**Static Methods and Properties in PHP**

by [ANKUR KUMAR SINGH](http://www.techflirt.com/author/admin/) on MARCH 24, 2013 · [7 COMMENTS](http://www.techflirt.com/tutorials/oop-in-php/static-methods-and-properties-in-php.html#comments) · in [PHP](http://www.techflirt.com/category/php/)

Static [methods[http://cdncache-a.akamaihd.net/items/it/img/arrow-10x10.png](http://www.techflirt.com/tutorials/oop-in-php/static-methods-and-properties-in-php.html)](http://www.techflirt.com/tutorials/oop-in-php/static-methods-and-properties-in-php.html) and properties in php is very useful feature. Static methods and properties in php can directly accessible without creating object of class. Your php class will be static class if your all methods and properties of the class is static. [**Static[http://cdncache-a.akamaihd.net/items/it/img/arrow-10x10.png](http://www.techflirt.com/tutorials/oop-in-php/static-methods-and-properties-in-php.html)**](http://www.techflirt.com/tutorials/oop-in-php/static-methods-and-properties-in-php.html)**Methods and Properties in PHP will be treated as public if no visibility is defined**.

**Static Properties/Variables in PHP**

Static properties of class is a property which is directly accessible from class with the help of **::**(scope resolution operator). You can declare static  property using **static** keyword. In other word you can make any property [static[http://cdncache-a.akamaihd.net/items/it/img/arrow-10x10.png](http://www.techflirt.com/tutorials/oop-in-php/static-methods-and-properties-in-php.html)](http://www.techflirt.com/tutorials/oop-in-php/static-methods-and-properties-in-php.html) by using static keyword. following is the basic example of static variable in php class:

class test  
{  
public static $a;//Static variable  
}  
test::$a = 5;  
echo test::$a;

You can not access regular property by static way. **It will generate fatal error**. For with in the class you can access static property using **self** keyword. If you are accessing parent class property then you need to use parent keyword.  
class testParent  
{  
public static $var1;  
}  
class testChild extends testParent  
{  
public static $var2;  
public $abc =2;  
function testFunction()  
{  
**self**::$var2 = 3;  
**parent**::$var1 = 5;  
}  
}  
echo testChild::$abc; //throw fatal error

Static variable or property are the best way to preserver value of the variable within the context of different instance. Please go through following example for better expatiation:

class test  
{  
private static $no\_of\_call = 0;  
public function \_\_construct()  
{  
self::$no\_of\_call = self::$no\_of\_call + 1;  
echo "No of time object of the class created is: ". self::$no\_of\_call;  
}  
}  
$objT = new test(); // Prints No of time object of the class created is 1  
$objT2 = new test(); //Prints No of time object of the class created is 2

So creating static variable or property is very useful if you want to share some data between the different object of the same class. We will get better example of the static property implementation in chapter PHP Design Patterns.

**Static Methods or functions**

As in general class various process are same for methods and properties, **same is with Static Methods and Properties in PHP**. You can create your function or method static using **static** keyword. You can access all visible static methods for you using **::** like in static variables.  
class test  
{  
**static** function abc($param1 , $param2)  
{  
echo "$param1 , $param2";  
}  
}  
**test::abc("ankur" , "techflirt");**

If you will use regular or normal method statically then you will get E\_STRICT warning. In case of variable or property it was throwing fatal. Let us take above example  
class test  
{  
function abc($param1 , $param2)  
{  
echo "$param1 , $param2";  
}  
}  
test::abc("ankur" , "techflirt"); //will work fine with warning.

Since static methods is called direct $this variable will not available in the method.

[**Download Code**](http://www.techflirt.com/download/oop/static_methods.zip) for static methods in PHP.

For further details about static methods and properties in php you can go to:  
<http://php.net/manual/en/language.oop5.static.php>  
<http://en.wikipedia.org/wiki/Static_method#Static_methods>

**Inheritance in PHP**

by [ANKUR KUMAR SINGH](http://www.techflirt.com/author/admin/) on MARCH 24, 2013 · [6 COMMENTS](http://www.techflirt.com/tutorials/oop-in-php/inheritance-in-php.html#comments) · in [PHP](http://www.techflirt.com/category/php/)

Inheritance is a concept in object oriented programming. With the help of inheritance we can get all property and method of one class in another class. This is principle to take re-fusibility on upper level.  Inheritance in php is introduced from php5 version.

