# OOP3 Final Project - Dakota Kronberger

# Description

My final project's purpose is to be a vehicle simulator. You are presented with an opening setup window, in which you can configure the game as you choose. After you set the configuration variables, you enter simulation mode, where you will see an image of your car, along with statistics on your car, its condition, fuel level, and your amount of money. You can start/stop the car, drive the car (which will consume fuel & possibly damage your car), pay for repairs to reset the state of your car's condition, and refuel your car.

You can only refuel/make repairs when your car is stopped, and you can only drive your car when it is running. When your car is running, it will burn minimal fuel in "idle" mode.

# Design Patterns

**Observer Pattern:**

**My project uses the observer pattern and delegates to bubble information back up to the** MainWindow**. The MainWindow acts as the observer, and the car acts as the subject. Whenever fields are updated on the car, such as fuel or condition, the car notifys the** MainWindow of the changes. Then, the MainWindow updates the UI.

**State Pattern:**

**I implemented the State pattern by adding a condition to the car. The condition of the car represents the "state". The car is the context, which has a "damage" method and a "repair" method. The damage method creates a new state for the condition, and the repair method returns the condition to the first state.**

**Factory Pattern:**

**To create cars, the initial condition of the car and maximum fuel is determined by what type of car it is. To create all of this, a CarFactory static class is used.**