CSCI 202: Object-Oriented Programming

Homework 2 - Due Thursday February 15, 2024 at 11:59pm

Resources:

- Java API index: https://docs.oracle.com/en/java/javase/17/docs/api/index-files/index-1.html

In this homework you will be creating a JavaFX program for drawing several vehicles on the screen. The program will have buttons for adding cars to the screen, adding trucks to the screen, moving the vehicles to random positions, and removing all vehicles from the screen. Each time a new vehicle is added it will be given a random color and position. When your program is finished it will look similar to the following (see Figure 1):

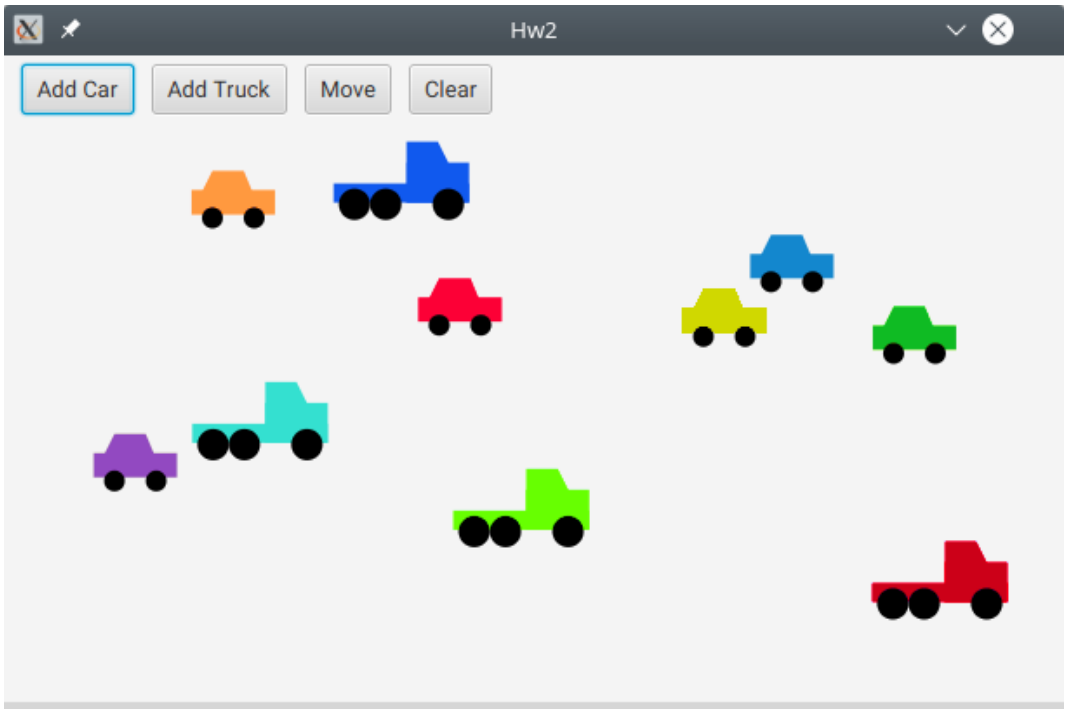

FIGURE 1. Completed Program

Download and unzip Hw2.zip onto your local computer. Then open up the unzipped project with IntelliJ.

The source files are in package hw2. Each vehicle on the screen is represented by an object. Cars are an instance of the `Car` class, and trucks are an instance of the `Truck` class. The abstract `Vehicle` class is the parent class of `Car` and `Truck`. It extends class `Pane`.

All vehicles have an x position, a y position, and a color. The x and y position is the upper left point of the bounding rectangle around the vehicle (the bounding rectangle is not actually drawn). The method

```
public void setPosition(double x, double y)
```

in class `Vehicle` should be overridden in subclasses to update the position of the subclass' children nodes.

Class `Hw2` consists of code for the main JavaFX application. It has three non-static private data fields: an `ArrayList` of `Vehicles` (called `vehicles`), a `Pane` called `drawPane`, and a random number generator. The vehicle objects are stored in `ArrayList vehicles` and are drawn on `drawPane`.

