ControlPanel

1 Overview

In our Painting Cars lab earlier in the semester we were challenged to create a program that would draw cars with various properties at various locations on a canvas. To do so we used boilerplate code that created a class which extended JComponent. So what exactly is a JComponent? A JComponent is a visual part of an application that can be integrated into a larger application. While we have already created our own component, many useful components come built into java, including buttons, text fields, sliders and more. These control based components are often refered to as widgets. In this lab we will take our old CustomCarComponent from our previous lab and use it in combination with the built in widgets to create a Graphical User Interface (GUI) for the user to add their own cars to the screen. We will start by upgrading our old component into a more complete, dynamic version of itself, and then build it into an application.

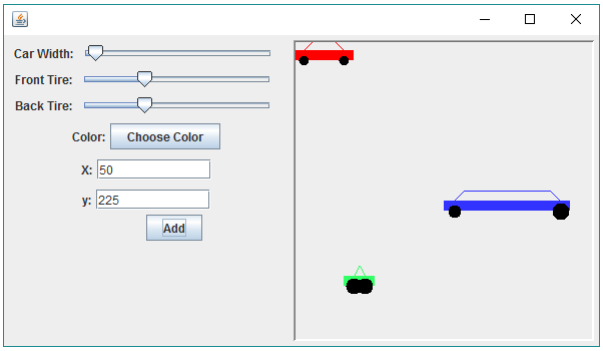

Fig. 1: A set of colored custom cars

2 Learning Outcomes

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

- write application using the Java Swing framework;
- utilize an event listener system to perform callbacks;
- 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:

Describe the following Swing classes. When would these classes be used? How do you instantiate these classes, and what accessors might be particularly useful? How would you write code to trigger when these components are used? Answer these questions for each class below, when appropriate for the class. Turn in your answers on BBLearn.

- JSlider
- JComponent
- JTextField
- JBorderLayout
- JButton
- JColorPicker

4 Lab Instructions

Do this part in lab:

Step 1

Before we can go about making our control panel we will need to revisit our CustomCarComponent and modernize it a bit. Retrieve a team members CustomCar and CustomCarComponent classes from the previous lab. You should be able to access your own BBLearn submission to retrieve your work if you need to. You can bring your PaintBucket over as well for testing, but it will not be used in this lab. Make sure your code compiles and runs. Now we will go back into car component and make sure it is a cohesive class which can be controled from the outside. In our previous assignment it was likely that the cars you generated were inside the paintComponent method. That was good for the time, but now we will want to have our car component object draw cars dynamicly based on an internal list. Add an internal list of cars to draw, an internal list of points to draw those cars at (I recomend using the Point2D class) and add a method addCar that accepts both a CustomCar and a point. Paint component should go through the internal car list and draw all the cars.

Also, instead of hoping the frame is the right size, lets have our component define its own size. Override JComponent's getPreferredSize() method to return a dimension object specifying 300 by 300. Modify your main code to not

set the size of the frame, but to instead call the pack() method after the component is added.

Move your logic to build cars to the main method of your CustomCarComponent class, utilizing the new addCars method, and ensure that you can still build the same scene using this new structure. Once this is done we will be ready to add our new control panel.

Step 2

Now we will create our new application. Where as previously we just used a frame to show our component, in this application we will have other controls alongside our custom car component. Create a class for the frame and build a main method that instantiates the frame. Your frame will consist of a set of controls along with an 'Add' button. When the add button is pressed, a new car is built and added to the CustomCarComponent. The location and properties of this car are controled based on the control inputs. You will add the following controls:

Car width JSlider A slider that ranges from 0 to 200, controling how long the car is.

Front tire size JSlider A slider that ranges from 0 to 50 controlling the diameter of the front car wheel.

Back tire size JSlider A slider that ranges from 0 to 50 controlling the diameter of the front car wheel.

Color A widget to choose the color of the car. Mine uses a button that opens a JColorChooser.

X coordinate A text box where the user can enter the X axis value for the top left of the car.

Y coordinate A text box where the user can enter the Y axis value for the top left of the car.

Each of your widgets should be labeled as shown on the example screenshot. In order to get the labels, controls and CustomCarComponent placed properly you will have to understand and make use of layout managers. You are encouraged to lookup Swing tutorials online to help you put together the frame and components into a pleasing arangement. A recommended approach is to make your 'Add' button first, and have it stamp a car with hardcoded values. One by one add new controls and replace your hard coded values with values from the widget.

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