**Creating and catching Exceptions Homework**

In this homework, you will add code in your Car class definition to detect bad situations and **throw** **exceptions** when these occur. **Do not put try-catch in the Car class** (unless you want to do something very cool or trick). After you do that, you will modify the UI code in program.cs to surround certain operations with **try-catch** structures, and then catch any exceptions thrown by the Car object.

First, modify the car class to throw 3 new exceptions

[1] In the Modify the StartEngine method:

Change the logic such that if the user code calls the **StartEngine** method when the engine is already started, the StartEngine method will throw a new ApplicationException and that exception should have a Message property set to “Engine is already started.”.

[2] Modify the StopEngine method:

Change the logic such that if the user code calls the **StopEngine**  engine when the engine is already stopped, the StopEngine method will throw a new ApplicationException and that exception should have a Message property set to “Engine is not running.”.

[3] Modify the Accelerate method:

Change the logic such that if the user code gives calls the **Accelerate** method when the engine is not running, the Accelerate method will throw new ApplicationException and that exception should have a Message property set to “Please start engine.”. (Note that braking should work even if the engine is on or off, so no change is needed for that code.)

[4] Run your program now, and try these 3 nonsensical things (trying to accelerate when the engine is not running, trying to start the engine when it is already started, and trying to stop the engine when it is already stopped) and verify the system pops up “uncaught exception” boxes showing each of your 3 unique error messages. **Only** when you have all 3 “throws” working, **then** move on to modify the console program.cs:

So now modify the 3 *appropriate* cases in your switch statements

* by adding a Try Catch code to three of them.
* Now when you do the 3 nonsense things, you should **catch the errors** and use the returned error Message property (from your throw code over in the car class) and use that message property to update your ErrorMessage variable. If you do this correctly, then the DisplayCarState method will not say “all is well” but will instead display this unique error message.
* When you catch the error

catch (ApplicationException e) make sure to declare the name of your new exception object, which is traditionally called e. When you do that, you can get the error message you created in the throw by using e.Message

[5] But once you modify your ErrorMessage to display the new error, that error message will be stuck no matter what good things the user does. To fix this, modify all paths such that when no problem is encountered to overwrite any previous value in the ErrorMessage variable with the standard message "All is well" such that the errors go away after the user moves on and does something that is not an error. In other words, the status message should always be showing current **correct** status.

Make sure you understand how the error message gets set, and where it is written out: **Do not add any new Console.WriteLines statements!** If you feel the need to, you do not understand the program requirements.