ITSC 1212 Media Programming Lab 8

You can use any lab machine or personal machine that has been setup with Java and DrJava.

OVERVIEW

1) Part A: Decrease the blue in a picture with parameter
2) Part B: Modify green for a portion of the picture
3) Part C: Modify red for a subset of the picture
4) Bonus 1: Report on RGB average, min, max
5) Bonus 2: Random asterisk generator
6) Bonus 3: Print out the alphabet, vowels as caps
7) Bonus 4: Invert a picture

Part A: Decrease blue, with parameter

1) Create a new method in the Picture.java class, called `decreaseBlue(…)`, that decreases the blue in every pixel of an image by an amount specified through a parameter. The parameter should be in the form of a percentage (passing in 0.4 would decrease the blue value by 40%, passing in 0.25 would decrease the blue by only 25%, etc.). For example, if a given pixel has a blue value of 80, after calling `decreaseBlue(0.4)`, that pixel should have a blue value of 48 (Not 32! What's the formula for decreasing N by X%? Hint: It's NOT to multiply N by X!). You must use an array of Pixels and a loop. Test your method in the Interactions Pane by typing the following commands:

```
> Picture myPic = new Picture(FileChooser.pickAFile());
> myPic.show()
> myPic.decreaseBlue(0.4)
> myPic.repaint()
```

Note: Any time you see the `FileChooser` method you'll need to choose an image file. If we want you to use a specific one we'll tell you. Otherwise, pick any one you like. Most of the image files you'll need will be in the `mediasources` folder.

What are the conditions for your `decreaseBlue(…)` method?
pre-conditions: _____
post-conditions: _____
return value: _____

2) Add two **`if`** statements at the beginning of the method to check that the parameter passed in meets the pre-conditions. One **`if`** statement should test whether the parameter passed in is less than or equal to 0.0 and the other **`if`** statement should test whether the parameter passed in is greater than 1.0. If either of these conditions is true, the method should print out an error message (see below for the exact wording) and exit (`return;`) out of the method without doing anything else. Compile and then test this by trying the following in the Interactions Pane:

```
> Picture myPic = new Picture(FileChooser.pickAFile());
> myPic.show();
> myPic.decreaseBlue(1.3);
Error! Parameter greater than one.
```

Then test the other precondition:

```
> Picture myPic = new Picture(FileChooser.pickAFile());
> myPic.show();
> myPic.decreaseBlue(-0.5);
Error! Parameter less than or equal to zero.
```

Make sure that in addition to printing an error message that the method is returning without executing the rest of the code in the method you wrote. Explore your picture before and after to make sure that nothing is changing (when the preconditions are not met). When you are sure that your `decreaseBlue(…)` method is working properly, you are ready to move to Part B. For this part you just need to submit the updated Picture.java file.

Part B: Modify green for part of the picture

1) Create a new method in Picture.java called `modifyGreen(…)` that changes the green in a **portion** of an image by a specified percentage. This time, the green can increase or decrease, so the appropriate parameter should be limited to values greater than 0.0 but less than or equal to 2.0. Values less than 1 would decrease the amount of green to that percentage (not by that percentage). For example, 0.3 should decrease the green so it's only 30% of the original value (not by 30%, to 30% - this is different from Part A.). Values greater than 1 would increase the amount of green. For example, a value of 1.2 should increase the green to 120% of its original value.

   The percentage parameter should come first, but you also need a second parameter. You want this method to work only on a specified portion of the picture. The technique in the textbook has the picture stored as a one-dimensional array of pixels. What if you wanted to change the green value in only the top half of the picture? What about if you wanted to change only the top third of the picture, or the top tenth of the picture? Think about using a **`while`** loop to accomplish this then fill in the table below to determine what the **`while`** loop condition should be for each of these examples.

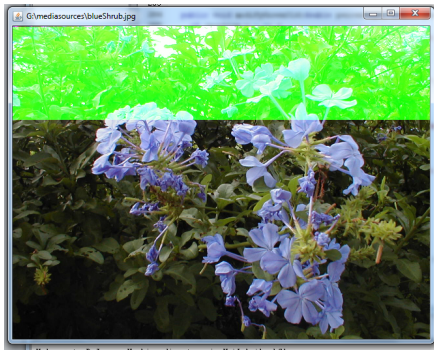
Part of image to change	`while` loop condition	Percentage of array affected
The whole image	`index < pixelArray.length`	100%
The top half	`index < (int) pixelArray.length*0.5`	
The top third		
The top tenth		
The top ¾		

We highly recommend using your sketchbooks to draw a picture of a small (10-15 element) one-dimensional array. Number the elements of the array. See how the percentage you pass in (make it something easy like 10% or 50%) determines which array index values need to be accessed to change those array elements.

