Lab 10: Going For A Drive

Assigned: Thursday, April 25, 2019
Due: Wednesday, May 1 at 11:30pm

1 Objective

In this assignment you will write a "normal" class – one that does not have a `main` method or any static methods at all, but instead has instance variables, a constructor, and instance methods. This class will not be a complete program, but instead a component that could be used in a program, the same way that we have been using `Scanner`, `File`, `Random`, and other such classes.

2 Overview

In this lab you will be writing a class that represents a gasoline-powered motor vehicle, such as a car, van, bus, or truck. A vehicle has many attributes that could be fields and many behaviors that could be methods – too many for us to include them all. Instead, we will abstract away the unimportant details and focus on just a few things:

- A gasoline-powered vehicle has a tank for storing fuel. Driving the vehicle uses fuel, and refueling the vehicle replaces some. There are limits on the amount of fuel that can be stored.
- A vehicle uses a sensor called the trip odometer to remember how many miles it has been driven. Driving the vehicle increases the value of the trip odometer, and it can be reset back to 0.
- The fuel efficiency of a vehicle determines how much fuel is needed to travel a certain distance. This is different from one vehicle to another, and also different when driving at different speeds.
- Each vehicle has a unique vehicle identification number that is assigned when it is manufactured.

3 Setup

Create a `Lab10` Java Project in Eclipse, as usual. Create a new class in it named `Vehicle`. Download the file `CarProgram.java` from D2L and move it into your project's `src` directory. This file will have syntax errors until you get through step 4 of the instructions, at which point you will start working with it.

As always, you can submit to AutoLab after every step to get feedback.

4 Assignment

Make the following changes to the project:

1. (10 points) Add the following fields to the `Vehicle` class:
   (a) `vehicleIdentificationNumber` stores the vehicle's unique identifier (just an arbitrary `String`, because these identifiers have letters too).
   (b) `maximumFuelLevel` stores the amount of gasoline that can fit in the vehicle's tank, in gallons.
   (c) `currentFuelLevel` stores the amount of gasoline in the vehicle's tank, in gallons.
   (d) `tripOdometerReading` stores the value displayed by the vehicle's trip odometer (the number of miles driven since it was last reset).
   (e) `maximumFuelEfficiency` stores the fuel efficiency when the vehicle is traveling at an optimal speed, in miles-per-gallon.

2. (10 points) Add a constructor to the `Vehicle` class. It should have three parameters:
   - The VIN of the new vehicle.
   - The maximum fuel level of the new vehicle.
   - The maximum fuel efficiency of the new vehicle.

   The constructor should initialize the fields as follows:
   - The `vehicleIdentificationNumber` field should be copied from the associated parameter.
   - The `maximumFuelLevel` field should be copied from the associated parameter.
   - The `currentFuelLevel` field should be copied from the `maximumFuelLevel` field (every vehicle starts with a full tank of gas).
   - The `tripOdometerReading` field should be set to 0.0 (new vehicles have not yet driven any miles).
   - The `maximumFuelEfficiency` field should be copied from the associated parameter.

   It would not make sense to create a gasoline-powered vehicle whose tank stores a negative amount of fuel, or even 0. Nor does it make sense for a vehicle's maximum fuel efficiency to be negative or 0. So if either of these parameters is not positive, throw an `IllegalArgumentException` with an appropriate message.

3. (10 points) Create accessor methods for the vehicle identification number, current fuel level, and trip odometer reading.

4. (15 points) Create a method named `toString` that has no parameters and returns a `String`. That `String` should have exactly the following format (but not those exact values – it should use the field values):

```
Vehicle 1001
Fuel level is 16.3 / 18.0 gallons.
The odometer says 50.0 miles.
Maximum fuel efficiency is 30.0 miles per gallon.
```

Note that this format is rounding all `double` values to a single digit after the decimal place. We have learned how to do that before when printing, with the `printf` method, but now we are not printing – we are constructing a `String` that we want to return. Thankfully, there is another method that can help us. There is a method named `String.format` that takes exactly the same kind of parameters as `printf` and interprets them exactly the same way, but writes all of the output to a `String` that it returns instead of printing it.

You now have enough of the class written that my `CarProgram` class will compile and be useful. Read through that class to see how the `Vehicle` class might be used. Note how I divided the program up into methods. Run it (only menu options 1, 2, and 9 will currently work), and make sure that they do.

5. (15 points) Create a method named `getEffectiveFuelEfficiency` that has one parameter: a speed (in miles per hour). It should return the fuel efficiency that the vehicle has when driving at that speed. Although every vehicle already has a maximum fuel efficiency, most vehicles have worse than that maximum performance when they are driving too quickly or slowly. The purpose of this method is to compute the fuel efficiency of the vehicle at the speed given in the parameter, according to this table:

Speed Range	Effective Fuel Efficiency
Below 25mph	50% of maximum
At least 25mph but below 45mph	80% of maximum
At least 45mph but below 65mph	100% of maximum
At least 65mph but below 85mph	80% of maximum
At least 85mph	50% of maximum

6. (15 points) Create a method named `drive`. It should have two parameters:

   - The amount of time that the vehicle is being driven (in minutes).
   - The speed at which the vehicle is being driven (in miles per hour).

   This method does not return a value. Instead, it checks that both of its parameters are positive, and if not throws an `IllegalArgumentException` with an appropriate message. Then it computes the total distance the vehicle will travel during the trip. Then it computes the effective fuel efficiency the vehicle will get during the trip (use the method you just wrote). Then it computes the amount of fuel that would be used during the trip. If the vehicle does not have enough fuel to make the trip, an `IllegalArgumentException` should be thrown with an appropriate message. Otherwise, the current fuel level and trip odometer reading should be updated.

   You can now un-comment line 106 in `CarProgram` and run it to test that driving the cars works.

7. (10 points) Create a method named `refuel` that has one parameter: the amount of fuel that should be added to the vehicle's tank. This method does not return a value. If the amount to be added is not positive, or if adding that amount would cause the tank to overflow, the method should throw an `IllegalArgumentException` with an appropriate message. Otherwise, it should adjust the current fuel level accordingly.

   You can now un-comment line 118 in `CarProgram` and run it to test that refueling the cars works.

8. (5 points) Create a method named `resetTripOdometer` that has no parameters and does not return a value. This method should set the trip odometer reading to 0.0 miles.

   You can now un-comment lines 45 and 48 in `CarProgram` and run it to test that resetting the trip odometer works.

9. (10 points) AutoLab will now automatically check for Javadoc comments and good style. (It can't find all types of bad style, though, so I will still be checking some things.) Ensure that you are making it happy and that you think you will make me happy as well.

5 Submitting Your Work

Make sure that you have submitted your final version to the `Lab 10: Going For A Drive` AutoLab page, and that (unless you are satisfied with a lower score) you have 100/100 autograded points.

You do not need to submit your work through D2L – I will look at whatever your last AutoLab submission is.