Basics of OOP in PHP

Object oriented programming is nothing but a technique to design your application. Application could be any type like it could be web based application, windows based application. OOP is a design concept. In object oriented programming, everything will be around the objects and class. By using OOP in php you can create modular web application. By using OOP in php we can perform any activity in the object model structure. There are many benefit of using oop over the parallel or procedural programming. Further in this part we will cover some basic of object and class and its implementation in php.

What is Object?

If you want to see theoretical definition of object described in the typical book of oop then following is the best definition of object is:

Anything in the world is an object. Look around and you can find lots of object. Your laptop, pc, car everything is an object. In this world every object has two thing properties and behaviors. Your car has property (color, brand name) and behavior(it can go forward and backward). If you are able to find properties and behaviors of real object then it will be very easy for you to work with Object Oriented Programming.

In real world different objects have different properties and behaviors. For example your television has property size, color, and has behavior turn on, turn off. If you observe carefully then you can find that every object has some property and behavior from other object. This phenomenon is called inheritance.

For example car object has property and behavior from engine object. If you are able to understand what is object then good to go ahead. If not then please start visualizing properties and behavior of object around you. You will definitely understand.

Object in programming is similar to real word object. Every programming object has some properties and behaviors. For example if you have object for interest calculator then it has property interest rate and capital and behavior simple interest calculation and compound interest calculation. Interest calculator has some property and method from calculator object like addition, multiplication etc.

What is Class?

Class is something which defines your object. For example your class is Car. And your Honda car is object of car class. Like object explanation, here we will take an example of the real word and then we will move further in programming definition.

Blueprint of the object is class. Class represents all properties and behaviors of object. For example your car class will define that car should have color, number of door and your car which is an object will have color green and 2 doors. Your car is object of class car. Or in
In terms of programming we can say your car object is an instance of the car class. So structural representation (blueprint) of your object is class.

Now let us take an example of the programming. Your interest calculator object is instance of class interest calculator. Interest calculator class defines properties like capital rate, and behavior like simple interest calculation and compound interest calculation. Your interest calculator object has property rate as 3% and capital 300 USD. So you are describing your class definition of rate by giving rate value equals to 3% and capital 300USD in your interest calculator object. Now in your object when interest calculation behavior will be applied it will take your rate of interest and capital and provide you the result. Again your interest calculator class will inherit the definition of its property and behavior from calculator class.

**Advantage of Object Oriented Programming:**

There are various advantage of using OOP over the procedural or parallel programming. Following are some of the basic advantages of using oop techniques.

1. **Re-Usability of your code:** If you will use OOP technique for creating your application then it will gives you a greater re-usability. For example, if you have created calculator class at one place then you can use the same calculator class in your application.

2. **Easy to Maintain:** Application develop using oop technique are easier to maintain than normal programming. Again let us take an example of your interest calculator class. Suppose your business need to change the calculation logic. They want to add some charges if your capital is less than 200 USD. Just think about your application is big and developed using normal programming techniques. So first you have to analyse that at how many places we have calculated interest, and then you will change. But just think of oop technique. You just need to change in your method of interest calculation at one place.

3. **Good Level of Abstraction:** Abstraction means making something hidden. By using oop technique you are abstracting your business logic from implementation. It will provide you greater ease. Again let us take an example of interest calculator. If you have created class for interest calculation and your team is going to use that class. Now you are only concern about how interest calculation will be performed, because you have created that. Your team member is always have understanding that if they will set rate and capital property and apply interest calculation method then it will return interest.

4. **Molecularity:** If you are creating separate class for your every problem then you are making it modular. So if someone need to change in the business logic part then he will always go to your business logic code part.

