Comp 110-003 - Assignment 6:

Structured Objects and Graphics

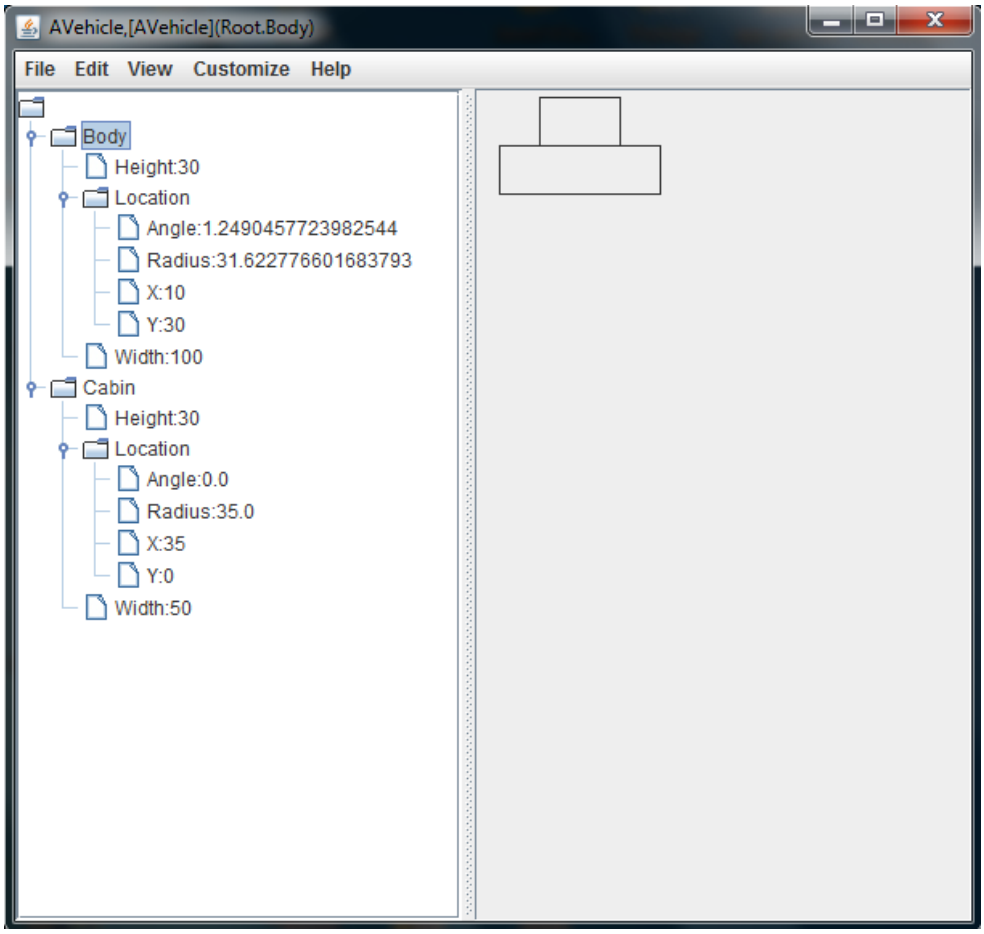
Date Assigned: Wed Oct 26, 2011

Completion Date: Wed Nov 2, 2011 (midnight)

In this assignment, you will gain first-hand experience with graphics and structured objects. You will also practice working with class members.

Much like ATemperatureConverter and ATemperatureSpreadsheet served as a basis of your assignments so far, the code you write in this assignment will serve as a basis for the next couple of assignments.

The main task in this assignment is to draw a vehicle. The UI for the vehicle generated by ObjectEditor should look similar to the one shown below. To complete the assignment, do each of the parts below in the sequence they are specified.



Part 0 – Point and ACartesianPoint

First you need to have the basic Point working. To do this, create Point.java and ACartesianPoint.java files. Copy and paste the `Point` interface and `ACartesianPoint` class code provided in the course notes into the two files, respectively.

Part 1 – Rectangle Interface:

Create an interface called `Rectangle` that defines the following editable properties:

- a property of type Point called Location
- a property of type int called Width
- a property of type int called Height

Look at the `AnotherLine` interface in the course notes for an example of how to do this.

Part 2 – ARectangle Class:

Create a class called `ARectangle` that implements the `Rectangle` interface from above. Your class should implement the following constructor:

```
public ARectangle(int initX, int initY, int initWidth, int initHeight)
```

Look at the `AnAnotherLine` class in the course notes for an example of how to do this.

Part 3 – Vehicle Interface:

Create an interface called `Vehicle` that defines the following read-only properties:

- a property of type Rectangle called Cabin
- a property of type Rectangle called Body

Part 4 – AVehicle Class:

