# Basic Inheritance

**The goal for this exercise** is to make sure that you can use basic inheritance.

You should familiarize yourself with how inheritance works, using one of your instructor's classic examples: automobiles!

For this exercise you will create a collection of classes that are related by inheritance: you will create a "Car" class, and two more, more specialized versions of the Car class -- a "Ford F-250" class and a “Toyota Prius” class.

The base class (sometimes also referred to as a superclass), is going to be named **Car**. When designing a base class, we look for attributes that all of the derived classes (sometimes called the subclasses) have in common, and make them into data fields on the base class. In this case it seems reasonable to assume that all cars have a gas tank of some size (for right now we will ignore 100% electric cars), and that each car has a weight (in pounds). In order to represent this you will create a class named Car that has appropriate instance variables to represent these two attributes that all cars have in common. Make sure that you make the data field **private**. Also, add the **public** GetWeight/SetWeight, and GetTankSize/SetTanksSize methods onto the class; for the Set methods, your class should only update the private variable if the weight/tank size is non-negative. You should make sure that for all other setter methods in this exercise (and in this lesson) if the setter method is given a negative number, and that negative number doesn’t make sense for that attribute, that your setter method simply does not update the attribute. Additionally, you will create a method named **Print** that will print information about an individual car out to the console. In order to pass the NUnit tests, you’ll need to print out the following, in exactly the same format (including capitalization):

Weight:2000.25

Gas Tank:10.00

(Where 2000.25 and 10.00 will be replaced with the values specific to your Car object)(Note the space between **Gas** and **Tank**, and also note that there are no spaces before or after the colon ( **:** ) )( Note that the real numbers must have exactly two decimal places in the output – make sure you use Console.WriteLine(“…{***x*:0.00**}”); in order to ensure the proper output (where x = 0, or 1, or 2, as appropriate))

At this point, you are expected to put any classes that you create near the Car class, so that your work is easy for the instructor to find, but you are not required to put the new classes in a single, exact location.

Within Basic\_Inheritance.RunExercise() you should create an instance of this new Car class, and tell it to print, in order to make sure that everything works as you expect to.

Next, you should create a subclass that derives from your **Car** base class – a Ford F-250. Make sure that it derives from the Car class. Add to this class a new data member – in this case, add an instance variable **maxPoundsOfCargo** to the truck, to represent how much stuff the truck can carry. Make sure that you add the GetMaxCargo/SetMaxCargo methods, as well. Also add a method named PrintF250(), which will print out all the attributes of the class (feel free to make use of the Car.Print method, if you know how) You’ll create another method, named F250.**Print** method (instead of **PrintF250**) later, once we go over the **new** keyword. For now, you’ll need the PrintF250 method so that your NUnit tests pass. An example of the output that PrintF250 might produce is:

Weight:2000.25

Gas Tank:10.00  
Max Cargo:200.70

(Again, with 2000.25, 10.00, and 200.70 replaced with your values)

Within Basic\_Inheritance.RunExercise() you should create an instance of this new, derived class, make sure to initialize all of its instance variables, and then print them all out, in order to make sure that everything works.

Notice that the new F-250 has the **maxPoundsOfCargo**, as well as the gas tank size, and the overall weight of the car, despite the fact that the F-250 only had to declare the **maxPoundsOfCargo** instance variable. In a sense, this is the goal of inheritance: to allow us to quickly re-use existing code (in this case, the variable declarations) rather than retyping it a second time. Another thing to notice is that this is a very typical use of inheritance – the derived class *specializes* the base class further. In other words, the derived class is everything at the base classes, plus some extra stuff that makes it a more specific category. In a sense, you can think of a sub*class* as being a sub*category* of the base class.

Once you’ve done this, go back and do something similar a second time, such as a Toyota Prius. There are no NUnit tests for this part – you should define this class, create a couple of instances of this new class, and make sure that you’re able to make use of this second class. The purpose of this portion of the exercise is to make sure you’ve gotten some extra practice & repetition with the syntax and concepts.

Remember that the exact syntax to give inheritance is not difficult – if you already have a class named **Base** to find somewhere within a project, you can create a new class that inherits from it with the following code:

public class Derrived : Base

{

}

**What you need to do for this exercise:**

1. Create the Car class, create a Car object or two, and call Print on the Car objects in a console application. Make sure this all works.
2. Create the F250 class, create a object of this class (or two), and call PrintF250 them in your console application. Make sure this all works.
3. Create the Prius class (or whatever you choose), create an object of this class (or two), and call PrintPrius (or whatever) them. Make sure this all works.