In this chapter we will explore about basics concept of inheritance  After basic we will discuss implementation of inheritance in php. This tutorial for the beginner who want to learn basic concept of inheritance in php. Before going further I am assuming that have some idea of oop in php. Later in this chapter we will cover some advance aspect of the inheritance also.

**What is inheritance?**

Inheritance is nothing but a design principle in oop. By implementing inheritance you can inherit(or get) all properties and methods of one class to another class. The class who inherit feature of another class known as **child class**.The class which is being inherited is know as **parent class**. Concept of the inheritance in oop is same as inheritance in real world. For example, child inherits characteristics  of their parent. Same is here in oop. One class is inheriting characteristics of another class.

With the help of inheritance you can increase re-usability of code. Let us take an example in terms of generic programming practices. Suppose you are going to create classes to render different html tag(div, span, form, table etc).  Now you will create class with name html\_div, html\_span ,  html\_form. You are creating different class because every element is different in nature. For example form has action and method and you will have different input element in form. But table will have tbody, tr, th and td.

Now just think for some moment. There are some element and their rendering is same in all element. For example all html mention above is having name, id, class attribute which is same. Also rendering of those element is also same. So in above case you can create parent class with name HTML and you can inherit that class across all of your classes like div, span, form. Following is the generic code structure of inheritance in oop taking your HTML attribute in consideration. I am taking php syntax for better understanding  
class HTML  
{  
protected $name;  
protected $id;  
protected function basicAttribute  
{  
return "name='$this->name' id='$this->id'";  
}  
}  
Class HTML\_div **extends** HTML  
{  
public function \_\_construct($id , $name)  
{  
$this->id = $id;  
$this->name = $name;  
}  
public function getDiv($content)  
{  
$basicAttribute = $this->basicAttribute();  
return "<div $basicAttribute >$content</div>"  
}  
}  
Class HTML\_span **extends** HTML  
{  
public function \_\_construct($id , $name)  
{  
$this->id = $id;  
$this->name = $name;  
}  
public function getSpan($content)  
{  
$basicAttribute = $this->basicAttribute();  
return "<span $basicAttribute >$content</span>"  
}  
}

Above code is and example of basic inheritance in php. All method(protected and public) from HTML class is directly accessible in your class HTML\_div and HTML\_span class. In both child classes you no need to write rendering of id and name logic again and again. This really saves time and give some good modulations in the code.  
Hope your basic understanding about inheritance is clear. Now let us move to implementation of inheritance in php.

**Inheritance in php**

Concept of inheritance in php is as simple as in other oop languages as from php5 community target is to provide healthy oop concept. If you will analyze basic code of my previous topic, this is typical example of inheritance in php. To implementing inheritance in php you need at least 2 classes. One will be parent class and other will be child class. In child class you can inherit all properties and methods(protected and public only) from parent class. You can implement inheritance in php using keyword extends. Let us take above example again with some modification:  
class HTML  
{  
**protected** $name;  
**public** $id;  
**private** $with;  
**protected** function basicAttribute  
{  
return "name='$this->name' id='$this->id'";  
}  
}  
Class HTML\_div **extends** HTML  
{  
**public** function \_\_construct($id , $name)  
{  
$this->id = $id;  
$this->name = $name;  
}  
**public** function getDiv($content)  
{  
$basicAttribute = $this->basicAttribute();  
return "<div $basicAttribute >$content</div>"  
}  
}'  
$objDiv = new HTML\_div("bloc\_main" , 'avc');  
$objDiv->getDiv('this is and example of inheritance in php');  
Now in above code class HTML\_div is inheriting property and method from class HTML.