Part 1: Finishing the Car class

The shapes for drawing a `Car` object are stored as non-static private data fields of class `Car`. We must update the x and y positions of each of these shapes whenever a car's position is changed. This is to be done in method `updateChildren()` (your task to write). All other properties of these shapes (such as their color, radius, width, and height) are to be initialized in constructor `Car(double xPos, double yPos, Color theColor)` (your task to do).

Your task: initialize/update the properties of the shapes for drawing a car. Use the relative coordinates (offsets from (x, y)) in the following image (see Figure 2):

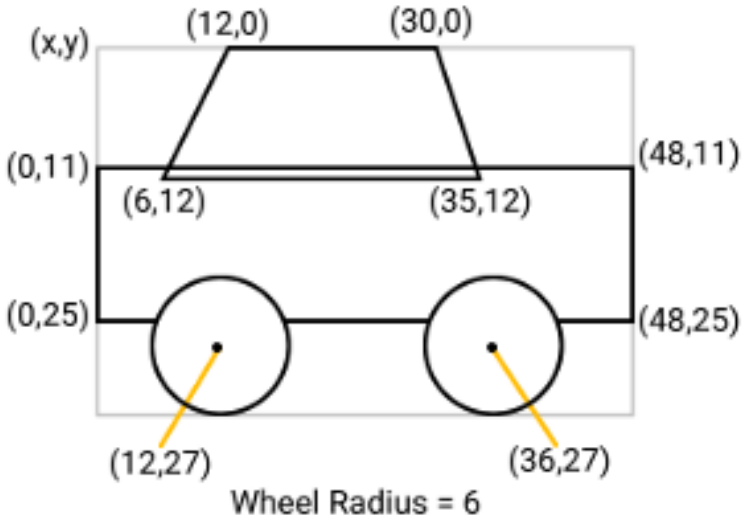

FIGURE 2. Relative Car Coordinates

The x and y coordinate of the vehicle can be retrieved with `getX()` and `getY()` respectively.

The color of the car must be set to the vehicle's color. We can get the vehicle's color by calling `getColor()`.

Set the color of the wheels to `Color.BLACK`.

The width and height of a rectangle `r` can be set to `w` and `h` respectively with:

```
r.setWidth(w);
r.setHeight(h);
```

The coordinates of a rectangle `r` can be set to position (`x`, `y`) with:

```
r.setX(x);
r.setY(y);
```

The center of a circle `c` can be set to position (`x`, `y`) with:

```
c.setCenterX(x);
c.setCenterY(y);
```

The radius of a circle `c` can be set to `r` with:

```
c.setRadius(r);
```

We can reposition a `Polygon p` with:

```
ObservableList<Double> points = p.getPoints();
points.clear(); // remove all points from p
points.addAll(
    x1, y1,
    x2, y2
    x3, y3);
```

Here `(x1, y1)`, `(x2, y2)`, and `(x3, y3)` are the new points to position `p` to (add more points as needed).

When a car is being drawn, its shapes are not drawn unless they are added to its list of children. To accomplish this, at the end of constructor `Car(double xPos, double yPos, Color theColor)` add the line

```
getChildren().addAll(roof, body, backWheel, frontWheel);
```

Part 2: Finishing the Truck class

Finish writing the `Truck` class so that it draws a truck when being displayed. This is similar to the `Car` class. Remember, all drawing must be done relative to the x and y coordinate of the truck.

The color of the truck must be set to the vehicle's color. Set the color of the wheels to `Color.BLACK`.

Use the relative coordinates in the following image to draw the truck (see Figure 3):

