What is Inheritance?

In this lesson, you will be introduced to Inheritance, a powerful concept in object-oriented programming.

WE'LL COVER THE FOLLOWING

Now that you are familiar with the concepts of *objects* and *classes*, let's discuss **inheritance** which is another key concept in *object-oriented programming*.

Definition

Inheritance provides a way to create a new class from an existing class. The new class is a specialized version of the existing class such that it inherits all the *non-private* fields (*variables*) and *methods* of the existing class. The existing class is used as a starting point or as a *base* to create the new class.

The *IS A* Relationship

After reading the above definition, the next question that comes to mind is, *when do we use inheritance?* Wherever we come across an ***IS A*** relationship between objects, we can use inheritance.

Square
C#
Soda
IS A
IS A
IS A

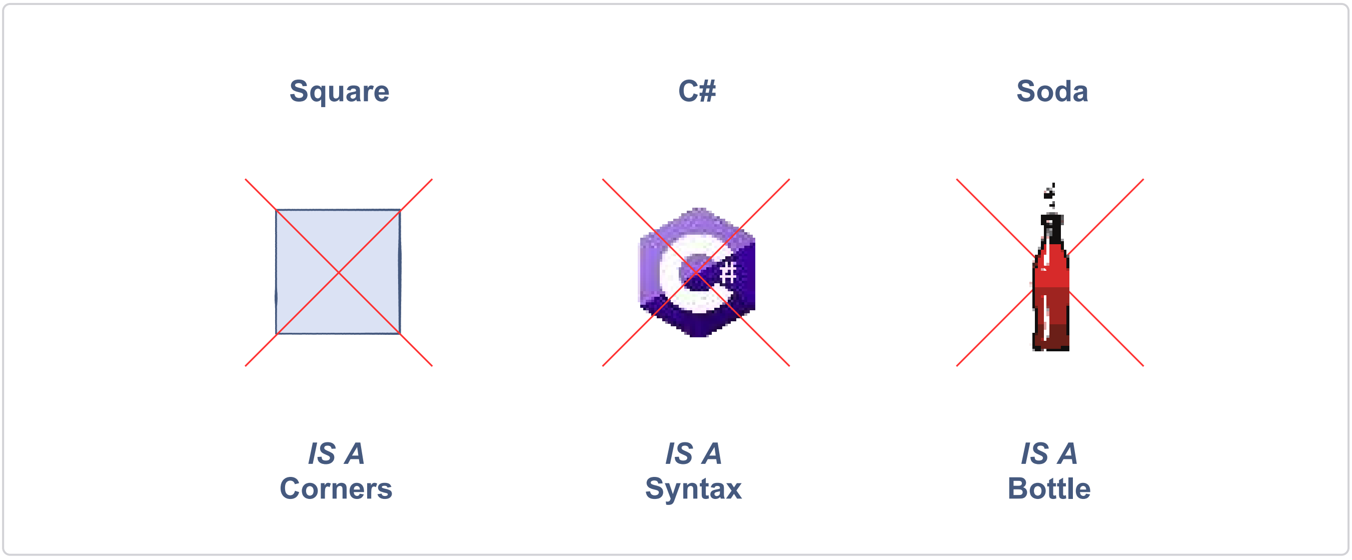
In the above illustration, we can see there are three classes having an ***IS A*** relationship between them. We can write it as:

- Square ***IS A*** shape
- C# ***IS A*** programming language
- Soda ***IS A*** beverage

From the above descriptions regarding *inheritance,* we conclude that we can build new classes by extending on the *existing classes*. The new classes extend the existing one in certain ways. For example, a soda *is a* beverage, but on top of the characteristics of any beverage, it adds fizz. Let's have a look at some of the classes which can be derived using the `Shape`, `ProgrammingLanguage` and `Beverages` classes:

Existing Class	Derived Classes
Shape	Square, Circle, Triangle
Programming Language	C#, Java, Python
Beverage	Soda, Beer, Wine

Let's find out where an ***IS A*** relationship doesn't exist.



In the above illustration, it’s obvious that we cannot use *inheritance* since an ***IS A*** relationship doesn’t exist between the objects.

The C# Object class

Let’s have a look at a beautiful example of inheritance that comes pre-implemented in the **.NET framework**. When working with the .NET framework using C#, whenever we create a `class`, it directly or indirectly inherits all the *non-private methods* and *fields* from the built-in C# class named `Object`. The methods defined in the `Object` class come in very handy when you create *new classes*. To find out more about the C# `Object` class and its functionalities, you can visit here.

Are things getting interesting? Let’s move to the next lesson in which we will discuss the syntax and terminologies related to inheritance.