**Implementation of OOP in PHP:**

In this section we will discuss about some basic aspect of oop in php.
**Class in PHP:**

Class is a blueprint of any object in oop. So class is the first alphabet of oop. In php creation of class is very simple. You can create class using tag `class`. In class block you can define your properties as class variable and function as class behavior. So let us create a class for interest calculator and define its properties like rate, capital, duration and behavior like calculate interest.

```php
class interestCalculator
{
    var $rate;
    var $duration;
    var $capital;
    function calculateInterest()
    {
        return ($this->rate*$this->duration*$this->capital)/100;
    }
}
```

Above is a very simple and basic class to calculate interest. Let us explore all basic aspect of this class.

You can create class in php by using `class` keyword. Here `class interestCalculator{ }` is class block. You can define all of your properties and methods (behavior of class, we will use method or function instead of behavior) of your class inside of your class block. All variable started with `var` keyword is property of your class. Commonalty we can say these are variable of class also. And the function is methods of this class. You can design your own class with your own variable and function.

**Object in PHP:**

As we have already discussed that **object is an instance of any class**. So we will take our interestCalculator class as an example. Creating object of the class is very easy in php. You can create object of class with the help of `new` keyword. Following is very basic example of creation of object of your class interest calculator:

```php
$calculator = new InterestCalculator();
```

In above declaration you are creating object of your class interestCalculator in **variable $calculator**. Now your variable $calculator is an object of class interestCalculator. Next step is to set property or variable of object calculator and perform calculation of interest.

```php
$calculator = new InterestCalculator();
$calculator->rate = 3;
$calculator->duration =2;
$calculator->capital = 300;
echo $calculator->calculateInterest();
```
Here object of your class interestCalculator() is your php variable $calculator. In next 3 lines of above code you are setting properties of class. You can access property of class with ->. So in above code rate property is set using $calculator->rate = 3; Finally after setting all required properties you have called method calculateInterest().

For Indepth Coverage on OOP theory your can further read on wikipedia: http://en.wikipedia.org/wiki/Object-oriented_programming

Class in PHP:

Concept of class (or basic object oriented structure) introduced from php4. But complete coverage of class like access modifier or interface is introduced from php5. Creating class is very easy in php. You can create class with help of using class keyword in php. Following is a basic class example:

```php
class myOwnClass
{
    //variables of the class
    var $variable1;
    var $variable2;
    //Function of class
    function mergeVariable()
    {
        return $this->variable1 . $this->variable2;
    }
}
```

Class myOwnClass is created by using keyword class. Your name of the class will be general string without space. Also complete block of the class in enclosed within { }(see bold braces). All variables of this class are defined in the beginning of the class. Variables are starting with var keyword. From php5 you can declare variable using its level of visibility also. For example if you want to declare $variable1 to be accessible from anywhere then you can use public $variable1 instead of var $variable1. If you will use var $variable1 in php5, the variable will be treated as public by default.

Next part is function declaration of your class. As per the above example you can directly declare function as function mergeVariable(). It is very basic creation of function within your class and supports from php4. In php5 you can apply visibility on your function also. For the same function you can write in php5 like public function mergeVariable(). If you will not define your visibility factor by default your function will be treated as public.

Following is an example of same above class in php5:

```php
class myOwnClass
{
    //variables of the class
    public $variable1;
```
So in the basic architecture of php4 and php5 class is almost same except use of visibility. For now you just think that visibility is access factor of your class's method and variable. If you want your object to allow access of your variable or function then make it public. If you do not want your object to access methods and properties then make it private.

You can directly pass value to your class from by the class function parameter also. Function of the class always works like general function in the php. For example

```php
function mergeVariable($third_var)
{
    return $this->variable1 . $this->variable2 . $third_var;
}
```

You cannot create class with namestdClass in PHP. It is reserved class of the php. stdClass represent standard object. It is used to create empty object. You can use it without creating it. If you will forcefully try to create class with name stdClass then PHP will throw fatal error with following message.

```
Fatal error: Cannot redeclare class stdClass
```

It is recommended to not create your function of the class starting with __ like __call. Because function started with __ is seems like magic function in php.

