Project 1 - Fuel Monitor

Due Sep 16 by 12pm **Points** 80

Overview

In this assignment, you will develop your first complete class, following the techniques described in Chapter 3. You will also need to use several techniques from Chapter 4 regarding data types and arithmetic, in order to manipulate the numeric values involved. You will also develop some tests to help ensure your class is working properly.

Details

Start a new project in Eclipse for this assignment. In this project, you'll develop a class called `FuelMonitor` in the `edu.vt.cs5044` package that models the behavior of an automobile's fuel gauge, with indicators for miles-per-gallon and miles remaining, with two driving modes. A basic template of the required class file is provided to help get you started. You should assume that the methods of your class will be called by other components within the vehicle (but external to this assignment). Each method's required behavior is already described by Javadoc comments in the provided source code file; these comments act as your requirements by specifying the expected behavior for each method.

You have freedom to choose the number and types of instance variables you use, the approaches to the calculations, and other such internal design considerations. This is the most fundamental concept of encapsulation in an object-oriented system. It doesn't matter exactly *how* your class works, as long as it works properly whenever the public accessor and mutator methods are invoked. There are many possible designs, and there is no single "right" or "best" solution that meets all of the functional expectations.

However, we're also taking this opportunity to start to practice development in the idiomatic style of Java, so please be sure to review the **project grading rubric** and follow its guidelines to maximize your score from the human review process. Of course you may always ask for help in the forum at any time!

Note that our `FuelMonitor` class isn't a complete stand-alone program. It can't be launched on its own, as it doesn't include a main() method. Further, no methods accept any user input, nor do any methods produce any user output. This is intentional! This class is meant to be just one small component of a much larger system.

Still, you obviously need to execute your code somehow, to ensure that it works correctly. To do this, you'll need to develop a second class called `FuelMonitorTester` (also in the `edu.vt.cs5044` package) in order to exercise your `FuelMonitor` class. Your tester class does have a main() method, and will generally use `System.out.println()` to display both the expected and the actual results to the console, as suggested in the textbook. A starting tester class is provided, to get you started, but you're expected to add several more test cases to this.

Downloads

- **FuelMonitor.java** This class file contains all the method declarations you need, along with Javadocs describing their required behaviors, but the method implementations are all just placeholders. You will need to design and develop the method implementations, adding private instance variables as necessary. You are encouraged to add private methods as well, wherever it can help reduce redundancies in your code. Please ensure you don't add, remove, or change the headers of any of the public methods. The existing public methods represent the external interface of your class, and must be retained. Other components of the larger system (outside of this assignment) will expect to work with your class using exactly these methods.
- **FuelMonitorTester.java** This class file runs some basic functional tests on a FuelMonitor object. Be sure to read through all the comments of this file before starting the assignment, as there are many sample calculations that might help clarify some of the behavioral requirements. You're expected to develop additional test code in this file. **You must add enough tests to exercise each of the FuelMonitor methods at least two times each** (above and beyond the existing tests).
- **expected_output.txt** This is the expected output of running the above tester code (as provided) on a properly working implementation.

Submission to Web-CAT

Submit your solution to **Web-CAT** (https://web-cat.cs.vt.edu/) via the Eclipse plug-in. Please feel free to improve and resubmit your code as many times as you like (before the due date) in order to improve your score.

Important Notes:

- Your submission will be evaluated both by Web-CAT (40 points) and by human (40 points). As such, Web-CAT's automated score will show at most 40/80 points.
- Your tester class won't be evaluated by Web-CAT for this project. However, your tester class will be evaluated by a (human) grader as part of your overall score. Javadocs are not required in the tester source code, and the normal style rules won't be enforced, but you must still add some comments to describe, at least generally, the intent of each of your additional test cases.