Painting Cars

1 Overview

In this lab we will be designing classes to represent parametrized cars as well as a bucket of paint for mixing colors. We will be creating three different java classes, including a PaintBucket class to let us mix colors, a CustomCar class to let us generate immutable, drawable car objects, and a CustomCarComponent that will be responsible for using the prior two classes in order to draw a scene.

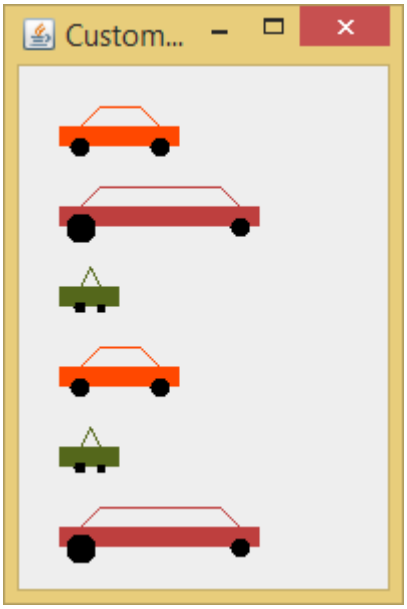

Fig. 1: A set of colored custom cars

2 Learning Outcomes

By the end of this project students should be able to:

- create cohesive java classes;
- store state within objects.
- write programs featuring encapsulated functionality;
- work effectively with a partner using pair-programming;
- write an effective report that describes the students' problem solving process.

3 Pre-Lab Instructions

Do this part and turn in your answer on BBLearn before you come to lab:

This project will require you to write classes and methods.

- Read Big Java chapter 3

- Look up the API for the following classes and list examples of accessors and mutators:
  - Graphics2D
  - Ellipse2D.Double
  - Color

  Write down each of these classes, followed by a labeled list of accessors and a labeled list of mutators and turn in your answers on BBLearn.

4 Lab Instructions

Do this part in lab:
This lab consists of three steps: Making a car class, writing a paint bucket class, and then using them together to create a Swing component.

Step 1

First we will write a class representing a customized car. This will be working off of the example presented in Big Java 3.8 in which a car is drawn using Swing. Our car class will be similar to the one described in the book, but will be modified to allow us to specify the length of the car, the size of the wheel, and the color of the car. Like the Car example in the book, we will feature only a constructor and and a draw method. Notice the lack of mutators for this class. This means our CustomCar objects will be immutable, unable to change after they are created.

Start by deciding what instance variables you will need to represent the state of the object. Include the process of determining your instance variables in the planning section of your report. Next, implement both the constructor and the draw method. The constructor should accept parameters for car length, front and back wheel size, and the color. The draw method should accept a Graphics2D object as a parameter.

To test our CustomCar class we will need a component to view it in. Create a CustomCarComponent class as well, and provide a main method that instantiates a JFrame. In the example screen shot, the first car shown is 60 pixels long with 10 pixel wheels. The second is 100 pixels long with 15 pixel back tires, and the final car is 30 pixels long with wheels 5 and 6 pixels in diameter. Feel free to as creative as you would like in drawing your cars, but make sure they work for a variety of lengths and wheel sizes.

Step 2

First we will write a class representing a paint bucket. Our class will represent a well in which we pour different amounts of color, and can extract the color

mixed together. Real paint uses a subtractive model of color, where additional color tends to darken, while computers use additive color, where colors brighten as more color is added. In our implementation, we will simply average the colors that have been added numerically.

For example, if one gallon of red was mixed with one gallon of green and one gallon of blue it would become gray in our implementation. As a numeric example, if one part was added with RGB values of 100,200,0 was added to 3 parts with RGB values 0, 0, 200, our resulting color would have RGB values 25, 50, 150. Another example would be RGB values 100,100,100 being mixed evenly with RGB values 200, 0, 100 would yield a final color of 150, 50, 100.

You are to write a PaintBucket class with a no argument constructor, one mutator and one accessor. Our mutator will add paint, where the user of the class may add a color at some some quantity. Our accessor will allow us to retrieve the color we have mixed. Your PaintBucketClass should **NOT** contain a setColor method; The color can only be changed by adding color at a quantity. As with the CustomCar class class, start by planning what instance variables you will need and record your reasoning in the planning section of your report. Remember that a single paint bucket can have paint added to it multiple times. Next develop your methods. Finally, write a main method to test your PaintBucket class. Use a text printout to confirm that the above color examples work as intended. We won't see any actual color on the screen, just text for now. In step 3 we will get to see what our colors look like.

Step 3

Finally, Modify your CustomCarComponent to show at least three different cars, each colored differently by manipulating and drawing colors from a single PaintBucket object. Make sure a screen shot of your finished scene is included in your report.

5 Lab Report

Each pair of students will write a single lab report together and each student will turn in that same lab report on BBLearn. Submissions from each student on a pair should be identical. In order to recieve points, you MUST make a BBLearn submission.

Your lab report should begin with a preamble that contains:

- The lab assignment number and all partner names
- Your name(s)
- The date

- The lab section

It should then be followed by four numbered sections:

1. Problem Statement

In this section you should describe the problem in **your** own words. The problem statement should answer questions like:

- What are the important features of the problem?
- What are the problem requirements?

This section should also include a reasonably complete list of requirements in the assignment. Following your description of the problem, include a bulleted list of specific features to implement. If there are any specific funtions, classes or numeric requirements given to you, they should be represented in this bulleted list. It is recomended that you complete this section before you begin coding the problem, as gathering these requirements may help you organize your thoughts before you begin your soulation.

2. Planning

In the second section you should describe what planning you did in order to solve the problem. You should include planning artifacts like sketches, diagrams, or pseudocode you may have used. You should also describe your planning process. List the specific data structures or techniques you plan on using, and why. How do you plan on breaking up the problems into functions, classes and methods?

3. Implementation and Testing

In the third section you should demonstrate your implementation. As directed by the lab instructor you should (as appropriate) include:

- a copy of your source code (Submitted in BBLearn as .java files, should not be in your report document)
- a screen shot of your running application / solution. This should include a screenshot of the output, be it a terminal window or graphical interface
- results from testing. This is specific to the lab, and not every lab will include this element.

4. Reflection

In the last section you should reflect on the project. What part of the projec was the most difficult? Where there other solutions to this problem that you considered? Where there any new solutions you can think of now that you couldn't then? Do you think the way you chose was the best option, or would

you go about the problem differently if you had to start over? How might the problem had been broken up into easier problems? If you had one more day to work this problem, what improvements might you make? Every problem has alternative solutions and solution has tradeoffs or improvements, you are required to identify some of these elements.

5. Partner Rating

Every assignment you are required to rate your partner with a rating between 1 and 10. For each patener a name and rating should be submitted in the comment section of the BBLearn submission, and not in the report document. You do not have to tell your partner the rating you assign them. A rating 8 or more indicates that your partner was particularly helpful or contributed exceptional effort. A rating of 5 indcates that your partner met the class expectations of them. Rating your partner 3 or less means that they refused contribute to the project, failed to put in a resonable effort or actively blocked you from participating. In the case where you give a very low rating, please explain why in the comment section of the submission. Those who recieve low scores may be asked to explain their actions to the lab staff, and additional low ratings after a warning could lead to losing a letter grade, or even failing the course. Consistant high ratings from partners are noted durring the grading process, and may be taken into account when rounding your final grade.

Colophon