**Object in PHP:**

Classes are useless without objects. Object is an instance of your class. If you have class then you need to create object of the class to solve your problem using class. You can create object of your class by using new keyword.

```php
$objClass = new myClass();
```
Now in above code you are creating object of class myClass in variable $objClass. You can create multiple object of your same class. Every object is different from other.

```php
$objClass1 = new myClass();
$objClass2 = new myClass();
```

To completely understand object Let us create full class and their object. Here I will create class for interest calculation and then I will create object of that class and calculate interest.

```php
//Creating class interestCalculator
class interestCalculator
{
    public $rate;
    public $duration;
    public $capital;
    public function calculateInterest()
    {
        return ($this->rate*$this->duration*$this->capital)/100;
    }
}

//Creating various object of class interestCalculator to calculate interest on various amount
$calculator1 = new InterestCalculator();
$calculator2 = new InterestCalculator();
$calculator1->rate = 3;
$calculator1->duration = 2;
$calculator1->capital = 300;
$calculator2->rate = 3.2;
$calculator2->duration = 3;
$calculator2->capital = 400;
$interest1 = $calculator1->calculateInterest();
$interest2 = $calculator2->calculateInterest();
echo "Your interest on capital $calculator1->capital with rate $calculator1->rate for duration $calculator1->duration is $interest1 <br/> ";
echo "Your interest on capital $calculator2->capital with rate $calculator2->rate for duration $calculator2->duration is $interest2 <br/> ";
```

Please run above code in browser. You will get following output.
Now please analyse above code carefully. We have created two object of interestCalculator class in variable $calculator1 and $calculator2. Now property value of both objects is different. For example $calculator1 capital is 300 and $calculator2 capital is 400. Whenever you will call calculateInterest function of the both object then they will calculate interest on their own properties.

Now just analyse code of your class interestCalculator

```php
class interestCalculator
{
    public $rate;
    public $duration;
    public $capital;
    public function calculateInterest()
    {
        return ($this->rate*$this->duration*$this->capital)/100;
    }
}
```

You can find that class has 3 variable or properties ($rate, $duration, $capital). Now look into function calculateInterest. In the body of the function we have used variable $this. $this is system defined object variable of the class. $this is object of self class in the current context. For the both object of interestCalculator class $this object is different. When you have object $calculator1 then $this->rate is 3 and in case of $calculator2 $this->rate is 3.2

```php
public function calculateInterest()
{
    $rate = 5;
    return ($this->rate*$this->duration*$this->capital)/100;
}
```
In above function of class $this->rate and $rate is different. $this->rate will always has values assigned by the object of the class but $rate is fix value. If you will replace $this->rate to $rate your rate of interest will always be 5.

```php
public function calculateInterest()
{
    $rate = 5;
    return ($this->rate*$this->duration*$this->capital)/100;
}
```

You can create object of the class in some different way also. Following is some of the example of creating object of class.

```php
$className = 'interestCalculator';
$calc1 = new $className();
```

From php 5.3 onward you can create object of class

```php
$cls1 = new interestCalculator();
$cls2 = new $cls1;
```

**Constructor of Classes and Objects:**

Constructor is nothing but a function defined in your php class. Constructor function automatically called when you will create object of the class. As soon as you will write $object = new yourClass() your constructor function of the class will be executed. In php4 you can create constructor by creating function with same name of your class. But from php5 you can also create constructor by defining magic function __construct. Please go through the blow example of the constructor.

