The purpose of this project demo is to better understand the utility of interfaces.

In this instance, we're assigning our interface 'ITakeDamage' as an extension of our two MonoBehaviour classes 'Car' and 'Barrel'

Then, the Car and Barrel class each **implement** the TakeDamage() method created in out interface. **And the key thing to note here is that they implement the same method in different ways: this is the power and flexibility that an interface provides .**

In the DamageOnClick script, we create a variable ‘damageable’ to get the component ‘ITakeDamage’ from any collider that contains the interface.

Since both our classes ‘Barrel’ and ‘Car’ inherit from our interface and determine their own implementation for the method created within the interface, they will contain an ITakeDamage component.

We then check to see if ‘damageable’ != null, and if it isn’t, we call our newly created instance of ITakeDamage - ‘damageable’ and set TakeDamage = 1