First Class: Colors and Pixels with Class Composition

CS 5004 Object Oriented Design

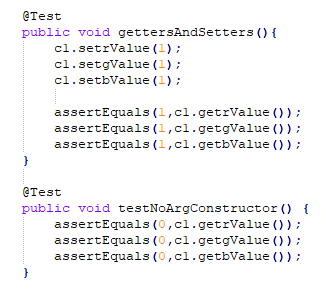
1. **Goals**

* Create a class from scratch
* Create a driver to test that class
* Create multiple types of constructors
* Create getters and setters for private variables
* Demonstrate data encapsulation with a setter test
* Reverse engineer an existing JUnit test file

ICE:

Your TA will describe this week’s assignment and then divide you up into groups. In your groups do not share code, but do

* Review the assignment itself
* Review the test file and briefly describe what each test is testing and what the test implies
* Discuss how to build a frame for this assignment
* Describe how composition will be used
* Build a frame by creating the files you are going to need in your IDE and loading the testing file
* Help each other pass the first two point tests:



If you finish all of this, go ahead and break off and feel free to get started with the reminder of the lab assignment.

Do not share code past this point or look at each other’s screen’s.

Submit the names of each person in your group, the progress you made, and screenshots of passing the first two tests with your completed frame for the ICE.

1. **Instructions:**

Testing File

For this assignment, I’ll give you one of the testing files instead of asking you to create it. A part of your grade will be making sure you pass all of the tests in this file. Feel free to reverse engineer these tests to make sure you get it right, but don’t change any of the tests. When you add tests, put them in their own section and make sure to add comments clearly identifying the tests you added.

<https://www.dropbox.com/scl/fi/4fdb33dj8afa07p5def4w/ColorTests.java?rlkey=6bukjzy886sytatducp67mt6s&dl=0>

Part 1: Color Class

We will get started by creating a class to represent a 255 RBG color. In most applications that deal with color we define that color using three separate values: a red, green, and blue value. These values are used by the computer to create a single pixel for display on the screen. An RGB value of 0,0,0 means no color and thus black. An RGB value of 255,255,255 represents maximum saturation leading to white.

Here are the requirements for this class.

1. Create a class called Color with an int rValue, gValue, and bValue
2. Make sure all values are private and include getters and setters for access.
3. Make sure the user can not set a value greater than 255 or less than 0. If they try, just set the value to 0. Make sure this can’t be circumvented. It’s always better to avoid having to change border values like this in multiple places so create a checker helper function, but don’t allow outside access to it.
4. Create a no-argument and an all-argument constructor.
5. Create a printColor method that will print the RGB values separated by commas
6. Create testers to check to see if a color is white or if it is black. *(0,0,0 is black and 255,255,255 is white)*
7. Create a combineColor method that will add the r,g,b values of a sent color to the existing color, but if it goes over 255 set it to 0.
8. Add JavaDoc Style comments for the class and for each method. (This is likely the only time I’ll ask for JavaDoc comments.)
9. Make sure you only have to make adjustments in one place if you wanted to change the 0 to 255 rules

Part 2 : Create a Pixel test file

A pixel is a class that has an x and y value and a color. Test driven development means we create our tests first. Use the following objectives to create a pixel test file.

1. a no argument constructor that sets the x and y to 0 and uses the default color constructor
2. a 2 argument constructor that accepts x and y values then uses the default color constructor
3. a color constructor that uses this chaining to set the default x and y values to 0 then sets the color using a sent color
4. an all-argument constructor
5. a printPixel method that prints the x and y coordinates separated by commas and then uses composition to add on the color objects information
6. a moveHor method that increases the x coordinate given a set amount
7. a movVer method that increases the y coordinate a given amount
8. a movMeth that moves the x and y coordinates a given amount (use code reduction)
9. a setBorder that sets a max and min x and y values
10. a checker helper method that makes sure values don’t exceed the border amount
11. Once border values are set, make sure the x and y values can only be less than or equal to the max border and greater than or equal to the min border.

Part 3 : Create the Pixel code to match your tests

Finally, create a pixel test file that will implement all of your part 2 tests. Make sure you have 100% coverage on both files.

**3. Extensions:**

Outside academics you will not get specific requirements. Each lab assignment is worth 100 points, but the base requirements will only get you to 85% - 90%. If you want an A, you’ll have to find a way to go above and beyond what is asked. I’ll often make some suggestions to you in this section, but it is entirely up to you what you’d like to add to the assignment. Make sure you know who your grader is and discuss extension expectations with them. You won’t have to do all of the extensions to get credit.

Extension suggestions:

1. Add additional functionality to the existing requests
   1. test for more colors
   2. test for shades of gray
   3. add more complex color manipulation
2. Go above and beyond on your testing
3. Make improvements to the existing testing file
4. Add more complex pixel manipulation or movement
5. Create a driver and include some driver tests

**4. Report:**

Each assignment must include a short report. The generation of this report should take you no more than 15 minutes. This gives you a chance to reflect back on what you learned and it makes grading easier on your grader. For this report, I want the following sections:

1. Reflection (*What did you learn? How did you feel about working from a test file?)*
2. **Extensions (*What extensions are you requesting?)***
3. Grading Statement (*Based on the rubric, what grade do you feel you deserve? Be honest.)*
4. Academic integrity statement *(found on the landing page)*

**5. Submission:**

Submit your files as a single zip file named: “Your Name”\_”Assignment”.zip

Unless your grader requests it, do not submit your entire project folder. Do not include any JavaDocs, and make sure you remove any package statements.

* Color.java
* ColorTest.java
* Pixel.java
* PixelTest.java

Submission checklist:

* Did you include all files requested?
* Did you include adequate comments?
* Did you include comment blocks at the top of each file?
* Did you name your files as requested?
* Does your code compile?
* Did you remove any package lines generated by your IDE?
* Did you take care of any warnings presented by your IDE?

|  | **Possible** | **Given** |
| --- | --- | --- |
| Color Tests passed after implementation completed | | |
| gettersAndSetters - pass | 5 |  |
| noArgs - pass | 5 |  |
| allArgs - pass | 5 |  |
| borderTest - pass | 5 |  |
| colorTests - pass | 5 |  |
| code reduction done | 5 |  |
| Pixel Tests implemented as requested and pass with implementation | | |
| default constructor tests | 5 |  |
| x,y constructor tests | 5 |  |
| color constructor tests | 5 |  |
| printPixel tested | 5 |  |
| all move methods tested | 5 |  |
| borderSetter checked | 5 |  |
| checking to make sure pixel can’t “escape” border done | 5 |  |
| Pixel Implementation | | |
| implemented as requested | 5 |  |
| code reduction in place | 5 |  |
| Other | | |
| 100% coverage achieved on both files | 10 |  |
| Not included in total possible: | | |
| Does not compile | -100 | 0 |
| Extensions (Not calculated without report) | 15 | 0 |
| No report or missing report sections | -100 | 0 |
| Code Quality(correct indentation, comment blocks, variable naming, etc) | -50 |  |
| Late penalty | -20 | 0 |
| Creative or went above and beyond | 10 | 0 |
| Code contains warnings | -20 | 0 |
|  | |  |
| TOTAL POINTS POSSIBLE out of 100 | 85 | 0 |