# Inheritance: Calling methods in the base class

**The goal for this exercise** is to make sure that you can call a method defined in a base class, from within a method that’s defined in a derived class.

In previous exercises, you added a a **Print** method in the base class, and then went on to add a PrintWithSlightlyDifferentName() method in the derived classe(es). For example – Car.Print, F250.PrintF250(), and Prius.PrintPrius(). This worked well enough, but was somewhat awkward – really, we’d like to just create a Print method on each of the classes that we’re implementing, rather than having a zillion differently named, but conceptually the same, methods.

For this exercise, you should create two classes: A Plane class, and a subclass named JetPlane. The Plane class can carry a certain (whole!) number of passengers, and can carry a certain weight in luggage. Make sure that you also include the GetNumPassengers/SetNumPassengers methods, along with the GetWeight/SetWeight methods. The JetPlane has everything the Plane does, and in addition, it has a certain (whole!) number of jet engines on the plane; the JetPlane class should have the appropriate GetNumEngines/SetNumEngines methods. Make sure to set up at least one constructor (on each of the base, and derived, classes) so that you can initialize all the instance variables (fields) of a Plane (or JetPlane) object when you create either one. These classes should be placed near the Calling\_Methods\_In\_Base\_Class class, which is provided for you in the starter project (which is normally named something like 03\_PCE\_Student\_Code). All instance variables should be declared private, as per normal.

Next, add a **Print** method to the base class (Plane) that will print out all the information about the base class. Example output from this method is:
 NumPassengers:10

 Weight:12001.50

(Note that the output that’s listed above must be generated EXACTLY – there can be no spaces between NumPassengers, the “:”, and the number, etc, etc. Further, 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))

Next, add a **Print** method to the derived class (JetPlane), that will print out all the information about the base class first, then print out all the information that's specific to the more derived class. Example output from this method is:
 NumPassengers:10

 Weight:12001.50

 NumEngines:20

(Note that the output that’s listed above must be generated EXACTLY, including two decimal places on the floating-point numbers)

Syntactically, remember that you'll have to declare the Print method in the derived class as not just **public void Print**, but **new public void Print**. Research what the keyword **new** means in this context, and put a brief explanation into your code (in comments) explaining what it means, and why it's important.

In order to call the base class's method, remember that you need to reference base.Print, rather than just print, like so:

// This code goes into the JetPlane class:

public **new** void Print()

{

 **base.**Print(); // This will call the Plane.Print method

 Console.WriteLine("NumEngines:{0}", numEngines );

}

 Make sure that you can get this to work. Also, try removing the **base.**, and see what happens (put a comment in the code explaining what the error is, and where we've seen this before in BIT 142).

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

1. Create the Plane class, as explained above.
2. Create the JetPlane class, as explained above
	1. You should set up the “Set” methods on all of your classes so that they only update the private variables if the provided parameter is non-negative (unless it makes sense to have a negative number)
3. Include an easy-to-find comment explaining what **new** means
4. Include an easy-to-find comment explaining what happens when you leave out the **base.**, as well as a short guess/explanation as to what’s causing this problem, exactly.