## **Build an Interface**

For this lesson we will be designing a new set of transportation machines that satisfy the requirements of BOTH car designers and the highway patrol. First the highway patrol tells us: "Every automobile on the road must have these properties and methods accessible to us:"

speed
license plate number
number of wheels
ability to honk

The patrol needs this information to write speeding tickets and prevent bad behavior on the highway.

In other words, the patrol makes these requirements so that it can interact with automobiles in a certain way. In C#, this group of interactions is called an *interface*. The interface is a set of properties, methods, and other members. They are declared with a signature but their behaviors are not defined. A class *implements* an interface if it defines those properties, methods, and other members.

For example, if the patrol requires automobiles to have a license plate, then the IAutomobile interface contains a LicensePlate property. A class implements this interface if it defines a LicensePlate property.

The skeleton of an interface looks a bit like a class:

## interface TAutomobile

Every interface should have a name starting with "I". This is a useful reminder to other developers and our future selves that this is an interface, not a class. We can add members, like properties and methods, to the interface. Here's an example of a fake property and method:

```
interface IAutomobile
{
   string Id { get; }
   void Vroom();
}
```

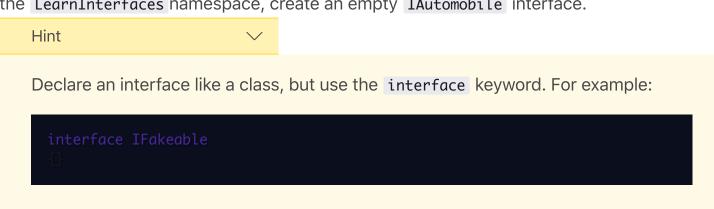
Notice that the property and method bodies are not defined. An interface is a set of actions and values, but it doesn't specify how they work.

In our highway example, the highway patrol doesn't care HOW the license plate property and honk method work, they just care whether every automobile has it.

## **✓**Instructions

1.

Just like classes, interfaces are best organized in their own files. In **IAutomobile.cs** within the **LearnInterfaces** namespace, create an empty **IAutomobile** interface.



**2.** Add these three properties and one method to the interface:

```
a string called LicensePlate
a double called Speed
an int called Wheels
a void method called Honk()
```

The properties only need a getter. Use the <code>get</code> shorthand demonstrated in the narrative for <code>Id</code>.

