Picture lab

These pages will be used to check off programs and submit your answers to questions. It saves us from printing off the entire 28 page packet. You must read the Student Guide to understand the assignment.

**Activity 1: Introduction to digital pictures and color**

1. How many bits does it take to represent the values from 0 to 255?

2. How many bytes does it take to represent a color in the RBG color model?

3. How many pixels are in a picture that is 640 pixels wide and 480 pixels high?

**Activity 2: Picking a color**

1. How can you make pink?

2. How can you make yellow?

3. How can you make purple?

4. How can you make white?

5. How can you make dark gray?

**A3: Exploring a picture**

1. What is the row index for the top left corner of the picture?

2. What is the column index for the top left corner of the picture?

3. The width of this picture is 640. What is the right most column index?

4. The height of this picture is 480. What is the bottom most row index?

5. Does the row index increase from left to right or top to bottom?

6. Does the column index increase from left to right or top to bottom?

7. Set the zoom to 500%. Can you see squares of color? This is called *pixelation*. Pixelation means displaying a picture so magnified that the individual pixels look like small squares.

**Exercises**

|  |  |
| --- | --- |
|  | 1. Modify the main method in the PictureExplorer class to create and explore a different picture from the images folder. |
|  | 2. Add a picture to the images folder and then create and explore that picture in the main method. If the picture is very large (for instance, one from a digital camera), you can scale it using the scale method in the Picture class. For example, you can make a new picture (“smallMyPicture.jpg” in the images folder) one-fourth the size of the original (“myPicture.jpg”) using:  Picture p = new Picture("myPicture.jpg");  Picture smallP = p.scale(0.25,0.25);  smallP.write("smallMyPicture.jpg"); |

**A4: Two-dimensional arrays in Java**

**Exercises**

|  |  |
| --- | --- |
|  | 1. Write a getCount method in the IntArrayWorker class that returns the count of the number of times a passed integer value is found in the matrix. There is already a method to test this in IntArrayWorkerTester. Just uncomment the method testGetCount() and the call to it in the main method of IntArrayWorkerTester. |
|  | 2. Write a getLargest method in the IntArrayWorker class that returns the largest value in the matrix. There is already a method to test this in IntArrayWorkerTester. Just uncomment the method testGetLargest() and the call to it in the main method of  IntArrayWorkerTester. |
|  | 3. Write a getColTotal method in the IntArrayWorker class that returns the total of all integers in a specified column. There is already a method to test this in  IntArrayWorkerTester. Just uncomment the method testGetColTotal() and the call to it in the main method of IntArrayWorkerTester. |

**A5: Modifying a picture**

1. Open Picture.java and look for the method getPixels2D. Is it there?

2. Open SimplePicture.java and look for the method getPixels2D. Is it there?

3. Does the following code compile?

DigitalPicture p = new DigitalPicture();

4. Assuming that a no-argument constructor exists for SimplePicture, would the following code compile?

DigitalPicture p = new SimplePicture();

5. Assuming that a no-argument constructor exists for Picture, does the following code compile?

DigitalPicture p = new Picture();

6. Assuming that a no-argument constructor exists for Picture, does the following code compile?

SimplePicture p = new Picture();

7. Assuming that a no-argument constructor exists for SimplePicture, does the following code compile?

Picture p = new SimplePicture();

