Using the information you have gained so far, modify the code. Click “Submit Assignment” in the upper right corner of the screen to submit your work. Be sure and save the files as YourNameMod13PictureLabAssignmentSix

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.
   1. /\*\* Method to set values other than blue to 0 \*/
   2. public void keepOnlyBlue() {
   3. Pixel[][] pixels = this.getPixels2D();
   4. for (int i = 0; i < pixels.length; i++) {
   5. for (int j = 0; j < pixels[i].length; j++) {
   6. pixels[i][j].setRed(0);
   7. pixels[i][j].setGreen(0);
   8. }
   9. }
   10. }
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.
   1. // negative image
   2. public void negateImage() {
   3. Pixel[][] pixels = this.getPixels2D();
   4. for (int i = 0; i < pixels.length; i++) {
   5. for (int j = 0; j < pixels[i].length; j++) {
   6. Pixel currentPixel = pixels[i][j];
   7. int currentRed = currentPixel.getRed();
   8. int currentGreen = currentPixel.getGreen();
   9. int currentBlue = currentPixel.getBlue();
   10. currentPixel.setRed(255 - currentRed);
   11. currentPixel.setGreen(255 - currentGreen);
   12. currentPixel.setBlue(255 - currentBlue);
   13. }
   14. }
   15. }
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.
   1. // grayscale the image
   2. public void greyscaleImage() {
   3. Pixel[][] pixels = this.getPixels2D();
   4. for (int i = 0; i < pixels.length; i++) {
   5. for (int j = 0; j < pixels[i].length; j++) {
   6. Pixel currentPixel = pixels[i][j];
   7. int currentRed = currentPixel.getRed();
   8. int currentGreen = currentPixel.getGreen();
   9. int currentBlue = currentPixel.getBlue();
   10. int average = (currentRed + currentGreen + currentBlue) / 3;
   11. currentPixel.setRed(average);
   12. currentPixel.setGreen(average);
   13. currentPixel.setBlue(average);
   14. }
   15. }
   16. }
6. Extra Credit — 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.
   1. // blacks out pixels that seem like they are quite blue (not fish)
   2. public void fixUnderwater() {
   3. Pixel[][] pixels = this.getPixels2D();
   4. for (Pixel[] row : pixels)
   5. {
   6. for (Pixel currentPixel : row)
   7. {
   8. // these values seem to work best?
   9. int r = Math.abs(currentPixel.getRed() - 22);
   10. int g = Math.abs(currentPixel.getGreen() - 161);
   11. int b = Math.abs(currentPixel.getBlue() - 176);
   12. int diff = r + g + b;
   13. if (diff > 20) {
   14. currentPixel.setRed(0);
   15. currentPixel.setGreen(0);
   16. currentPixel.setBlue(0);
   17. }
   18. }
   19. }
   20. }