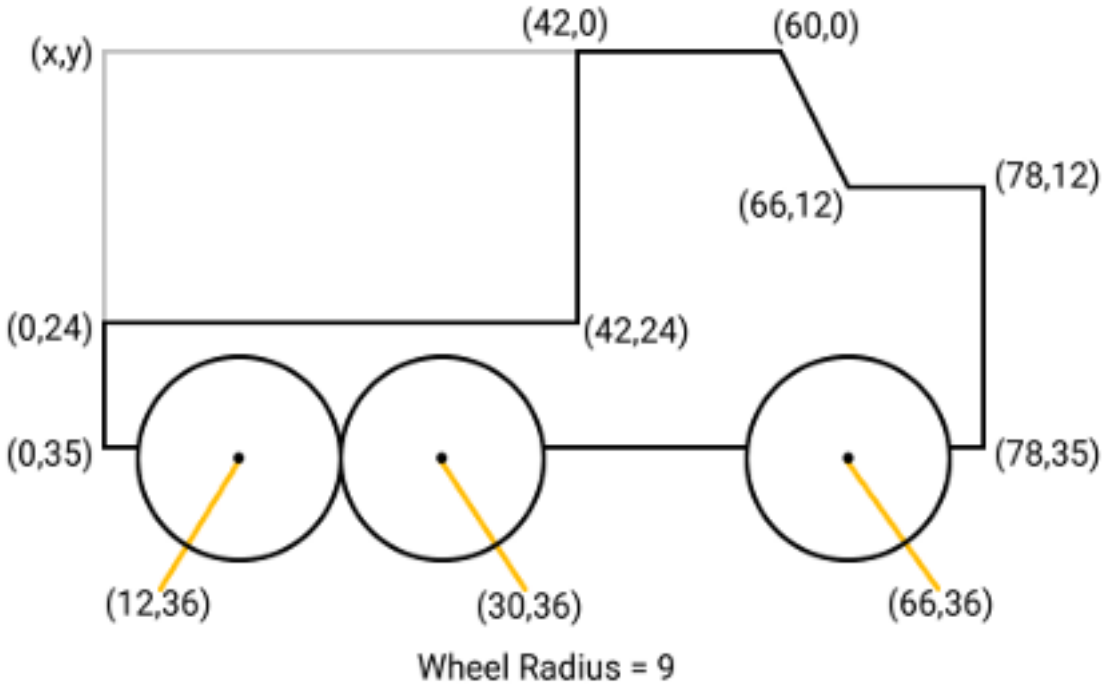

FIGURE 3. Relative Truck Coordinates

Do not add anymore members to the public interface of class `Truck`. You may, however, add any needed private members. Note: overloading an inherited public method does not add it to the public interface because it is already a part of the public interface.

Part 3: Creating the method `addTruck`

In class `Hw2` create a private method called `addTruck(ActionEvent e)` that is similar to the `addCar(ActionEvent e)` method. This method will be called when the "Add Truck" button is clicked (the "Add Truck" button will be created in another step). In the body of this method create a new `Truck` object that is initialized to a random color. Then set this truck to a random position, add this truck to the `vehicles` ArrayList, and finally add this truck to `drawPane`.

Part 4: Creating the Buttons

In the `start` method of class `Hw2` add the following buttons:

1. A button with the text "Add Truck" on it for adding a new truck. Connect this button to the method `addTruck(ActionEvent e)`. Add this button to `topBox`.
2. A button with the text "Move" on it. Connect this button to the method `move(ActionEvent e)`. Add this button to `topBox`. Note: the move button will not do anything yet since the `move(ActionEvent e)` method is not yet completed.
3. A button with the text "Clear" on it. Connect this button to the method `clear(ActionEvent e)` to remove all vehicles on the screen. Add this button to `topBox`.

Part 5: Writing the functionality for the "Move" button

The `move(ActionEvent e)` method of class `Hw2` is called whenever the "Move" button is clicked. Its task is to move each vehicle on the screen to a new random position.

The vehicles in our program are stored in the `ArrayList vehicles`.

Your task is to write the body of `move(ActionEvent e)` to do the following: for each `Vehicle v` in the `ArrayList vehicles`, move `v` to a random position on the screen by calling:

```
randomlyPosition(v);
```

Note: In method `randomlyPosition(Vehicle v)` the call

```
v.setPosition(
        generator.nextDouble()*MAX_X_POS,
        generator.nextDouble()*MAX_Y_POS);
```

does dynamic dispatching. The variable `v` is polymorphic. At runtime the JVM will look at the data type of the object that `v` references to determine which version of `setPosition` to call (`Car`'s `setPosition` or `Truck`'s `setPosition`).

When you are done, zip up your project in a file called Hw2.zip and upload this .zip file to Canvas (click on Assignments and go to Homework 2).