**PHP 4 constructor( will work in php 5 also)**

```php
class interestCalculator
{
    var $rate;
    var $duration;
    var $capital;
    //Constructor of the class
    function interestCalculator()
```
PHP5 constructor

class interestCalculator
{
    public $rate;
    public $duration;
    public $capital;
    //Constructor of the class
    public function __construct()
    {
        $this->rate = 3;
        $this->duration = 4;
    }
}

In both whenever instance of the class will be created rate will be set to 3 and duration will be set to 4. But difference is in way of calling the constructor. In php4 you were limited to create constructor by creating function with same name of the class. But in php5 you can either create function with same name or create a function __construct to create constructor of the class. You can also pass parameter in the constructor.

class interestCalculator
{
    public $rate;
    public $duration;
    public $capital;
    //Constructor of the class
    public function __construct($rate , $duration)
    {
        $this->rate = $rate;
        $this->duration = $duration;
    }
}

$objCls = new interestCalculator(3.2 , 7) //passing value of $rate and $duration

If you have created parameter in the constructor you need to pass value for them on the time of object creation. $objCls = new interestCalculator(3.2 , 7). If you will not send value php will throw error.
Playing with visibility and other feature of the constructor:

Let us explore in depth of constructor for our classes and objects in php.

All implantation is described here are considered only for php5. Did you notice that I have created my constructor function public. If not then please go to above section and explore. Reason behind creating constructor function public is it is accessible from outside of the class. This function is executed when we are creating object. So php will always through error if you will create your constructor private. Let us try below code:

```php
class interestCalculator
{
    public $rate;
    public $duration;
    public $capital;
    //Constructor of the class
    private function __construct($rate, $duration)
    {
        $this->rate = $rate;
        $this->duration = $duration;
    }
}
$objCls = new interestCalculator(3.2, 7); //passing value of $rate and $duration
```

It will give you following output

**Fatal error:** Call to private interestCalculator::__construct() from invalid context

As you can define your constructor by creating function with same name of class(event in php5), if you will use following code then your output will be same:

```php
class interestCalculator
{
    public $rate;
    public $duration;
    public $capital;
    //Constructor of the class
    private function interestCalculator($rate, $duration)
    {
        $this->rate = $rate;
        $this->duration = $duration;
    }
}
```

Following error you will receive

**Fatal error**: Call to private interestCalculator::interestCalculator() from invalid context

So in short you can’t make your constructor private. If you will make your constructor private then you will receive an error.

**Now Just think that you can define your constructor either by creating function with same name of the class or by crating function with name __construct.** Now what happen if you will use both thing in your single class. Let us try this code:

```php
class test{
public function __construct(){
   echo 1;
}
function test(){
   echo 2;
}
}
$t = new test(); //Output will be 1
```

It will give you output 1. Means your __construct function is called. So if you have __construct then it will be the first preference. **If __construct function is not present then it will search for the function with the same name of class.** Think what happen in case if you have both option and your function __construct is private. Try this code

```php
class test{
private function __construct(){
   echo 1;
}
function test(){
   echo 2;
}
$t = new test();
```
You will get following error:

**Fatal error:** Call to private test::__construct() from invalid context

**Best Practice of Classes and Objects:**

Following are some best practice of using classes and objects in your application.

1. Instead of assigning variable of the classes after creating object it is good if you use constructor.
2. Use visibility as required. Do not make your variable and method either more secure or completely open. Over security will effect your flexibility, under security will distrust your structure.
3. Follow some convention in your classes and objects. Like start all public method with camel case, all protected method and variable prefix with _ etc. It will give you better visibility.
4. Do not try to do everything in single class. Create class very specific to your requirement. It will save your time and execution.
5. Always try to create every class in separate file and follow some naming convention.

**Download Code** (Experimental) for Classes and Objects.

For further detail on Classes and Object in PHP you may read following:


Magic methods in php are **some predefined function by php compiler which executes on some event**. Magic methods start with prefix __, for example __call, __get, __set. __construct is a magic method which automatically calls on creating object of the classes. There are various magic methods in php. Here we will discuss some of the most common magic methods of php which will be use in object oriented programming. First of let us review all magic method with short description.

