Name:
Date:

08.07 Picture Lab Worksheet

Directions: Make note of your responses to the following questions as you work through the activities and exercise in the lesson.

Activity 5 Questions

Question	Yes	No
1. Is the method `getPixels2D` in the `Picture.java` class?		No
2. Is the method `getPixels2D` in the `SimplePicture.java` class?		No
3. Will the following code compile? `DigitalPicture p = new DigitalPicture();`		No
4. Assuming a no-argument constructor exists for `SimplePicture`, will the following code compile? `DigitalPicture p = new SimplePicture();`	Yes	
5. Assuming a no-argument constructor exists for `Picture`, will the following code compile? `DigitalPicture p = new Picture();`	Yes	
6. Assuming a no-argument constructor exists for `Picture`, will the following code compile? `SimplePicture p = new Picture();`	Yes	
7. Assuming a no-argument constructor exists for `SimplePicture`, will the following code compile? `Picture p = new SimplePicture();`		No

Activity 5 Exercise Results

1. Describe your method for `keepOnly` red, blue, or green.

It looks at the image and uses the explore class to isolate one of the RGB values such that the only value that the image displays is red, green or blue. The .explore method lets the image show up, then the colors are saturated as required, and then the image is shown again with the new edits.

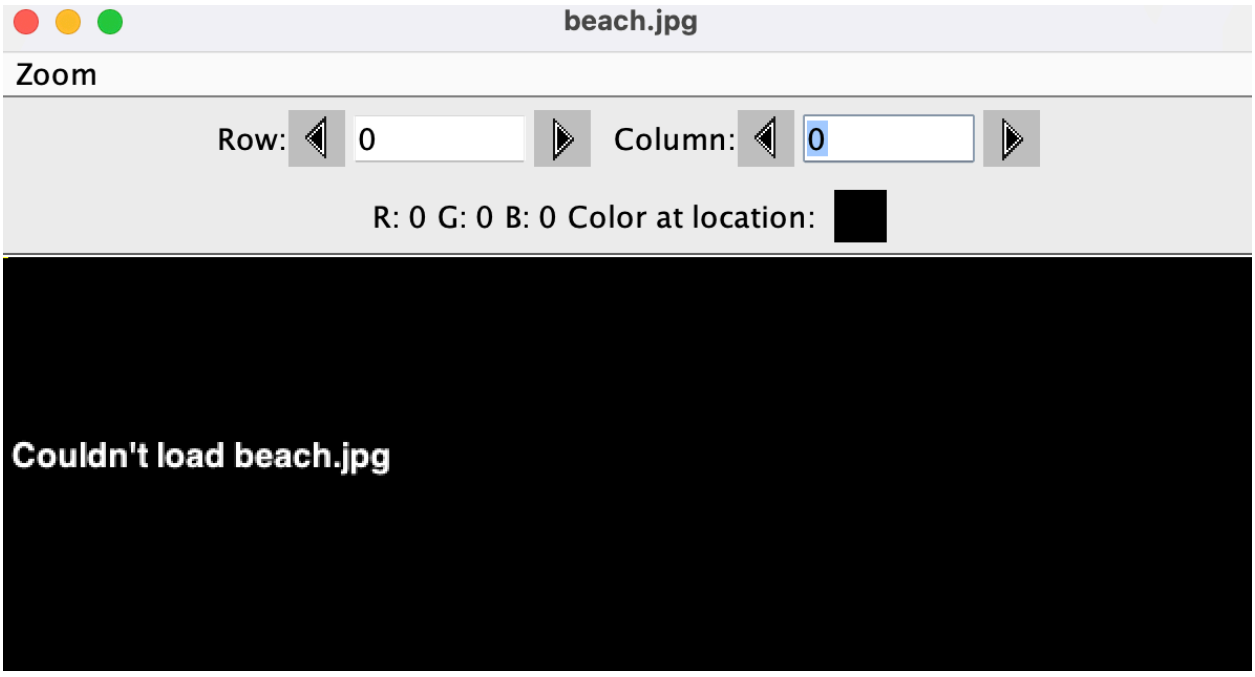
2.