**Multilevel and Multiple inheritance in PHP**

In php multilevel inheritance is possible but multiple inheritance is not possible. In simplified terms in php child class can not inherit more than one parent class. But hierarchical inheritance is possible in php. Hierarchical means Parent inherit property of grand parent class. Grand child inherit property of parent class. So in multilevel inheritance child can get some property of from grand parent class also.

**Example of Multiple inheritance in PHP**  
class test  
{  
//Your class body  
}  
class test1  
{  
//Your class body  
}  
class test3 extends test1 test2  
{  
//your class body  
}   
Above code will not work in php. Because php is single inheritance language.

**Example of Multilevel inheritance in PHP**class grandParent  
{  
 //Body of your class  
}

class [parent[http://cdncache-a.akamaihd.net/items/it/img/arrow-10x10.png](http://www.techflirt.com/tutorials/oop-in-php/inheritance-in-php.html)](http://www.techflirt.com/tutorials/oop-in-php/inheritance-in-php.html) extends grandParent  
{  
 //Body Of your class  
}

class child extends parent  
{  
//Body of your class  
}  
This is very basic example of multilevel [inheritance[http://cdncache-a.akamaihd.net/items/it/img/arrow-10x10.png](http://www.techflirt.com/tutorials/oop-in-php/inheritance-in-php.html)](http://www.techflirt.com/tutorials/oop-in-php/inheritance-in-php.html). In php it is possible to implement multilevel inheritance. In above example parent class is inheriting grand parent property. And and child is inheriting parent property. So child have some parent and grand parent property.

**Static Methods and Property in Inheritance in PHP**

As in our example of HTML\_div class we have explored that we can use **$this->** keyword to get all property and method of parent(HTML) class. But if your parent or child method is static, then you can access static methods or properties using **self** and **parent** keyword. Also this is not necessary to make method static if you want to use self or parent keyword. This is very useful if your parent and child both method is having property or method with same name. If both [classes[http://cdncache-a.akamaihd.net/items/it/img/arrow-10x10.png](http://www.techflirt.com/tutorials/oop-in-php/inheritance-in-php.html)](http://www.techflirt.com/tutorials/oop-in-php/inheritance-in-php.html) having same property and you want to call specific property or method then you can use this keyword.

**Self and parent in case of static methods**:  
class parent  
{  
public static abc()  
{  
//your function body  
}  
}  
class child  
{  
public static xyz()  
{  
//your function body  
}  
function callStatic()  
{  
self::xyz();  
parent::abc();  
}  
}

Self and Parent without static  
class parent  
{  
protected function xyz()  
{  
//Your function body  
}  
}  
class child extends parent  
{  
public function xyz()  
{  
//your function body  
}  
public function calll()  
{  
self::xyz();  
parent::xyz();  
}  
}

[**Download Code**](http://www.techflirt.com/download/oop/inheritance_php.zip) for Inheritance in PHP

Further details about inheritance in PHP you can read:  
<http://php.net/manual/en/language.oop5.inheritance.php>  
<http://en.wikipedia.org/wiki/Inheritance_%28object-oriented_programming%29>

Next >> [**Abstract Classes and Interface**](http://www.techflirt.com/tutorials/oop-in-php/abstract-classes-interface.html)

**Abstract Classes and Interface in PHP**

by [ANKUR KUMAR SINGH](http://www.techflirt.com/author/admin/) on MARCH 24, 2013 · [39 COMMENTS](http://www.techflirt.com/tutorials/oop-in-php/abstract-classes-interface.html#comments) · in [PHP](http://www.techflirt.com/category/php/)

Abstract class and Interface in php play very important role in oop. In this section we will discuss following point

1. What is abstract classes.
2. What is interface
3. How to implement abstract classes in php
4. How to implement interface in php
5. Different between abstract classes and interface.

**What is abstract Classes**

As from name it seem like something that is hidden. Yes nature of the abstract classes are same. Abstract classes are those classes which can not be directly initialized. Or in other word we can say that you can not create object of abstract classes. Abstract classes always created for inheritance purpose. You can only inherit abstract class in your child class. Lots of people say that in abstract class at least your one method should be abstract. **Abstract method are the method which is only defined but declared**. This is not true definition as per my assumption. But your any class has at least one method abstract than your class is abstract class.