**Exercises**

|  |  |
| --- | --- |
|  | 1. Open PictureTester.java and run its main method. You should get the same results as running the main method in the Picture class. The PictureTester class contains class (static) methods for testing the methods that are in the Picture class. |
|  | 2. Uncomment the appropriate test method in the main method of PictureTester to test any of the other methods in Picture.java. You can comment out the tests you don’t want to run. You can also add new test methods to PictureTester to test any methods you create in the Picture class. |
|  | 3. Using the zeroBlue method as a starting point, write the method keepOnlyBlue that will keep only the blue values, that is, it will set the red and green values to zero. Create a class (static) method to test this new method in the class PictureTester. Be sure to call the new test method in the main method in PictureTester. |
|  | 4. Write the negate method to negate all the pixels in a picture. To negate a picture, set the red value to 255 minus the current red value, the green value to 255 minus the current green value and the blue value to 255 minus the current blue value. Create a class (static) method to test this new method in the class PictureTester. Be sure to call the new test method in the main method in PictureTester. |
|  | 5. Write the grayscale method to turn the picture into shades of gray. Set the red, green, and blue values to the average of the current red, green, and blue values (add all three values and divide by 3). Create a class (static) method to test this new method in the class PictureTester. Be sure to call the new test method in the main method in PictureTester. |
|  | 6. **Challenge** — Explore the “water.jpg” picture in the images folder. Write a method fixUnderwater() to modify the pixel colors to make the fish easier to see. Create a class (static) method to test this new method in the class PictureTester. Be sure to call the new test method in the main method in PictureTester. |

**A6: Mirroring pictures**

**Exercises**

|  |  |
| --- | --- |
|  | 1. Write the method mirrorVerticalRightToLeft that mirrors a picture around a mirror placed vertically from right to left. Hint: you can copy the body of mirrorVertical and only change one line in the body of the method to accomplish this. Write a class (static) test method called testMirrorVerticalRightToLeft in PictureTester to test this new method and call it in the main method. |
|  | 2. Write the method mirrorHorizontal that mirrors a picture around a mirror placed horizontally at the middle of the height of the picture. Mirror from top to bottom as shown in the pictures below (Figure 8). Write a class (static) test method in PictureTester to test this new method and call it in the main method. |
|  | 3. Write the method mirrorHorizontalBotToTop that mirrors the picture around a mirror placed horizontally from bottom to top. Hint: you can copy the body of mirrorHorizontal and only change one line to accomplish this. Write a class (static) test method in PictureTester to test this new method and call it in the main method. |
|  | 4. **Challenge** — Work in groups to figure out the algorithm for the method mirrorDiagonal that mirrors just a square part of the picture from bottom left to top right around a mirror placed on the diagonal line (the diagonal line is the one where the row index equals the column index). This will copy the triangular area to the left and below the diagonal line as shown below. This is like folding a square piece of paper from the bottom left to the top right, painting just the bottom left triangle and then (while the paint is still wet) folding the paper up to the top right again. The paint would be copied from the bottom left to the top right as shown in the pictures below (Figure 9). Write a class (static) test method in PictureTester to test this new method and call it in the main method. |

**A7: Mirroring part of a picture**

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++)

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++)

**Exercises**

|  |  |
| --- | --- |
|  | 1. Check the calculation of the number of times the body of the nested loop executes by adding an integer count variable to the mirrorTemple method that starts out at 0 and increments inside the body of the loop. Print the value of count after the nested loop ends. |
|  | 2. Write the method mirrorArms to mirror the arms on the snowman (“snowman.jpg”) to make a snowman with 4 arms. Write a class (static) test method in PictureTester to test this new method and call it in the main method. |
|  | 3. Write the method mirrorGull to mirror the seagull (“seagull.jpg”) to the right so that there are two seagulls on the beach near each other. Write a class (static) test method in PictureTester to test this new method and call it in the main method. |

**A8: Creating a collage**

**Exercises**

|  |  |
| --- | --- |
|  | 1. Create a second copy method that adds parameters to allow you to copy just part of the fromPic. You will need to add parameters that specify the start row, end row, start column, and end column to copy from. Write a class (static) test method in PictureTester to test this new method and call it in the main method. |
|  | 2. Create a myCollage method that has at least three pictures (can be the same picture) copied three times with three different picture manipulations and at least one mirroring. Write a class (static) test method in PictureTester to test this new method and call it in the main method. |

**A9: Simple edge detection (optional)**

**Exercises**

|  |  |
| --- | --- |
|  | 1. Notice that the current edge detection method works best when there are big color changes from left to right but not when the changes are from top to bottom. Add another loop that compares the current pixel with the one below and sets the current pixel color to black as well when the color distance is greater than the specified edge distance. |
|  | 2. Work in groups to come up with another algorithm for edge detection. |