## **Make Inherited Members Abstract**

Now we want to add one more method to <code>Vehicle</code> called <code>Describe()</code>. It will be different for every subclass, so there's no point in defining a default one in <code>Vehicle</code>. Regardless, we want to make sure that it is implemented in each subclass.

This might sound similar to an interface. Why not add this method to the <code>IAutomobile</code> interface? We want <code>Describe()</code> to be available to all vehicles, not just automobiles.

To do this we need one more modifier: abstract. This line would go into the Vehicle class:

## public abstract string Describe()

This is like the Vehicle class telling its subclasses: "If you inherit from me, you must define a Describe() method because I won't be giving you any default functionality to inherit." In other words, abstract member have no implementation in the superclass, but they must be implemented in all subclasses.

If one member of a class is abstract, then the class itself can't really exist as an instance. Imagine calling <code>Vehicle.Describe()</code>. It doesn't make sense because it doesn't exist! This means that the entire Vehicle class must be abstract. Label it with <code>abstract</code> as well:

#### abstract class Vehicle

If you don't do this, you'll get an error message like this:

# error CS0513: 'Vehicle.Describe()' is abstract but it is contained in non-abstract class 'Vehicle'

Once we write the abstract method and mark the class as abstract, we'll need to actually implement it in each subclass. For example in Sedan:

```
public override string Describe()
{
  return $"This Sedan is moving on {Wheels} wheels at {Speed} km/h, with
  license plate {LicensePlate}.";
}
```

To make it clear that this <code>Describe()</code> method in <code>Sedan</code> is overriding the <code>Describe()</code> method in <code>Vehicle</code>, we will need label it <code>override</code>.

### Instructions

1.

Add the abstract method Describe() to the Vehicle class.

Describe() should be public and return a string

Vehicle will also need to be labeled abstract

You might see an error after this.

Hint ~

Here's an example abstract method. Make sure to include a semicolon (;) at the end:

public abstract int FakeIt();

2. You probably saw this error:

error CS0534: 'Bicycle' does not implement inherited abstract member 'Vehicle.Describe()'

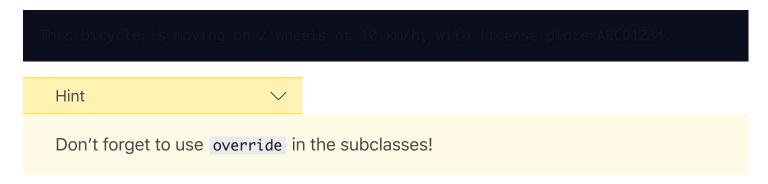
Fix this by implementing <code>Describe()</code> methods in <code>Bicycle</code>, <code>Sedan</code>, and <code>Truck</code>. Each method should:

mention the type, e.g. the bicycle version of the method returns a string containing "bicycle"

mention the license plate, speed, and wheels

be labeled with override

For bicycles, the returned string might look like this:



3. In **Program.cs**, call **Describe** on each instance and print the result to the console.