Usually abstract class are also known as base class. We call it base class because abstract class are not the class which is available directly for creating object. It can only act as parent class of any normal class. You can use abstract class in class hierarchy. Mean one abstract class can inherit another abstract class also.

**Abstract classes in PHP**

Abstract classes in php are simillar like other oop languages. You can create abstract classes in php using **abstract** keyword. Once you will make any class abstract in php you can not create object of that class.  
abstract class abc  
{  
public function xyz()  
{  
return 1;  
}  
}  
$a = new abc();//this will throw error in php  
above code will throw error in php.

Abstract classes in php are only for inheriting in other class.  
abstract class testParent  
{  
public function abc()  
{  
//body of your funciton  
}  
}  
class testChild extends testParent  
{  
public function xyz()  
{  
//body of your function  
}  
}  
$a = new testChild();  
In above example you are creating of testChild Class. TestChild class is inheriting testParent abstract class. So your abstract class is only available for inheritance. Main motive of creating abstract classes in php is to apply restriction of direct initialization or object creation.

**Implementation of abstract method**

As we know that abstract functions are those functions of abstract class which is only defined. It will be declared in your child class. You can create any method abstract using keyword **abstract**. You can only create abstract method either in abstract class or interface. Following is example of the abstract method implementation:

abstract class abc  
{  
abstract protected function f1($a , $b);  
}  
class xyz extends abc  
{  
protected function f1($name , $address)  
{  
echo "$name , $address";  
}  
}  
$a = new xyz();

In class abc we have defined an abstract function f1. Now when we have inherited class abc then declared function f1. **If you have an abstract method in your abstract class then once you inherit your abstract class then it is necessary to declare your abstract method. If you will not declare your abstract method then PHP will throw error in that case**.

You can declare your abstract method in child class with the same visibility or less restricted visibility.

abstract class parentTest  
{  
abstract protected function f1();  
abstract public function f2();  
//abstract private function f3(); //this will trhow error  
}

class childTest extends parentTest  
{  
public function f1()  
{  
 //body of your function  
}  
public function f2()  
{  
 //body of your function  
}  
protected function f3()  
{  
//body of your function  
}  
}  
$a = new childTest();

In above code you can see that you have declare 3 function in abstract class. But private declaration of the abstract method will always throw error. Because private method is available only in the same class context. But in case of f1. This is protected. Now in child class we have defined it as public because public is less restricted than protected. And for function f2 which is already public so we have defined it as public in our child class. We have defined it public because **no any visibility is less restricted than public**.

**What is Interface ?**

Interface in oop enforce definition of some set of method in the class. By implementing interface you are forcing any class to must declaring some specific set of methods in oop. For example if you are creating class to render HTML element then it is necessary to set id and name of your html tag. So in this case you will create interface for that class and define method like setID and setName. So whenever someone will create any class to render HTML tag and implemented your interface then he must need to define setId and setName method in their class. In other word you can say that by help of interface you can set some definition of your object. Interface is very useful if you are creating architecture of any oop base application. Inter

**Interface in PHP**

Interface in php can be implemented like other oop lanugage. You can create interface in php using keyword interface. By implementation of interface in php class you are specifying set of the method which classes must implement.

You can create interface in php using interface keyword. Rest of the things are typically identical to classes. Following is very small example of interface in php.  
**interface** abc  
{  
public function xyz($b);  
}  
So in above code you are creating interface with name abc. Interface abc has function xyz. Whenever you will implement abc interface in your class then you have to create method with name xyz. If you will not create function xyz then it will throw error.

You can implement your interface in your class using **implements** keyword. Let us implement our interface abc in our class  
class test **implements** abc  
{  
public function xyz($b)  
{  
//your function body  
}  
}

You can only define method in interface with public accessibility. If you will use other than public visibility in interface then it will throw error. Also while defining method in your interface do not use**abstract** keyword in your methods.