Now write a **`while`** loop that can handle the portion being passed in to your method as a parameter. The portion parameter should be a number greater than 0.0 up to and including 1.0 and should be used to vary how much of the image has the green component changed. For now we're only concerned with changing the pixels in the top X% of the picture. Be sure to name

your parameter variables something meaningful! Compile and test your program the same way you tested your **`decreaseBlue(…)`** method.

If you increased the green in the top third of **`blueShrub.jpg`**, your modified picture should look something like this:



3) Add some `if` statements at the beginning of your `modifyGreen(…)` method that test the preconditions. These `if` statements should display an error message then cause the method to exit without making any changes if the parameters are not in the correct range. The following shows what to test for:

```
> Picture myPic = new Picture(FileChooser.pickAFile());
> myPic.modifyGreen(2.0, 1.4);
Error! Portion greater than one.
> myPic.modifyGreen(2.0, -0.5);
Error! Portion less than zero.
> myPic.modifyGreen(9.0, 0.4);
Error! Parameter greater than two.
> myPic.modifyGreen(-35.0, 0.4);
Error! Parameter less than zero.
```

Once you are satisfied that your `modifyGreen(…)` method is working properly, you are ready to move to Part C. For this part you just need to submit the updated Picture.java file.

Part C: Modify red for a subset of the picture

1) I'm not all that fond of my official headshot, but I think we can make it better by giving me laser eyes:

To do this, you need to change part of the picture not from the beginning of the pixel array to an end point, but from a specific starting point to the end point. To get the red laser eyes, I had to fiddle around to find the correct percentages so the increased red started above my

eyes and ended below my eyes. In the example above, I doubled the red from 28% of the pixel array up to 33% of the pixel array, in order to get the result the way I wanted it (the image you see is cropped so it will fit on the page but 34% to 38% will work on that image).

2) Write a `modifyRed(…)` method in Picture.java that will modify a horizontal part of the image (like the example above) by changing the red values of the selected pixels between the starting and ending pixels by a percentage from zero to two. Hint: this is very similar to the `modifyGreen(…)` method except you will need to change the loop's starting point. You'll also need to pay special attention to casting, since the parameters are doubles, but a loop index is always an int. Use your sketchbook!

3) Once your `modifyRed(…)` method is working properly, add tests to the beginning of the method to check the following:
   - The red modification parameter is between 0.0 and 2.0 inclusive
   - The beginning portion for red change is between 0.0 and 1.0 inclusive
   - The same for the end portion for the red change
   - The beginning portion of red changes is less than the ending (for example it wouldn't make sense to change the red in the image starting at the 20% point and ending at the 10% point, but it would make sense to change it from the 10% point to the 20% point.

   Here's an example of the errors your method should detect and how it should respond:

```
> Picture myPic = new Picture(FileChooser.pickAFile());
> myPic.modifyRed(0.4, 0.46, 1.7);
Error! Second portion greater than one.
> myPic.modifyRed(0.4, -0.6, 0.5);
Error! First portion less than zero.
> myPic.modifyRed(0.4, 0.7, 0.5);
Error! First portion comes after second portion.
> myPic.modifyRed(3.4, 0.4, 0.6);
Error! Parameter greater than two.
```

   When your tests are working, you are done with required parts of this lab. For this part you just need to submit the updated Picture.java file.

ITSC 1212 Media Programming Lab 8

Bonus 1: RGB Report

1) Write a method to report the average, min, and max for each of the red, green and blue components of an image. The method should be called `reportRGB()`, and take no parameters. It should print out the information to the console as in the following screenshot.

```
> Picture pic = new Picture(FileChooser.pickAFile());
> pic.reportRGB();
Average red value is 138
Average green value is 98
Average blue value is 107
Minimum red is 12
Minimum green is 2
Minimum blue is 0
Maximum red value is 247
Maximum green value is 210
Maximum blue value is 237
>
```

Use your sketchbook to plan how you're going to write your code. Perhaps instead of writing all the code at once you could write code to remember the largest red value you find after looking at each pixel and print that out. Then add the code to find the smallest. Then add the code to find the average. Once all those are working it should be really easy to add code for the green and blue.

If you use your own image you'll get different values than you see above. If you want to test your method and see if you get the same values as above, use the **Celine_Warholized.jpg** image from Canvas.

Bonus 2: Random asterisk generator

1) Download **Lab8PartE.java** and save it to the usual directory. This program is an empty shell that does nothing. Use the `main` method to write code that will generate a random number between 0 and 20, and then print out that many asterisks to the console. This should take about 7 lines of code inside the `main(…)` method. That's just a guide - don't worry if your solution takes more or less lines of code. Getting the solution right is your primary goal. In order to generate a random number, you will want to use the `Math.random()` function, which returns a random value that might be equal to 0.0 but will always be just less than (but never equal to) 1.0. Note that this value is a `double`, not an integer. Multiply this value by the largest number in the range you want plus 1 (in this case, multiply by 21), and then cast the result back to an `int`. That will give you a random integer between 0 and 20. Your output should look like this (but the number of asterisks you get may be different).

```
> run Lab8PartE
**********
```

