# Assignment 10: The Car Class

## Objectives

* Create a Car class that keeps track of how much gas is in the tank and the odometer.
* Create an application that utilizes that Car class to drive a car until it runs out of gas.

## Motivation

Object-Oriented Programming (OOP) is about rethinking how we write code. OOP centers around objects. In this case, the object is a Car. Objects do things. They have a status which can change based on what is doing.

This homework assignment has lots of math. It's simple math, but there's still lots to compute.

For instance, a new car will have an odometer of 0. This is the car's initial state. After the car drives 10 miles, the odometer will be 10. If it drives 10 more miles, the odometer will be 20. Each time the car is driven, the odometer will increase. **It should never decrease.**

Here are the fields that a Car should have (all of which will be represented with **double** values).

* The odometer. Initially this is 0.
* The fuel. This will be set by the user when the car is created. In this car, gas will drop to zero as the car is driven and the gas will never be refilled. When the gas in a car hits 0, the car stops moving.
* The MPG (miles per gallon) of the car. This will be set by the user and should be defined as **final**.

Here's a few important terms which will be necessary for completing this assignment.

* Distance is always measured in miles.
* Fuel is always measured in gallons.
* Time is always measured in hours. Thus 0.5 hours is 30 minutes.
* The speed of a car is the **miles per hour**.
* The car will drive a certain number of **miles per gallon** (or MPG).
* To compute trip distance, you **multiply** the speed and the time.
* To compute the maximum distance that a car will drive, you **multiply** the MPG and the amount of fuel in the tank.
* The gas required for a trip is the distance **divided by** the MPG of the car.
* If the gas required for a trip is less than the gas in the tank, the car drives the trip distance. The odometer is incremented by the trip distance. The fuel is decremented by the gas required.
* If the gas required for a trip is equal to or more than the gas in the tank, the car drives the maximum distance and then stops. The odometer is incremented by the maximum distance. The amount of fuel in the tank should be 0.

Example. My car has a 9 gallon tank and gets 33 miles to the gallon.

* If I drive 70 miles per hour for 3 hours, then
  + The trip distance is 210 miles.
  + The required gas is 6.4 gallons.
  + The odometer will be 210 at the end of the trip.
  + There should be 2.6 gallons of fuel left in the tank.
* If I drive 30 miles per hour for 0.5 hours, then
  + The trip distance is 15 miles.
  + The required gas is 0.5 gallons.
  + The odometer will be 225 at the end of the trip.
  + There should be 2.2 gallons of fuel left in the tank. (It's 2.2 due to rounding instead of 2.1.)
* If I drive 50 miles per hour for 1.5 hours.
  + The trip distance is 75 miles.
  + The required gas is 2.3 gallons. I don't have that much in the tank.
  + The maximum distance I can travel on 2.2 gallons is 72.0 miles. (It's 72.0 due to rounding.)
  + The odometer will be 297 at the end of the trip.
  + There should be 0 gallons of fuel left in the tank.

## Instructions

Name your project FirstnameLastnameAssignmentNumber

Have your program do the following.

Tips: The only primitive data type that I used in this assignment was **double**. All inputs were **double** too. Because everything is a double, you'll need to display everything to 1 decimal place using **printf**. Also, you must document your public methods in the **Car** class with a description, precondition, postcondition, a description of each parameter, and the returned value.

### Part 1: Car Class

Create a class named **Car** which contains the fields mentioned above.

In addition to the fields mentioned above, you'll need these public methods. The precondition for any method which takes an argument is that the argument must be positive. You'll have to determine if your method as a postcondition. (As postcondition is any method which changes the object's fields.) **Also, no methods will display anything and you'll be counted off severely if you have print statements in your Car class.**

#### public Constructor(double gas, double mpg)

This method will assign **gas** to the field for the **fuel** and **mpg** to the car's **milesPerGallon** field. The **odometer** field will always begin at 0.

#### public double getFuel()

This method will return the fuel (i.e. the amount of gas left in the tank).

#### public double getOdometer()

This method will return the odometer reading.

#### public double drive(double speed, double time)

This method will attempt to drive the car at this speed for this amount of time as long as there is enough gas in the tank. If there's enough gas or not, update the odometer and return the distance driven. In general, I did these steps.

1. Compute the trip distance.
2. Compute the required gas.
3. If there is more gas in the tank than is required,
   1. Then subtract from the fuel the required gas.
4. If there is not enough gas in the tank,
   1. Compute the maximum distance I can travel. This is the new trip distance.
   2. Set the gas tank to 0.
5. Update the odometer by adding to it the trip distance.
6. Return the trip distance.

### Part 2: The main method

1. Greet the user: "Welcome to Dr. Church's Car Simulator."
2. Prompt the user for the starting amount of gas in the tank for a car.
3. Prompt the user for the starting miles per gallon for a car.
4. Using the Car class that you created, create a Car object.
5. Report to the screen that you created a car object with an amount of gas in the tank and a specified miles per gallon. Display this to 1 decimal place.
6. Display a message to the screen: "We will loop while there is gas in the tank."
7. While the car's remaining fuel is greater than 0:
   1. Prompt the user for a speed (in miles per hour).
   2. Prompt the user for a time (in hours).
   3. Call the "drive" method on your car object to drive a particular speed and time. Make sure that you save the distance traveled.
   4. Print each of the following to one decimal place: distance traveled, the odometer, and the remaining fuel in the tank.

The program should continue to loop until the car's fuel tank is completely empty.

## Example

Here's an example run in which I used the same example from earlier in the assignment.

Welcome to Dr. Church's Car Simulator.  
Enter the number of gallons of gas in the tank: 9  
Enter the car's miles per gallon: 33  
We've created a car with 9.0 gallons of gas in the tank and get 33.0 MPG.  
We will loop while there is gas in the tank.  
  
Enter the speed (in miles per hour) at which you will drive the car: 70  
Enter the time (in hours) that you drove the car: 3  
The car drove a distance of 210.0 miles.  
The car odometer is at 210.0 miles.  
There is 2.6 gallons of gas left in the tank.  
  
Enter the speed (in miles per hour) at which you will drive the car: 30  
Enter the time (in hours) that you drove the car: 0.5  
The car drove a distance of 15.0 miles.  
The car odometer is at 225.0 miles.  
There is 2.2 gallons of gas left in the tank.  
  
Enter the speed (in miles per hour) at which you will drive the car: 50  
Enter the time (in hours) that you drove the car: 1.5  
The car drove a distance of 72.0 miles.  
The car odometer is at 297.0 miles.  
There is 0.0 gallons of gas left in the tank.

## Documentation

Your source code must include the following documentation:

* Your name
* The class (CS 2070) and the section number (on ground is 08, online is W1).
* The date on which you turned in the assignment.
* A short description of the software. Usually a sentence or two is sufficient.

## Turning it in.

To turn in your application, find the folder containing your entire project (not the folder with the "java" file), zip it up, and turn it in.