You can also extend interface like class. You can extend interface in php using **extends** keyword.  
interface template1  
{  
public function f1();  
}

interface template2 extends template1  
{  
public function f2();  
}

class abc implements template2  
{  
public function f1()  
{  
//Your function body  
}  
public function f2()  
{  
//your function body  
}  
}  
So here template2 has all property of tempate2. So whenever you will implement template2 in your class, you have to create function of both interfaces.

You can also extend multiple interface in one interface in php.  
interface template1  
{  
public function f1();  
}  
interface template2  
{  
public function f2();  
}  
interface template3 extends template1, template2  
{  
public function f3();  
}  
class test implements template3  
{  
public function f1()  
{  
//your function body  
}  
public function f2()  
{  
//your function body  
}  
public function f3()  
{  
//your function body  
}  
}

You can also implement more than one interface in php class.  
interface template1  
{  
public function f1();  
}  
interface template2  
{  
public function f2();  
}  
class test implments template1, template2  
{  
public function f1()  
{  
//your function body  
}  
public function f2()  
{  
//your function body  
}  
}  
**You can not implement 2 interfaces if both share function with same name. It will throw error**.

Your function parameter in class must be identical to the parameter in the interface signature. Following is example some example  
interface template1  
{  
public function f1($a)  
}  
class test implements template1  
{  
public function f1($a)  
{  
echo $a;  
}  
}  
Above will work. But following example will not work:  
interface template1  
{  
public function f1($a)  
}  
class test implements template1  
{  
public function f1()  
{  
echo $a;  
}  
}  
But it is not necessary to use the same name of the variable. Like $a. You can also use any name. For example:  
interface template1  
{  
public function f1($a)  
}  
class test implements template1  
{  
public function f1($name)  
{  
echo $name;  
}  
}  
If you are using default argument then you can change your value of the argument. For example  
interface template1  
{  
public function f1($a = 20)  
}  
class test implements template1  
{  
public function f1($name  = "ankur")  
{  
echo $name;  
}  
}

In above section we have discussed interfaces and abstract classes in php. Both are almost doing same things but has some difference.

**Differences between abstract class and interface in PHP**

Following are some main difference between abstract classes and interface in php

1. In abstract classes this is not necessary that every method should be abstract. But in interface every method is abstract.
2. Multiple and multilevel both type of inheritance is possible in interface. But single and multilevel inheritance is possible in abstract classes.
3. Method of php interface must be public only. Method in abstract class in php could be public or protected both.
4. In abstract class you can define as well as declare methods. But in interface you can only defined your methods.

For more information about abstract classes and Interface in PHP you can read :  
<http://en.wikipedia.org/wiki/Interface_%28computing%29>  
<http://php.net/manual/en/language.oop5.interfaces.php>  
<http://php.net/manual/en/language.oop5.abstract.php>

Next >> [**Overloading and Overriding in PHP**](http://www.techflirt.com/tutorials/oop-in-php/overloading-and-overriding.html)

**Overloading and Overriding in PHP**

by [ANKUR KUMAR SINGH](http://www.techflirt.com/author/admin/) on MARCH 24, 2013 · [15 COMMENTS](http://www.techflirt.com/tutorials/oop-in-php/overloading-and-overriding.html#comments) · in [PHP](http://www.techflirt.com/category/php/)

Function or method Overloading and overriding method is very basic and useful feature of any oop language. In this tutorial we will discuss implementation of method overloading and overriding in php.  Here first we will explore basics of overloading and overriding. After exploration of basics we will implement overloading and overriding in php. Before going further I am assuming that you have basic knowledge of classes and inheritance in php. Also you have understanding about magic method in php. Magic method because overloading in php can be implmented using magic methods.

**What is Method Overriding in OOP ?**

Basic meaning of overriding in oop is same as real word meaning. In real word meaning of overriding  phenomena of replacing the same parental behavior in child. This is same in case of method overriding in oop. In oop meaning of overriding is to replace parent class method in child class. Or in simple technical word method overriding mean changing behavior of the method. In oop overriding is process by which you can re-declare your parent class method in child class. So basic meaning of overriding in oop is to change behavior of your parent class method.

