Programming Paradigms
Lab 8 & 9

Jacek Wasilewski

Exercises

1. Design a set of classes that work together to simulate a car's fuel gauge and odometer.

   The FuelGauge Class: This class will simulate a fuel gauge. Its responsibilities are as follows:

   - To know the car's current amount of fuel, in gallons.
   - To report the car's current amount of fuel, in gallons.
   - To be able to increment the amount of fuel by 1 gallon. This simulates putting fuel in the car. (The car can hold a maximum of 15 gallons.)
   - To be able to decrement the amount of fuel by 1 gallon, if the amount of fuel is greater than 0 gallons. This simulates burning fuel as the car runs.

2. There are two types of fuel gauges: analog and electronic. Analog fuel gauge is not precise - it only shows if tank is full (15 gallons), 3/4 full (12 - 14 gallons), 1/2 full (8 - 11 gallons), 1/4 full (4 - 7 gallons) or empty (0 - 3 gallons). If there are 14 gallons in a tank, it should show 3/4; if 8 then 1/2; if 7 then 1/4. Electronic gauge shows everything precisely. Create classes that extend the FuelGauge and implement the above logic.

3. Design the **Odometer Class**. This class will simulate the car's odometer. Its responsibilities are as follows:

   - To know the car's current mileage.
   - To report the car's current mileage.

- To be able to increment the current mileage by 1 mile. The maximum mileage the odometer can store is 999,999 miles. When this amount is exceeded, the odometer resets the current mileage to 0.
- it is possible to manually reset the current mileage to 0.
- To be able to work with a FuelGauge object. It should decrease the FuelGauge object's current amount of fuel by 1 gallon for every 24 miles traveled. (The car's fuel economy is 24 miles per gallon.)

4. Demonstrate the classes by creating instances of each. Simulate filling the car up with fuel, and then run a loop that increments the odometer until the car runs out of fuel. During each loop iteration, print the car's current mileage and amount of fuel.

5. Our gauge and odometer exist outside a car. Create a **Car** class that consists of a fuel gauge and an odometer. Move the simulation code inside your car class.

6. We have cars with analog or electronic gauges. Create subtypes of Car class, one with analog other with electronic gauge.

7. There are also two types of odometers: electronic or mechanical. The only difference is how they present car's current mileage.

8. Our car can have either electronic or mechanical odometer - reflect this in your class model.

9. Car needs a driver - driver just drives a car and they can drive any type of cars.

10. Check if your classes follow "composition over inheritance" and "communication through interfaces" principles (is your car using interfaces/abstract or concrete objects?) . If not, change your code.