`) and copy4 with parameters (`100, 100, 200, 200`)

- In your main method, make four more new Picture objects verticalResult1, verticalResult2, horizontalResult2, horizontalResult2 that correspond to the following pictures.
    - vertical-rectangle-200-200-100-300.bmp
    - vertical-rectangle-100-100-200-200.bmp
    - horizontal-rectangle-200-200-100-300.bmp
    - horizontal-rectangle-100-100-200-200.bmp
- Call your `testFlipRectangle()` method on your horizontal and vertically flipped pictures.
- Print your result. Print a statement like the following:

```
Picture resultVertical1 = new Picture("vertical-rectangle-200-200-100-300.bmp");
System.out.println("Compare flipVerticalRectangle(200, 200, 100, 300) with correct answer
= "+ copy1.testFlipRectangle(resultVertical1));
```

Do the same print statements for:

`flipVerticalRenctangle(100, 100, 200, 200)` comparing `copy2` and `verticalResult2`, `flipHorizontalRectangle(200, 200, 100, 300)` comparing `copy3` and `horizontalResult1`, `flipHorizontalRectangle(100, 100, 200, 200)` comparing `copy4` and `horizontalResult2`.

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 **TestCollage.java, TestFlip.java and Picture.java. Write this as comments at top of Picture.java file.**

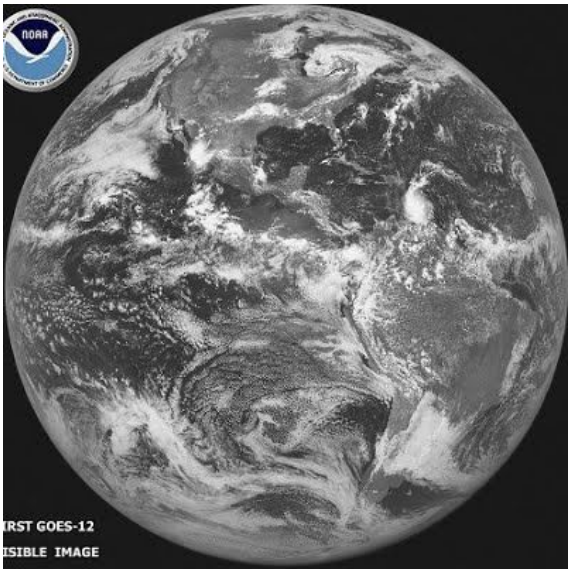
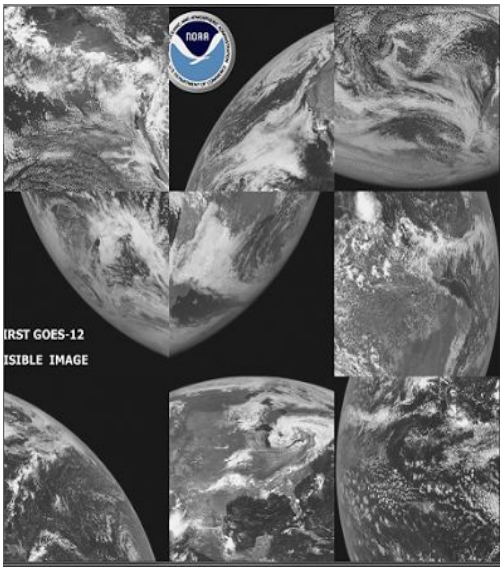
Commenting and Style (1 points)

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

Your star point mission, should you choose to accept it, is to implement a picture scramble method. We will be providing you with 3 images: a 300x300 pixel image, a 600x600 pixel image and a 900x900 pixel image here. Your task is to implement a method (which can call other custom methods you define, if you require it), that divides these images into a 3x3 grid, and randomly scrambles the order of each sector of the grid. See the following image for an example of this.

	
Before scramble	**After scramble**

The reason we are providing you with these 3 photos is so we can avoid any array index errors pertaining to dimensions of an image not being perfectly divisible by three (It makes the implementation simpler too! You can thank us later.). In terms of implementation, you can choose whatever algorithm/strategy you like. The purpose of this star point assignment is to test your algorithmic creativity. However, I do have the following constraints for your solution:

- The same method must work for **all 3 images**
- The sectors must be scrambled **randomly** every time your program is run
- You can implement your scrambling method (and any supporting methods) in Picture.java. You can then do the following at the end of your testing main to test out your starpoint method:

- Create 3 Picture objects (canvases) that are the same size as the 3 pictures I'm providing you.
- Have each canvas object call your scrambling method with the source picture as a parameter (so your scrambled method may potentially have the following form: **scramble(Picture originalPicture))** There are multiple ways to put the provided images into Picture objects. I think the easiest is just calling FileChooser.pickAFile() 3 times and selecting the appropriate files.
- Show all 3 scrambled images. Note that I'm not putting any constraints on the parameter list of the scramble method, as long as the scrambling is done randomly every time the method is called. I will tell you that the solution we did indeed have the form **scramble(Picture originalPicture) inside Picture.java**

We will also provide you with the following 3 hints that were helpful for my solution:

- Revisit the modulus operator (%) on page 21 of your textbook
- Remember what happens when you divide 2 integers that don't divide perfectly, and recall that the result is the floor() of the true quotient
- Fisher-Yates (Feel free to Google it, but please do NOT copy code directly. If you choose to utilize this hint, implement the algorithm yourself)

Again, you are free to implement ANY solution that works! These tips were just helpful for my particular approach, and there are probably a million and one ways to create this method, after all.

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 method in Picture.java that you can call in a new file called **StarPointPSA4.java**. Please name your new star point method **scramble** and add a comment in both Picture.java and StarPointPSA4.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 scramble1(Partner 1) and scramble2(Partner 2) in Picture.java and create 2 files StarPointPSA4Version1.java(Partner 1) and StarPointPSA4Version2.java(Partner 2) to call them.**

How to Turn in Your Homework

1**.** Place your psa4 folder in the home directory of your student account (cs8afxx).

2. Ensure that the psa4 folder contains at least all the following files: **Picture.java, TestCollage.java, and TestFlip.java. If you did the Star point you will also need to turn in StarPointPSA4Version1.java (and StarPointPSA4Version2.java)**

3. Run the command: cse8aturnin psa4

4. Follow the prompts.

5. Verify that your homework was turned in with the command: cse8averify psa4

Refer back to psa0 turnin instructions for additional details.