3. For the `negate` method, paste your code related to calculating and setting the values for red, blue, and green.

```
public void negate() {
 Pixel[][] pixels = this.getPixels2D();
 for (Pixel[] rowArray : pixels) {
   for (Pixel pixelObj : rowArray) {
     pixelObj.setRed(pixelObj.getRed() - 255);
     pixelObj.setGreen(pixelObj.getGreen() - 255);
     pixelObj.setBlue(pixelObj.getBlue() - 255);
   }
 }
}
```

14. Paste a copy of the image that is the result of calling the `grayscale`.

my editor is not able to load these images due to the fact that this is meant to be in BlueJ and I am not using BlueJ, so here is what I end up with. I can confirm that it does work, however, without error.



15.

16. For the method `fixUnderwater`, describe the algorithm you'd propose to accomplish the task.
17. Using a double array of pixels, I'd set up the rgb to make the photo look like it was taken underwater. Here is the code.

```
public void fixUnderwater() {
  Pixel[][] pixels = this.getPixels2D();

  int redAverage = 0;
  int greenAverage = 0;
  int blueAverage = 0;
  int totalPixels = 0;

  int maxRed = 0;
  int minRed = 255;
  int maxGreen = 0;
  int minGreen = 255;
  int maxBlue = 0;
  int minBlue = 255;

  for (int row = 26; row < 36; row++) {
      for (int col = 178; col < 198; col++) {
          totalPixels++;

          Pixel myPixel = pixels[row][col];

          redAverage += myPixel.getRed();
          greenAverage += myPixel.getGreen();
          blueAverage += myPixel.getBlue();

          if (myPixel.getRed() > maxRed) { maxRed = myPixel.getRed(); }
          if (myPixel.getRed() < minRed) { minRed = myPixel.getRed(); }
          if (myPixel.getGreen() > maxGreen) { maxGreen = myPixel.getGreen(); }
          if (myPixel.getGreen() < minGreen) { minGreen = myPixel.getGreen(); }
          if (myPixel.getBlue() > maxBlue) { maxBlue = myPixel.getBlue(); }
          if (myPixel.getGreen() < minBlue) { minBlue = myPixel.getBlue(); }
```

```
    }
  }

  redAverage = redAverage / totalPixels;
  greenAverage = greenAverage / totalPixels;
  blueAverage = blueAverage / totalPixels;

  Color averageColor = new Color(redAverage, greenAverage, blueAverage);

  // calculates the range
  int redRange = (maxRed - minRed);
  int greenRange = (maxGreen - minGreen);
  int blueRange = (maxBlue - minBlue);

  int redDistance = redRange;
  int greenDistance = greenRange;
  int blueDistance = blueRange;

  double maxDistance = Math.sqrt(redDistance * redDistance +
                                 greenDistance * greenDistance +
                                 blueDistance * blueDistance);

  double tolerance = 1.7; // higher tolerance means more pixels will be identified as "fish"

  // changes the image based on calculated distance from sample value
  for (int row = 0; row < pixels.length; row++){
    for (int col = 0; col < pixels[0].length; col++){
        Pixel myPixel = pixels[row][col]; //

        boolean closeEnough = myPixel.colorDistance(averageColor) < maxDistance * tolerance; // stopped here,
define this***
        if (closeEnough){
            myPixel.setBlue(myPixel.getBlue() + 50);
        }
        else {
            myPixel.setBlue(myPixel.getBlue() - 50);
        }
```

