# PHP – Inheritance

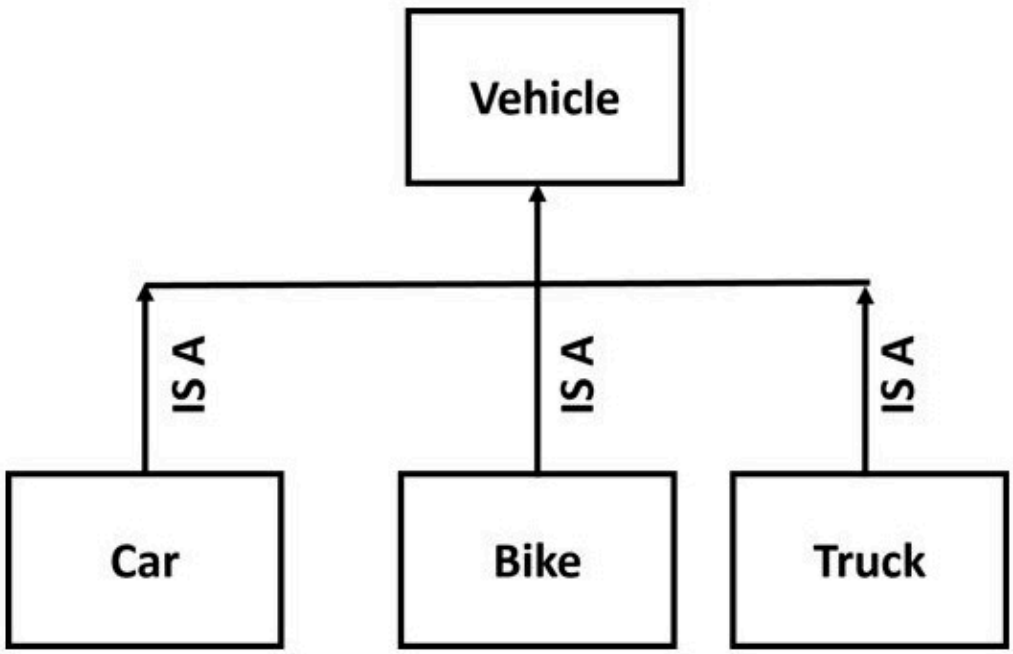
Inheritance is one of the fundamental principles of object-oriented programming methodology. Inheritance is a software modelling approach that enables extending the capability of an existing class to build new class instead of building from scratch.

PHP provides all the functionality to implement inheritance in its object model. Incorporating inheritance in PHP software development results in code reuse, remove redundant code duplication and logical organization.

Imagine that you need to design a new class whose most of the functionality already well defined in an existing class. Inheritance lets you to extend the existing class, add or remove its features and develop a new class. In fact, PHP has the "extends" keyword to establish inheritance relationship between existing and new classes.

```
class newclass extends oldclass {
   ...
   ...
}
```

Inheritance comes into picture when a new class (henceforth will be called inherited class, sub class, child class, etc.) possesses "IS A" relationship with an existing class (which will be called base class, super class, parent class, etc.).




In PHP, when a new class is defined by extending another class, the subclass inherits the public and protected methods, properties and constants from the parent class. You are

free to override the functionality of an inherited method, otherwise it will retain its functionality as defined in the parent class.

## Example

Take a look at the following example –

</> Open Compiler

```
<?php
   class myclass {
      public function hello() {
         echo "Hello from the parent class" . PHP_EOL;
      }
      public  function thanks() {
         echo "Thank you from parent class" . PHP_EOL;
      }
   }
   class newclass extends myclass {
      public function thanks() {
         echo "Thank you from the child class" . PHP_EOL;
      }
   }

   # object of parent class
   $obj1 = new myclass;
   $obj1->hello();
   $obj1->thanks();

   # object of child class
   $obj2 = new newclass;
   $obj2->hello();
   $obj2->thanks();
?>
```

It will produce the following **output** –

```
Hello from the parent class
Thank you from parent class
Hello from the parent class
Thank you from the child class
```

As mentioned before, the child class inherits public and protected members (properties and methods) of the parent. The child class may introduce additional properties or methods.

In the following example, we use the **Book class** as the parent class. Here, we create an **ebook class** that extends the Book class. The new class has an additional property – **format** (indicating ebook's file format – EPUB, PDF, MOBI etc). The ebook class defines two new methods to initialize and output the ebbok data – **getebook()** and **dispebook()** respectively.

## Example

The complete code of inheritance example is given below –

</> Open Compiler

```
<?php
   class Book {

      /* Member variables */
      protected int $price;
      protected string $title;

      public function getbook(string $param1, int $param2) {
         $this->title = $param1;
         $this->price = $param2;
      }
      public function dispbook() {
         echo "Title: $this->title Price: $this->price \n";
      }
   }

   class ebook extends Book {
      private string $format;
      public function getebook(string $param1, int $param2, string $param3) {
         $this->title = $param1;
         $this->price = $param2;
         $this->format = $param3;
      }
      public function dispebook() {
         echo "Title: $this->title Price: $this->price\n";
         echo "Format: $this->format \n";
      }
```

```
    }
    $eb = new ebook;
    $eb->getebook("PHP Fundamentals", 450, "EPUB");
    $eb->dispebook();
?>
```

The browser **output** is as shown below –

```
Title: PHP Fundamentals Price: 450
Format: EPUB
```

If you take a closer look at the getebook() function, the first two assignment statements are in fact there getbook() function, which the ebook class has inherited. Hence, we can call it with parent keyword and scope resolution operator.

Change the getebook() function code with the following –

```
public function getebook(string $param1, int $param2, string $param3) {
    parent::getbook($param1, $param2);
    $this->format = $param3;
}
```

Similarly, the first echo statement in dispebook() function is replaced by a call to the dispbook() function in parent class –

```
public function dispebook() {
    parent::dispbook();
    echo "Format: $this->format<br/>";
}
```

Explore our latest online courses and learn new skills at your own pace. Enroll and become a certified expert to boost your career.

## Constructor in Inheritance

The constructor in the parent class constructor is inherited by the child class but it cannot be directly called in the child class if the child class defines a constructor.

In order to run a parent constructor, a call to **parent::__construct()** within the child constructor is required.

## Example

Take a look at the following example –

```php
<?php
   class myclass{
      public function __construct(){
         echo "This is parent constructor". PHP_EOL;
      }
   }
   class newclass extends myclass {
      public function __construct(){
         parent::__construct();
         echo "This is child class destructor" . PHP_EOL;
      }
   }
   $obj = new newclass();
?>
```

It will produce the following **output** –

```
This is parent constructor
This is child class destructor
```

However, if the child does not have a constructor, then it may be inherited from the parent class just like a normal class method (if it was not declared as private).

## Example

Take a look at the following example –

```php
<?php
   class myclass{
      public function __construct(){
         echo "This is parent constructor". PHP_EOL;
      }
   }
```

```
   class newclass extends myclass{ }
   $obj = new newclass();
?>
```

It will produce the following **output** –

```
This is parent constructor
```

PHP doesn’t allow developing a class by extending more than one parents. You can have **hierarchical inheritance**, wherein class B extends class A, class C extends class B, and so on. But PHP doesn’t support **multiple inheritance** where class C tries to extend both class A and class B. We can however extend one class and implement one or more **interfaces**. We shall learn about interfaces in one of the subsequent chapters.