## **Override Inherited Members**

Say that we wanted to make one more vehicle that operates a bit differently than a sedan or truck. We want to use most of the members in Vehicle, but we need to write new versions of SpeedUp() and SlowDown().

What we want is to *override* an inherited method. To do that, we use the override and virtual modifiers.

In the superclass, we mark the method in question as virtual, which tells the computer "this member might be overridden in subclasses":

## public virtual void SpeedUp()

In the subclass, we mark the method as **override**, which tells the computer "I know this member is defined in the superclass, but I'd like to override it with this method":

## public override void SpeedUp

As an aside: there's another way to solve this problem. Instead of using virtual and override to override a member, we can define a new member with the same name. Essentially, the inherited member still exists, but it is "hidden" by the member in the subclass. If this sounds confusing, that's okay! We also think it leads to unnecessary confusion, and that leads to errors. We're going to stick with the virtual - override approach in this lesson.

## ✓ Instructions

1.

Our new Bicycle class will access the Wheels and Speed properties in Vehicle, so make both setters protected again in Vehicle.cs.

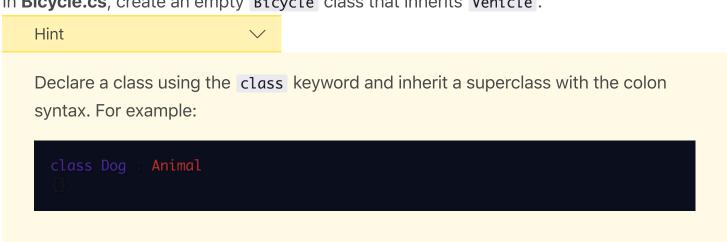
Wheels should already be protected — we set that a few exercises ago.

Hint

The format of an automatic property with get and protected set is:

public bool IsFake
{ get; protected set; }

2. In **Bicycle.cs**, create an empty Bicycle class that inherits Vehicle.



**3.** Define a constructor that:

has one double parameter for setting the Speed property calls base() with that parameter in its body, sets Wheels to 2

Hint

Here's an example of another class calling its base constructor:

class Dog : Animal

public Dog(int age) : base(age)
{
 }
}

4.

Define a public void SpeedUp() method that limits the Speed to 15. In other words, in the method body:

Add 5 to Speed

If Speed is greater than 15, set it to 15

An error might occur here, which is okay.

5.

You probably saw the warning:

warning CS0108: 'Bicycle.SpeedUp()' hides inherited member 'Vehicle.SpeedUp()'. Use the new keyword if hiding was intended.

The computer suggests using the new approach, but we prefer override for its clarity.

In Bicycle.cs, label SpeedUp() with override.

Hint  $\vee$ 

The override keyword belongs in the first line of the method, like:

public override void FakeIt

Some compilers will show a similar warning that mentions both override and new:

warning CS0114: 'Bicycle.SpeedUp()' hides inherited member 'Vehicle.SpeedUp()'. To make the current member override that implementation, add the override keyword. Otherwise add the new keyword.

**6.** Now your probably saw the error:

error (\$05005; Bicycle, SpeedUp() is cannot override inherited member

Vehicle. SpeedUp() because it is not marked virtual, abstract, or override

In Vehicle.cs, label SpeedUp() with virtual.

Hint

The virtual keyword belongs in the first line of the method, like:

public virtual void FakeIt()

7.
Repeat the process with SlowDown() in Bicycle.cs (let's assume that only sedans and trucks can go in reverse). It should override the inherited version and limit the Speed to 0. In

other words, the method:

Subtracts 5 from Speed

Sets it to 0 if Speed is less than 0

Is labeled override

8. In **Vehicle.cs**, label SlowDown() with virtual.