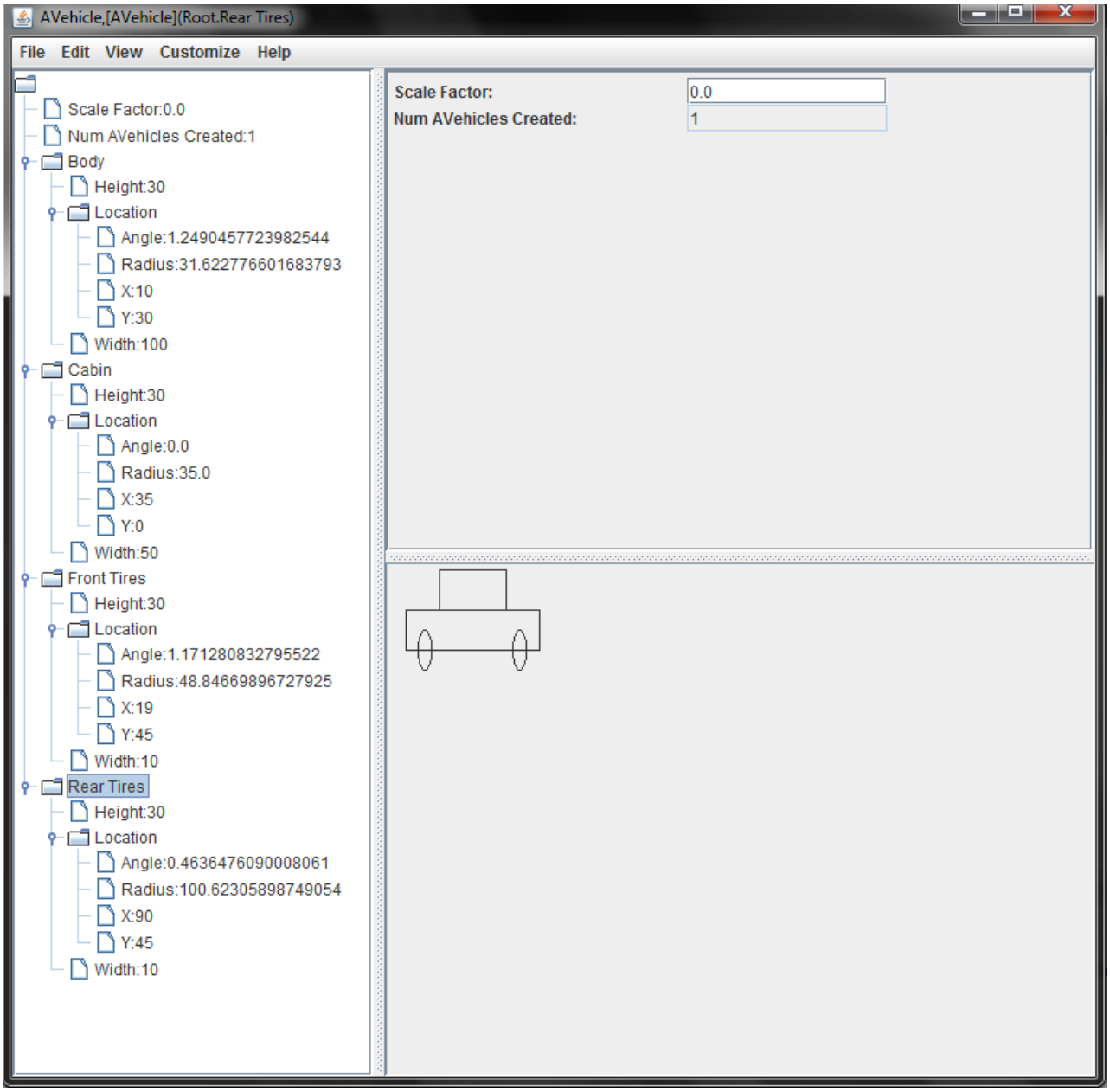
Create a class called `AVehicle` that implements the `Vehicle` interface from above. Your class should implement the following constructor:

```
public AVehicle(int width, int height)
```

where width > 3*height. Based on the height and width, you need to build your car as follows:

- the height of the cabin and the body should **both** equal the value of the `height` parameter
- the cabin should be half the width of the body
- the cabin should be horizontally centered on the body
- the bottom of the cabin should lie on the top of the body

Make sure that you avoid as much code duplication as possible.



Part 5 – Modifying AVehicle – Scaling

Extend `AVehicle` by adding to it an editable ScaleFactor property, which works as follows. When the property is updated, your vehicle size should be multiplied by the value of the property – if the `ScaleFactor` value is 2, then your vehicle should grow to two times its current size; if the `ScaleFactor` value is 3, then your vehicle should grow to three times its current size. When you apply the scale, you must make sure that design specifications in part 4 are satisfied.

Part 6 – Modifying AVehicle – Class Members

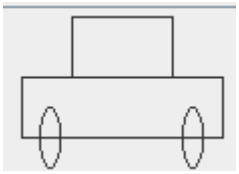
Extend `AVehicle` by adding to it an **int** property called `NumAVehiclesCreated` that returns the number of times that `AVehicle` has been instantiated. The UI shown above illustrates the NumAVehiclesCreated property. In the example, only one `AVehicle` was created so the value of the "Num AVehicles Created" is 1.

Part 7 – Adding Tires

Add a front and a rear tire to your vehicle. Tires should be displayed as circles and follow these design guidelines:

- the height of a tire should be the same as the height of the body
- the center of the tire should be vertically aligned with the bottom of the body
- the closest outside part of the front (rear) tire to the front (rear) of the vehicle should be exactly "one third of the body height" pixels away from the front (rear) of the vehicle

If you do this successfully, the UI generated by ObjectEditor will look something like the image shown below. Make sure that the `scale` method works with the tires, as well. Also, make sure to avoid code duplication wherever possible.



Hint: *You need to define an Oval interface and an AnOval class using the same procedure you used to define Rectangle and ARectangle, respectively. Also, you will need to add two read-only properties to AVehicle for the front and rear tires.*

Part 8 – Bonus (20%) – Written Part

To get the bonus points, modify the code you have written so far in some way but still keep it consistent with the ObjectEditor shape rules. For example, try changing the names of the properties, classes, and interfaces. Which ones can you change and how and still have ObjectEditor draw the car for you? How can you break the ObjectEditor shape rules? What happens in this case? Try changing some other things and report on your findings.

This part, you should write in a text document and include the text document when you turn in the assignment on Blackboard.