Normally method overriding required when your parent class have some method, but in your child class you want the same method with different behavior. By overriding of method you can complete change its behavior from parent class. To implment method overiding in oop we commonly create same method in child class.

**What is Method Overloading in OOP ?**

Overloading in oop is same as overloading in real word. In real word overloading means assigning extra work to same machine or person. In oop method overloading is same. By process of method overloading you are asking your method to some extra work. Or in some cases we can say some different work also.

Normally method overloading in oop is managed on the basis of the argument passed in function. We can achieve overloading in oop by providing different argument in same function.

**Overloading and Overriding in PHP**

Hope your basic concept of overloading and overriding is clear now. Now let us explore implementation of overloading and overriding in php.

Implementation of overriding in php is very easy. If your parent class has a function. You can create function with same name in your child class to override the function. Implementation of overriding can not be achieved by creating 2 function with same name and different argument in php. Because we can not create same name function more than 1 time in php class. To implement overloading we need to take help of magic method in php. In below section we will explore overloading and overriding one by one.

**Overloading in PHP**

As we know that we can not implement overloading by create 2 function in with same name in class. So to implement overloading in php we will take help of magic method \_\_call. Magic method \_\_call invoked when method called by class object is not available in class. So here we will not create method exactly and will take help of \_\_call method. Now call method will provide us 2 argument, 1st name of the method called and parameter of the function. Now with the help of either switch , case or if else we will implement overloading in php. Following is very simple example of overloading in php.

class test  
{  
public function \_\_construct()  
{  
//Your logic for constructor  
}  
**public function \_\_call($method\_name , $parameter)**  
{  
if($method\_name == "overlodedFunction") //Function overloading logic for function name overlodedFunction  
{  
$count = count($parameter);  
switch($count)  
{  
case "1":  
//Business log in case of overlodedFunction function has 1 argument  
echo "You are passing 1 argument";  
break;  
case "2": //Incase of 2 parameter  
echo "You are passing 2 parameter";  
break;  
default:  
throw new exception("Bad argument");  
}  
}  
else  
{  
throw new exception("Function $method\_name does not exists ");  
}  
}  
}  
$a = new test();  
$a->overlodedFunction("ankur");  
$a->overlodedFunction("techflirt" , "ankur");

As in above class test magic method \_\_call is implemented which is managing overloading

public function \_\_call($method\_name , $parameter)  
{  
if($method\_name == "overlodedFunction") //Function overloading logic for function name overlodedFunction  
{  
$count = count($parameter);  
switch($count)  
{  
case "1":  
//Business log in case of overlodedFunction function has 1 argument  
echo "You are passing 1 argument";  
break;  
case "2": //Incase of 2 parameter  
echo "You are passing 2 parameter";  
break;  
default:  
throw new exception("Bad argument");  
}  
}  
else  
{  
throw new exception("Function $method\_name does not exists ");  
}  
}  
As we know that \_\_call magic method invoked when method is not available in the class. So in case of above test class example we have not created function overlodedFunction. So whenever method overlodedFunction is called \_\_call invoked. \_\_call pass 2 variable, first  name of the called method and other is parameter passed in the called function.

Now in the \_\_call function I have applied if [condition[http://cdncache-a.akamaihd.net/items/it/img/arrow-10x10.png](http://www.techflirt.com/tutorials/oop-in-php/overloading-and-overriding.html)](http://www.techflirt.com/tutorials/oop-in-php/overloading-and-overriding.html) to ensure that our business logic of overloading works only for overlodedFunction function. In if block we have counted number of argument in parameter and applied business logic.

**Overriding in PHP**

Overriding in php is very easy. As we know that overriding is [process[http://cdncache-a.akamaihd.net/items/it/img/arrow-10x10.png](http://www.techflirt.com/tutorials/oop-in-php/overloading-and-overriding.html)](http://www.techflirt.com/tutorials/oop-in-php/overloading-and-overriding.html) of modifying the inherited method. So in case of inheritance you only need to create method with same name in your child class which you want to override. Following is example of overriding of method in php.

class testParent  
{  
public function f1()  
{  
echo 1;  
}  
public function f2()  
{  
echo 2;  
}  
}  
class testChild  
{  
**function f2($a)** //overriding function f2  
{  
echo "$a";  
}  
}  
$a = new testChild();  
$a->f2("ankur");//it will print ankur

In above example you are overriding function f2. While overriding you are free to change business logic, visibility and number of parameter.

