Jasque Saydyk and Tyler Smith

Professor Elwakil, Melton, and McCarty

CS 136L Section 3801

7 February 2017

Lab 03 - Painting Cars

1. Problem Statement

The goal of this program is to create a program that creates cars in a JFrame that allows to adjust the car's length, wheels, and color. As for the color, it's supposed to be created by mixing different colors together, like pouring two paints together to get a third different color paint.

Requirements

- CustomCar class based off of the Car example class that allows you to change the car's length, front and back wheels, and color in the constructor
    - No mutators
    - Implement draw method that uses graphics2D object as a parameter
- PaintBucket class that allows you to mix different colors together
    - Accessor for the color
    - Mutator that mixes inputted color with the current color
- A viewer and panel/component class to view the result with

2. Planning

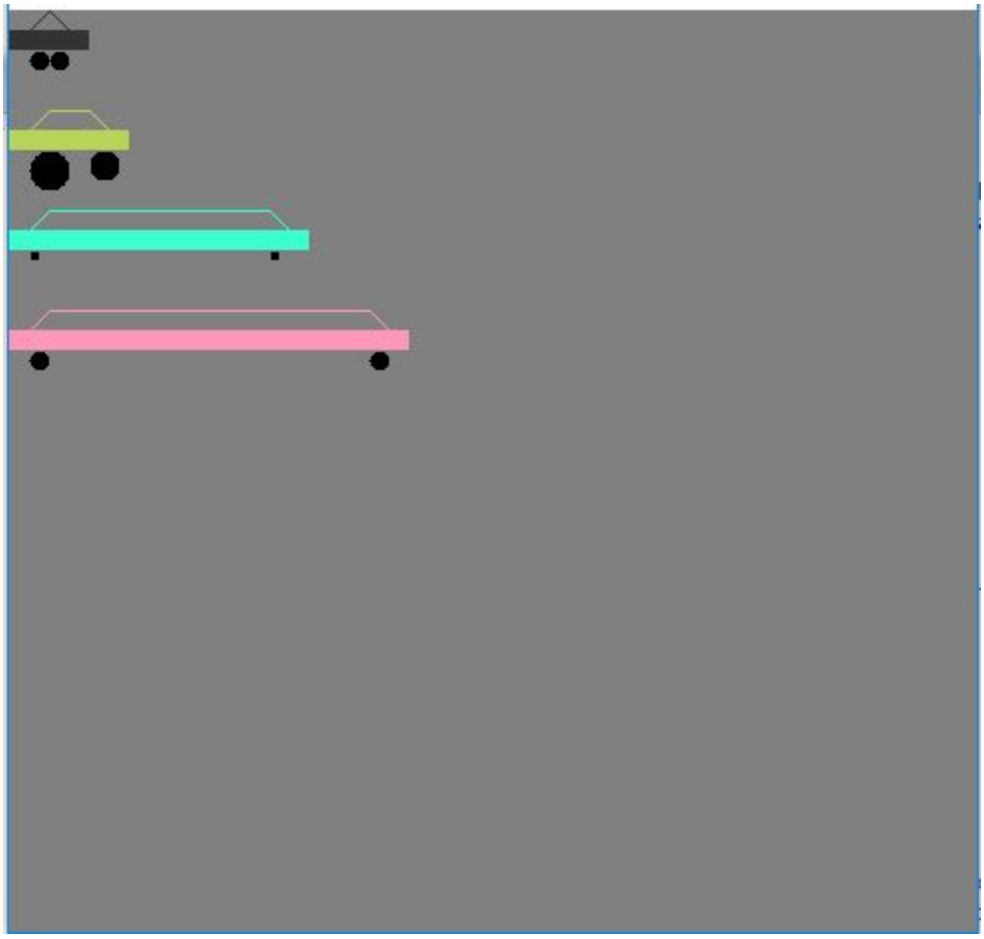
Once you're able to list out the requirements and see the given example code from the book, the plan is pretty straight-forward. Since the book has already implemented the Car class, CustomCar just inherits the class, then adds the needed parts.

The only other nuance is that the PaintBucket class will work with three ints to represent RGB in the class, but output a color class. In addition to that, I must explain how the math works for paint bucket. To put it simply, you're just taking the average of the new and old red, green, and blue numbers, then slapping them into a Color object. Specifically, first you multiply the color by the number of parts it represents, e.g. Old(O) red 200 is one part, and New(N) red 150 is 4 parts, so the result is O red 200 and N red 600. Second, you add the N red and O red together to get red 800. Third, you add the parts together, resulting in five parts. Lastly, you divide the red sum by the total number of parts, so 800 divided by 5, resulting the final result red 160.

This project is pretty straightforward, especially since the example code has done half the work.

3. Implementation and Testing

Implementing this was straight-forward, with not a lot of testing occurring. The only issue we ran into that took a while to solve was realizing that we had to change the X Y variables in the car class to be protected instead of private so that they could be inherited properly.



4. Reflection

In retrospect, I should look at the given code before I start to come up with a plan on how to tackle the program, since I was coming up with solutions to problems that were already solved in that code.

Compared to Lab 02, this project was significantly easier, and thus we don't have a lot to say past this point.