Ryan Erickson (rerickso)
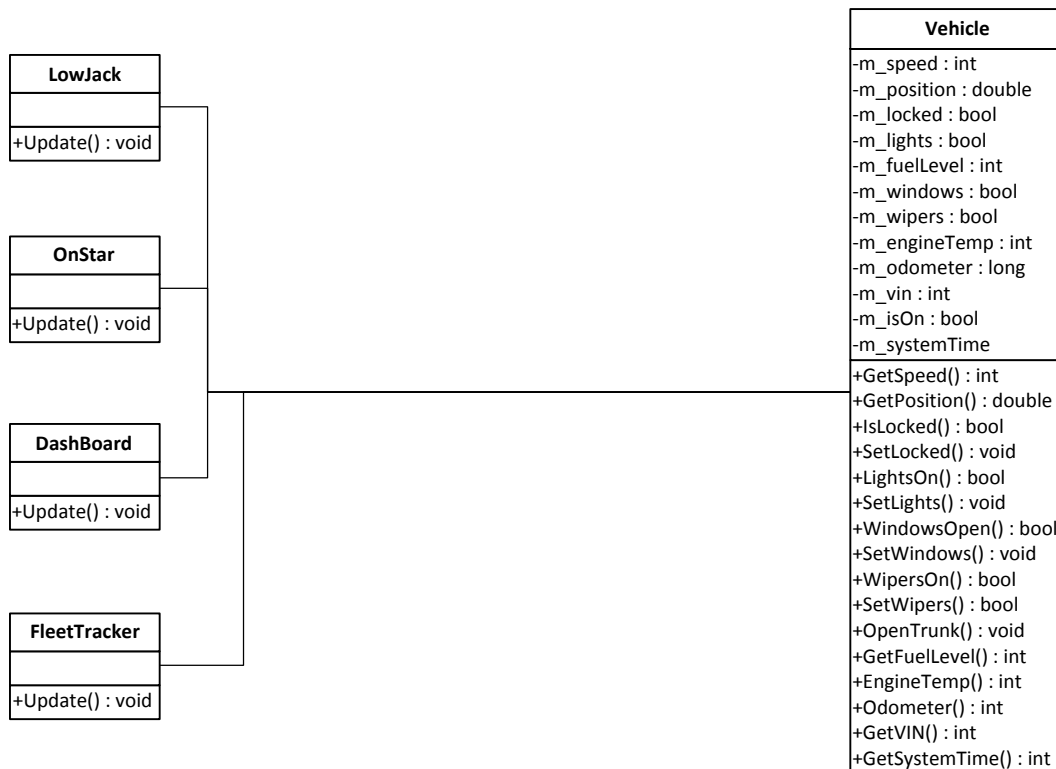Daniel Kuwahara (dskuwaha)
CS246 Assignment 3 – Part 3

a) The problem that we have selected for our assignment is that of a vehicle and its onboard status computer. The computer monitors the all of the vital data about the status of the vehicle. The computer receives this information from the onboard sensors, and then relays it to the OnStar, Low Jack, Fleet Tracker, and gauge cluster and dash board systems. These systems then interpret the data and abstract it for each of their own particular uses. For instance, the onboard status computer reports information about the status of the headlights on the vehicle (ON or OFF), this information would be irrelevant to a low jack system that is responsible for tracking the vehicle should it be stolen. All of the individual systems have unique uses for the data about the vehicle, but all of the data collectively comes from the same source.

```
┌─────────────────────┐
│       Vehicle       │
├─────────────────────┤
│ -m_speed : int      │
│ -m_position : double│
│ -m_locked : bool    │
│ -m_lights : bool    │
│ -m_fuelLevel : int  │
│ -m_windows : bool   │
│ -m_wipers : bool    │
│ -m_engineTemp : int │
│ -m_odometer : long  │
│ -m_vin : int        │
│ -m_isOn : bool      │
│ -m_systemTime       │
├─────────────────────┤
│ +GetSpeed() : int       │
│ +GetPosition() : double │
│ +IsLocked() : bool      │
│ +SetLocked() : void     │
│ +LightsOn() : bool      │
│ +SetLights() : void     │
│ +WindowsOpen() : bool   │
│ +SetWindows() : void    │
│ +WipersOn() : bool      │
│ +SetWipers() : bool     │
│ +OpenTrunk() : void     │
│ +GetFuelLevel() : int   │
│ +EngineTemp() : int     │
│ +Odometer() : int       │
│ +GetVIN() : int         │
│ +GetSystemTime() : int  │
└─────────────────────────┘
```

LowJack
+Update() : void

OnStar
+Update() : void

DashBoard
+Update() : void

FleetTracker
+Update() : void

We will be applying the Observer pattern to this problem to modulate the interface of display objects with the onboard computer status.

b) Below is the UML model for the vehicle information system with the observer pattern applied. As you can see, the pattern objects create an association between "observers" and the "subject". The observers subscribe to the updates of the vehicle's on board information computer. Each time something within the vehicle changes, the computer calls the notify method, this causes each of the individual subscribers to update their displayed information. With this design, I have chosen to have the individual subscribers use the accessors of the subject to update themselves. This is due to the specific nature of each of the different subscribers.