I hope your concept of the overloading and overriding in php is clear now.

For more detail about overloading and overriding in php you can read:  
<http://php.net/manual/en/language.oop5.overloading.php>  
<http://en.wikipedia.org/wiki/Function_overloading>  
<http://en.wikipedia.org/wiki/Method_overriding>

Next >> [Object Cloning In PHP](http://www.techflirt.com/tutorials/oop-in-php/object-cloning-in-php.html)

Overloading

The Methods with the same name but it differs by types of arguments and no, of arguments.

Overloading is Static Polymorphism.

Overloading refers to situation where there are two or more methods in a class with the same name but with different parameters list.

Overriding

The Methods with the same name and same no.of arguments and types, but one is in base class and second one is in derived class.

Overriding is dynamic Polymorphism.

Overriding refers to a situation where sub-class inherits a method from Base class which may result into adding or changing the method behavior

**What is Encapsulation?**   
Encapsulation is just when you want to hide stuffs into your object from the public. It's the idea of **hiding information.**For example you have an **Animal Class**, and it has a function called getName() then you've created a subclass of Animal called **Cow.**Now, when we used the function getName() it will return the name of the animal which is **Cow.**Now here's the problem **What if someone created an Animal Object?**Would you allow? Since it's a base class that some classes will inherit it's generic functions. you wouldn't want someone to create an object from this class right?

index.php  
animal.php  
cow.php  
  
**here's the content of animal.php**

<?php

class Animal {

public function getName($arg) {

echo "This is the ".$arg." Class";

}

}

?>

in this animal class we have 1 public function called getName() with an argument or parameter.

it's public so? You're right everything can accessed that function as long as I included that php file.

**for the cow.php**

<?php

class Cow extends Animal{

public function sayName() {

$this->getName(get\_class($this));

}

}

?>

in this I created an Animal class and we used the function **get\_class()** to get the name of the class  
notice that I've also it extends to the Animal class.. it means that this class is a subclass or Animal class so everything in the Animal class is shared in this class unless they're not private.  
  
**here's the content of index.php**

<?php  
include\_once('animal.php');

include\_once('cow.php');  
$cow = new Cow();  
$cow->sayName();

?>  
in this file, I've included animal.php and cow.php so we can use them both in this php file,

first I've created an object

$cow = new Cow();

then I called the sayName(); in the $cow object.. and the sayName() function calls the getName() function, we all know that getName() function is a function from the parent class of Cow which is Animal

so the output would be

Quote

This is the Cow Class

so what happens if we created an Animal object?

$animal = new Animal();

$animal->getName("blah");

output :

Quote

This is the blah Class

oooppsss that's bad since I only want that getName() function to be accessed from all the subclasses of Animal only.. so I have to change it from Public to Protected

<?php

class Animal {

protected function getName($arg) {

echo "This is the ".$arg." Class";

}

}

?>

now what will happened if we created an Animal object like we did before?

output :

Fatal error: Call to protected method Animal::getName()

uh oh.. we got an error because we can't just accessed that because it's protected and only it's base class and subclasses can access that method.

what if we made it private and we call that function inside our cow.php?

animal.php

<?php

class Animal {

private function getName($arg) {

echo "This is the ".$arg." Class";

}

}

?>

cow.php

<?php

class Cow extends Animal{

public function sayName() {

$this->getName(get\_class($this));

}

}

?>

index.php

<?php

include\_once('animal.php');

include\_once('cow.php');

$cow = new Cow();

$cow->sayName();

?>

here's the output if you tried to run it

Call to private method Animal::getName()

ofcourse.. you can't call that method because it's private and can only be accessed by it's base class.

Conclusion

Today we've learned the use of encapsulation, this technique in programming is really useful if we are creating a function or some variables that we want to hide from some users.. It's good if you are creating a framework, you wouldn't want some programmers messed up your code and use it the wrong way. Thank you so much for reading my tutorial.