**List of List of Magic Methods in PHP:**

<table>
<thead>
<tr>
<th>Magic Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>__construct</td>
<td>This magic methods is called when someone create object of your class. Usually this is used for creating constructor in php5.</td>
</tr>
<tr>
<td>__destruct</td>
<td>This magic method is called when object of your class is unset. This is just opposite of __construct.</td>
</tr>
<tr>
<td>__get</td>
<td>This method called when your object attempt to read property or variable of the class which is inaccessible or unavailable.</td>
</tr>
</tbody>
</table>
__set This method called when object of your class attempts to set value of the property which is really inaccessible or unavailable in your class.

__isset This magic method trigger when isset() function is applied on any property of the class which is inaccessible or unavailable.

__unset __unset is something opposite of isset method. This method triggers when unset() function called on inaccessible or unavailable property of the class.

__call __call magic method trigger when you are attempting to call method or function of the class which is either inaccessible or unavailable.

__callstatic __callstatic execute when inaccessible or unavailable method is in static context.

__sleep __sleep methods trigger when you are going to serialize your class object.

__wakeup __wakeup executes when you are unserializing any class object.

__toString __toString executes when you are using echo on your object.

__invoke __invoke called when you are using object of your class as function

Above list is the most common used magic methods in PHP object oriented programming. Above magic methods of PHP executes on some specific events occur on your class object. For example if you simply echo your object then __toString method triggers. Let us create group of related magic method and analyze how it is working.

__construct and __destruct magic method in PHP:

__construct method trigger on creation of object. And __destruct triggers of deletion of object. Following is very basic example of __construct and __destruct magic method in PHP:

```php
class test {
    function __construct() {
        echo 1;
    }
    function __destruct() {
        echo 2;
    }
}
$objT = new test(); //__construct get automatically executed and print 1 on screen
unset($objT); //__destruct triggers and print 2.
```

__get __set __call and __callStatic Magic Methods:

__get, __set, __call and __callStatic magic methods in PHP directly related with no accessible method and property of the class.
__get\ takes\ one\ argument\ and\ executes\ when\ any\ inaccessible\ property\ of\ the\ method\ is\ called.\ It\ takes\ name\ of\ the\ property\ as\ argument.

__set\ takes\ two\ property\ and\ executes\ when\ object\ try\ to\ set\ value\ in\ inaccessible\ property.\ It\ take\ first\ parameter\ as\ name\ of\ the\ property\ and\ second\ as\ the\ value\ which\ object\ is\ try\ to\ set.

__call\ method\ fires\ when\ object\ of\ your\ class\ is\ trying\ to\ call\ method\ of\ property\ which\ is\ either\ non\ accessible\ or\ not\ available.\ It\ takes\ 2\ parameter,\ First\ parameter\ is\ string\ and\ is\ name\ of\ function.\ Second\ parameter\ is\ an\ array\ which\ is\ arguments\ passed\ in\ the\ function.

__callStatic\ is\ a\ static\ magic\ method.\ It\ executes\ when\ any\ method\ of\ your\ class\ is\ called\ by\ static\ techniques.

Following\ is\ example\ of\ __get\ ,\ __set\ ,\ __call\ and\ __callStatic\ magic\ methods:

class\ test
{
    function\ __get($name)
    {
        echo  "__get\ executed\ with\ name\ $name";
    }
    function\ __set($name,\ $value)
    {
        echo  "__set\ executed\ with\ name\ $name,\ value\ $value";
    }
    function\ __call($name,\ $parameter)
    {
        $a = print_r($parameter,\ true);\ //taking\ recursive\ array\ in\ string
        echo  "__call\ executed\ with\ name\ $name,\ parameter\ $a";
    }
    static\ function\ __callStatic($name,\ $parameter)
    {
        $a = print_r($parameter,\ true);\ //taking\ recursive\ array\ in\ string
        echo  "__callStatic\ executed\ with\ name\ $name,\ parameter\ $a";
    }
}

$a = new\ test();
$a->abc = 3;  //__set\ will\ executed
$app = $a->pqr;  //__get\ will\ triggerd
$a->getMyName('ankur',\ 'techflirt',\ 'etc');  //__call\ will\ be
executed
test::xyz('1',\ 'qpc',\ 'test');  //__callStatic\ will\ be\ executed
__isset and __unset magic methods:

__isset and __unset magic methods in php are opposite of each other. 
__isset magic methods executes when function isset() is applied on property which is not available or not defined. It takes name of the parameter as an argument. 
__unset magic method triggers when unset() method is applied on the property which is either not defined or not accessible. It takes name of the parameter as an argument.

