What Interfaces Cannot Do

The `Sedan` needs to satisfy more than the highway patrol's rules (the `IAutomobile` interface). The car designers have asked that sedans are built and move in certain ways — it must have constructors and methods that aren't required by the `IAutomobile` interface. This is okay in C#! The interface says what a class MUST have. It does not say what a class MUST NOT have.

In fact, interfaces cannot specify two types of members that are commonly found in classes:

- Constructors
- Fields

☑**Instructions**

1.

Add a constructor to the `Sedan` class with one parameter, `speed`, of type `double`. It should

- set the `Speed` property to `speed`
- set a random `LicensePlate` value
- set `Wheels` to 4

To make a random license plate, a utility class is provided for you. Use it in the constructor like so: `Tools.GenerateLicensePlate()`.

Hint

Remember that a constructor looks like a method, but there is no return type listed and the method name is the name of its enclosing class:

```
class Forest
{
  public int Area;

  public Forest(int area)
  {
    Area = area;
  }
}
```

2.
Add a `void SpeedUp()` method that increases the `Speed` property by `5`.

Hint

Here's an example method, which increases the `Area` property:

```
public void IncreaseArea(int growth)
{
  Area = Area + growth;
}
```

3.
Did you get an error? There is no setter for the `Speed` property. Add a private setter to that property.

Hint

Here's an example of a property with get and set methods:

```
bool IsFake
{ get; set; }
```

Without a setter, the `SpeedUp()` method won’t be able to access `Speed`, and you’ll get an error like this:

```
error CS0200: Property or indexer 'Sedan.Speed' cannot be assigned to --
it is read only
```

4.
Add a `void SlowDown()` method that decreases the `Speed` by `5`.