**Class :**  Class describes the nature of a particular thing. Structure and modularity is provided by a Class in object oriented programming environment.

**MAGIC METHODS : -**

### \_\_get()

This magic method runs when you try accessing a property that doesn’t exist in the class. It accepts 1 parameter which is the name of the property you tried accessing.

### \_\_set()

This magic method runs when you try creating or updating a property that doesn’t exist in the class. It accepts 2 parameters which is the name and the value of the property you attempted creating or updating.

### \_\_call()

This magic method runs when you try using a method that doesn’t exist. It allows 2 parameters which is the name of the method and the parameters you tried passing in. The parameters are converted into 1 array.

**class** Customer {

**public** **function** \_\_call($name, $args) {

var\_dump($name);

echo "**\n**";

var\_dump($args);

echo "**\n**";

}

}

$c = **new** Customer();

$c->setName("Sunil","Bhatia");

### \_\_toString()

This magic method runs when you try echoing out the object itself. If you define this method, you must return a string and not echo anything.

### \_\_clone()

This magic method runs when you try cloning an instance of the class. To clone an instance, you use the keyword ***clone***. If you look in your ***index.php*** file, you’ll see an example of how this is done. By cloning an instance, you’re creating an object with the same properties and methods of the cloned object. They become 2 separate instances after that. This is useful when you want to save an instance at a certain state just in case.

### \_\_invoke()

This magic method runs when you try running an object as a method.

### \_\_destruct()

This magic method runs when your object is destroyed. By default, all objects are destroyed when the script is finished running. If you would like to destroy an object earlier, then you can use the ***unset()***function. Here’s an example of how that would look like.

\_\_sleep() **magic method** is called when the object of a class is about to be serialized. This magic method \_\_sleep() does not accept any parameter and returns an array. The array should contain a list of class members that should be serialized. This means that if you don’t wish to serialize a particular class member, you should not include it in the array. Look at the example below:

**class** Customer {

**private** $name;

**private** $credit\_card\_number;

**public** **function** setName($name) {

$this->name = $name;

}

**public** **function** getName() {

return $this->name;

}

**public** **function** setCC($cc) {

$this->credit\_card\_number = $cc;

}

**public** **function** getCC() {

return $this->credit\_card\_number;

}

**public** **function** \_\_sleep() {

return array("name"); *//because of this, only name is serialized*

}

}

$c = **new** Customer();

$c->setName("Sunil");

$c->setCC("1234567890123456");

$data = serialize($c)."**\n**";

echo $data."**\n**";

**Output:**  
O:8:”Customer”:1:{s:14:” Customer name”;s:5:”Sunil”;}

\_\_wakeup() **magic method** is the opposite of the \_\_sleep() method. It is called when the object of a class is about to be unserialized. This magic method \_\_wakeup() does not accept any parameter nor returns anything. The \_\_wakeup() method is responsible for setup operations after an object has been unserialized. Look at the example below:

#### Destructor :

* It is similar to constructor method.
* Destructor method which actually works upon destroying an object.
* You can explicitly create a destructor method by naming it \_\_destruct().
* It will be invoked automatically by PHP at the end of the execution of your codes.

**14) Explain what a method is?**  
A method will affect only a particular object to which it is specified. Methods are verbs meaning they define actions which a particular object will perform. It also defines various other characteristics of a particular object.

Or

The **functions** which are declared in a class are called **methods**. A class **method** is exactly similar to **PHP functions**. Declaring a **method** in a class is an easy task, use one of the keyword public, protected, or private followed by a **method** name. public : The **method** can be accessed from outside the class

**3) Explain what is an object?**  
An object is a combination of messages and data. Objects can receive and send messages and use messages to interact with each other. The messages contain information that is to be passed to the recipient object.

2) Explain about Object oriented programming?  
Object oriented programming is one of the most popular methodologies in software development. It offers a powerful model for creating computer programs. It speeds the program development process, improves maintenance and enhances reusability of programs.