Following is example of __isset and __unset magic method in php

```php
class test
{
    function __isset($name)
    {
        echo "__isset is called for $name";
    }
    function __unset($name)
    {
        echo "__unset is called for $name";
    }
}
$a = new test();
isset($a->x);
unset($a->c);
```

Download Code for magic method in PHP.

Above are some of the basic magic methods in the php. For complete list you can go to php.net site.

Available Visibility in PHP Classes:

There are 3 type of visibility available in php for controlling your property or method.

1. **Public**: Public method or variable can be accessible from anywhere. I mean from inside the class, outside the class and in child (will dicuss in next chapter) class also.
2. **Private**: Method or property with private visibility can only be accessible inside the class. You cannot access private method or variable from outside of your class.
3. **Protected**: Method or variable with protected visibility can only be access in the derived class. Or in other word in child class. Protected will be used in the process of inheritance.
Public Visibility in PHP Classes:

Public visibility is least restricted visibility available in php. If you will not define the visibility factor with your method or property then public will be by default applied. Public methods or variables can be accessible from anywhere. For example, It can be accessible from using object(outside the class), or inside the class, or in child class. Following is the example of the public visibility in php classes:

class test
{
    public $abc;
    public $xyz;
    public function xyz()
    {
    }
}
$objA = new test();
echo $objA->abc;//accessible from outside
$objA->xyz();//public method of the class test

So in above example class test is the very basic class. In this class everything is open. Minimum restriction in the class is to access its property and methods using object outside the class.

Private Visibility in PHP Classes:

Private method or properties can only be accessible within the class. You cannot access private variable or function of the class by making object outside the class. But you can use private function and property within the class using $this object. Private visibility in php classes is used when you do not want your property or function to be exposed outside the class. Following is the example of Private visibility in php classes.

Class test
{
    public $abc;
    private $xyz;
    public function pubDo($a)
    {
        echo $a;
    }
    private function privDo($b)
    {
        echo $b;
    }
}
public function pubPrivDo()
{
    $this->xyz = 1;
    $this->privDo(1);
}

$objT = new test();
$objT->abc = 3; //Works fine
$objT->xyz = 1; //Throw fatal error of visibility
$objT->pubDo("test"); //Print "test"
$objT->privDo(1); //Fatal error of visibility
$objT->pubPrivDo(); //Within this method private function privDo and variable xyz is called using $this variable.

**Protected Visibility in PHP Classes:**

Protected visibility in php classes are only useful in case of inheritance and interface. We will discuss in dept of interfaces and inheritance in other chapter of this tutorial. Protected method or variable can be accessible either within class or child class. Here we will take very basic example:

```php
class parent
{
    protected $pr;
    public $a;
    protected function testParent()
    {
        echo "this is test";
    }
}
class child extends parent
{
    public function testChild()
    {
        $this->testParent(); //will work because it
    }
}
$objParent = new parent();
$objParent->testParent(); //Throw error
$objChild = new Child();
$objChild->setChild(); //work because test child will call test parent.
```

If you will analyze above section you can found that method testParent() is not accessible from object of class. But it is accessible in child class.
Always use correct visibility in PHP classes to keep your structure healthy. Do not use code like this. It is break all visibility of your PHP class.

```php
class test {
    public function method($method) {
        $this->$method();
    }
    private function abc() {
        //Do Something
    }
    protected function xyz() {
        //do something
    }
}
$objT = new test();
$objT->method('abc');
$objT->method('xyz');
```

This is the very stupid implementation of public, private and protected. Because you can call any type of method in this implementation.