2) Now, add two tests to your code to find out whether the random number generated is odd or even. The modulo (%) operator will be useful for this. Print out on the line after the asterisks how many asterisks were printed, and whether that number is odd or even. Your program output should look like this:

```
> run Lab8PartE
**********
13 asterisks, which is an odd number.
```

Bonus 3: Print out the alphabet, with vowels as caps

1) Download **Lab8PartF.java** and save it to the usual directory. This program is an empty shell that does nothing. Inside the main method write code that will print out the letters of the alphabet (lowercase letters only, not capital letters). To do this, you need to know about ASCII (pronounced ass′-key), the **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange. Computers only understand numbers (zeroes and ones to be precise) so a numeric code is needed that translates the internal numeric values that computers use into characters like a capital A or a comma that we can use for displays or printouts. Modern computers like ours use ASCII. In ASCII, the code for the letter 'a' is 97, and for 'A' is 65. For 'b' it's 98, and for 'B' it's 66, etc. To print out the letter 'b' using the ASCII code, you would do the following:

```
> char letter = 98;
> System.out.println(letter);
b
```

Remember the char data type? It's always a single character (whereas a String could be a group of characters). This allows you to print out a letter, based on knowing its ASCII code (you can look up ASCII codes on line, just search for ASCII code table and you'll get lots of images that show you the translation). Your program should print out the 26 lowercase letters of the alphabet starting with 'a'. Your output should look like this:

```
> run Lab8PartF
abcdefghijklmnopqrstuvwxyz
>
```

Remember that when you have data of one type you can use casting to change it to a different type. Also remember that you have `println` and `print` statements you can use.

2) Now, add some tests inside your while loop to test whether you are hitting the vowels. Figure out how to use the ASCII codes for the uppercase versions of the vowels and print them out instead of the lowercase versions. Note that this can be done by writing five separate if statements, but it can also be done by writing one if statement, with a condition of the form (test1 || test2 || test3 || test4 || test5), where || means 'or'.

```
> run Lab8PartF
AbcdEfghIjklmnOpqrstUvwxyz
>
```

Once you are satisfied that your program is working show one of the professors or the TA.

Bonus 4: Invert a picture

1) Write a method in Picture.java called `invert()` that reverses the array of pixels in the image and returns the inverted Picture. If you do this right, you will get the following result:

2) In order to do this bonus, you need to create a separate area of memory in which to store a temporary copy of the pixels. The code below will be needed at the top of your `invert()` method:

```
// get pixelArray as usual
Pixel[] pixelArray = this.getPixels();
// create a new Picture of the same size (this allocates
// the appropriate amount of memory to use for temp storage)
Picture newPic = new Picture(this.getWidth(), this.getHeight());
// get a pixelArray for the new picture. Now you can copy pixel
// values between the two arrays
Pixel[] tempArray = newPic.getPixels();
```

3) Now you've got **tempArray** that points to the empty picture that's the same size as the picture you want to invert. All you have to do now is move the pixels from one picture to the other and place them in the proper place of course!

4) Here is another hint. This code does the job of reversing an array of 10 random integers:

```
// declare original array and a temp array for copying
int[] myIntArray = new int[10];
int[] myTempArray = new int[10];
int index = 0;
// create and print out an array of 10 random integers
while (index < myIntArray.length) {
  myIntArray[index] = (int)(Math.random()*100);
  System.out.print(myIntArray[index] + ", ");
  // copy myIntArray values into temp array
```

```
  myTempArray[index] = myIntArray[index];
  index++;
}
System.out.println();

// now copy from temp array back into intArray,
// but in reverse order
index = 0;
while (index < myIntArray.length) {
  myIntArray[index] = myTempArray[myIntArray.length - 1 - index];
  System.out.print(myIntArray[index] + ", ");
  index++;
}
```

5) We strongly suggest that you use your sketchbook to draw a picture of two small arrays (5x5 elements maybe). Show the indexes for each element then draw what happens when the first element of the source array goes to the last element of the destination array, then when the 2nd element of the source array goes to the next-to-the-last element of the destination array. Can you develop a formula that converts the index of the source array into the index where that element should go in the destination array?

Lab 8 Canvas submission

1. Navigate to the course Canvas page.
2. Click on Module 8
3. In Module 8 locate and click on the Lab 8 assignment and submit the following files
    a. Picture.java
4. If you completed the bonus submit your files and include a comment with your submission indicating which bonuses you completed.

You're done!

If you are using the lab computer, remember to logout.

So what did you learn in this lab?

1) How to create an array of Pixels
2) How to use a loop to access each element of an array
3) How to modify the color of an image
4) How to use parameters to control the operation of a loop
5) How to loop through an array keeping track of various values and performing computations
6) How to use a random number generator
7) How to use various print statements to control how console output looks
8) How to use the modulus operator
9) How to use the char datatype and cast to it from an integer
10) How to invert the elements of an array