Now that you have got familiar with the concepts of *objects* and *classes*, let's discuss **inheritance** which is another key concept in the *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 your mind is *What is the use case of inheritance?* Well, the answer is that wherever we come across an ***IS A*** relationship between objects, we can use inheritance.

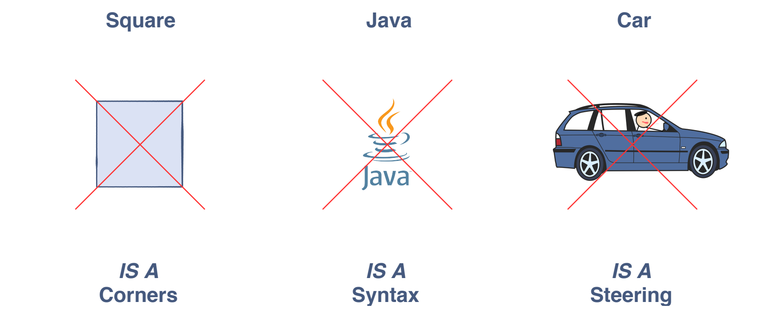
In the above illustration, we can see there are three objects having an ***IS A*** relationship between them. We can write it as:

- A square ***IS A*** shape
- Java ***IS A*** programming language
- Car ***IS A*** vehicle

So from the above descriptions regarding *inheritance,* we can conclude that we can build new classes by depending on the *existing classes*. We can build some new *classes*.

Existing Class	Derived Class
Shape	Square
Programming Language	Java
Vehicle	Car

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



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

The Java Object class

The basic purpose of object-oriented programming is to enable a programmer to model the *real world objects* using a programming language. In Java whenever we create a `class`, it inherits all the `non-private` *methods* and *fields* from the builtin Java `Object class` by default which makes it a very good example of inheritance in Java. The methods defined in the `Object class` come in very handy when you create *new classes*. To find out more about the Java Object class and its functionalities, you can visit here.