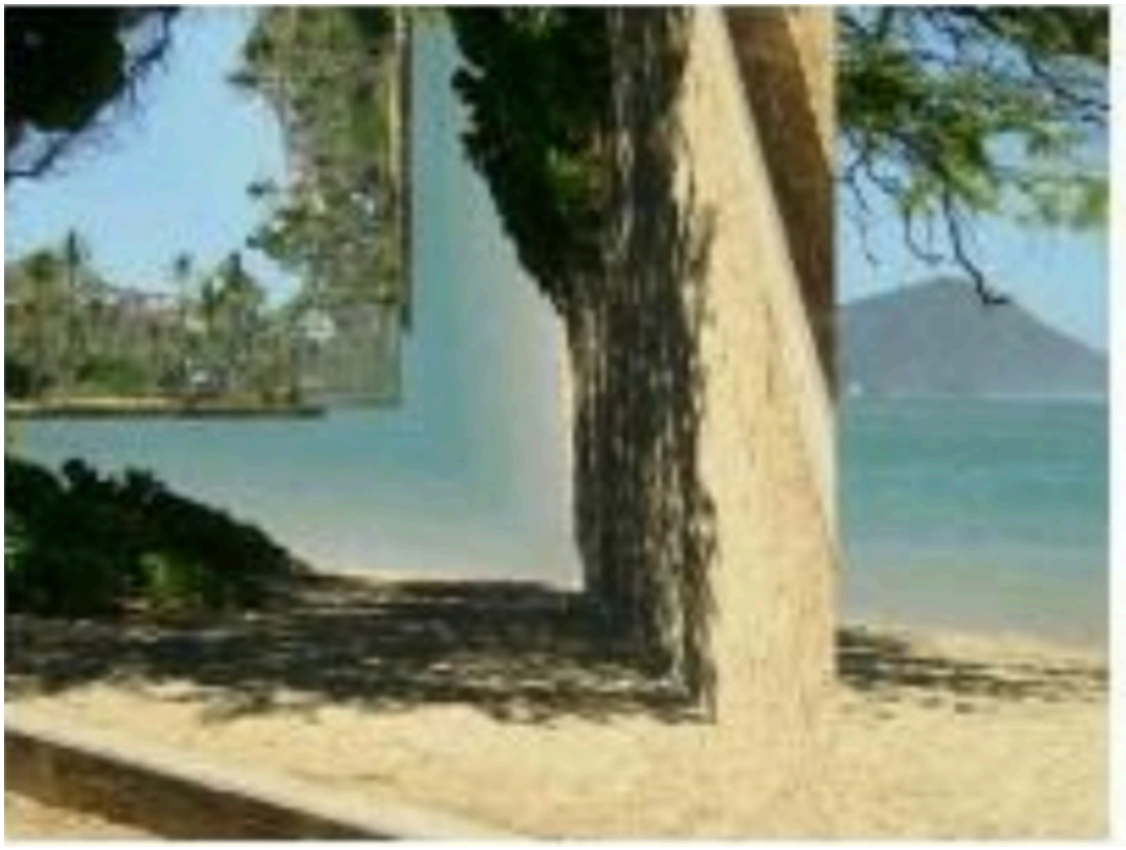
```
    }
  }
}
```

Activity 6 Exercise Results

1. Paste the image that is the result of calling the method `mirrorVerticalRightToLeft`.

2. Describe the algorithm for the method `mirrorHorizontal` works.
creates a double array called Pixels, makes top and Bottom pixels null, makes the height = pixels.length, uses a for loop looping through rows to set color for each row and column in the double array so that they mirror the color for the mirrored image.
3. Paste the image that is the result of calling the method `mirrorHorizontalBotToTop`.

Activity 7 Questions



1. How many times would the body of this nested `for` loop execute? _____

```
for(int row = 7; row < 17; row++)
    for(int col = 6; col < 15; col++)
  10 * 9 = this loop executes 90 times.
```

2. How many times would the body of this nested `for` loop execute? _____

```
for(int row = 5; row <= 11; row++)
    for(int col = 3; col <= 18; col++)
6 * 15, this loop executes 90 times as well.
```

Activity 7 Exercise Results

1. What value is displayed for `count` after the nested loop ends in the `mirrorTemple` method? ____
   18410
2. Paste the image that is the result of calling the method `mirrorArms`.
   I'm sorry, I cannot paste those images, as they do not show up on my editor. I am unable to make that work.
3. Paste the image that is the result of calling the method `mirrorGull`.
   I'm sorry, I cannot paste those images, as they do not show up on my editor. I am unable to make that work.