Static methods 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 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 by using static keyword. following is the basic example of static variable in PHP class:

```php
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 within 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 is 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

**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** for static methods in PHP.

For further details about static methods and properties in php you can go to:

**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 is known as **child class**. The class which is being inherited is known 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 elements and their rendering is same in all elements. For example all html mention above is having name, id, class attribute which is same. Also rendering of those elements 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 an example of basic inheritance in php. All methods (protected and public) from HTML class are 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 gives 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. To implement 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:

```php
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 an 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 cannot inherit more than one parent class. But hierarchical inheritance is possible in php. Hierarchical means Parent inherit property of grandparent class. Grandchild inherits property of parent class. So in multilevel inheritance child can get some property from grandparent class also.

**Example of Multiple inheritance in PHP**

```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**

```php
class grandParent
{
   //Body of your class
}
class parent extends grandParent
{
   //Body Of your class
}
class child extends parent
{
   //Body of your class
}
```

This is very basic example of multilevel inheritance. In php it is possible to implement multilevel inheritance. In above example parent class is inheriting grandparent property. And child is inheriting parent property. So child has some parent and grandparent 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 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:

```php
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:

```php
class parent
{
    protected function xyz()
    {
        //Your function body
    }
}
class child extends parent
{
    public function xyz()
    {
        //your function body
    }
```
Abstract Classes:

As from name it seems like something that is hidden. Yes nature of the abstract classes is same. Abstract classes are those classes which cannot be directly initialized. Or in other word we can say that you cannot 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 is the method which is not only defined but also 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 similar like other oop languages. You can create abstract classes in php using **abstract** keyword. Once you will make any class abstract in php you cannot create object of that class.

```php
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.

```php
abstract class testParent
{
public function abc()
{
//body of your function
}
}
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:

```php
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.

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

class childTest
{
    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 declared 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 declare 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.

**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.

```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.

```php
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.

```php
interface template1
{
    public function f1();
}
interface template2 extends template1
```
So here template2 has all property of template1. So whenever you will implement template2 in your class, you have to create function of both interfaces.

You can also extend multiple interfaces in one interface in php.

```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.

```php
interface template1
{
    public function f1();
}
interface template2
{
    public function f2();
}
class test implements template1, template2
{
    public function f1()
    {
        //your function body
    }
    public function f2()
    {
        //your function body
    }
}
```

You cannot 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

```php
interface template1
{
    public function f1($a)
}
class test implements template1
{
    public function f1($a)
    {
        echo $a;
    }
}
```

Above code 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 have 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 it 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:  

**What is Method Overriding in OOP?**

Basic meaning of overriding in oop is same as real word meaning. In real world 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 implement method overriding 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 world. In real world 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 do 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:

Implementation of overriding in php is very easy. If your parent class has a function then you can create function with same name in your child class to override the function. Implementation of overriding cannot be achieved by creating 2 function with same name and different argument in php. Because we cannot create same name function more than 1 time in php class.

Overloading in PHP:

As we know that we cannot implement overloading by creating 2 functions 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 to ensure that our business logic of overloading works only for overloadedFunction 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 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.

```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.


**Object Cloning in PHP:**

If you will directly copy objects in php, then it will copy by reference, not by value. Means if you will change main object data then copied object will be affected. Also if you will change value of the copied object then main object value will be changed. So if you want to create copy of the **object which should never referenced to original object then you can take help of object cloning in php.**
Object copy or by reference copy:

Typical copy of object in php works by reference. Means both(main and copied) object will be interlinked. For example

```php
class test
{
    public $a;
    private $b;
    function __construct($a, $b)
    {
        $this->a = $a;
        $this->b = $b;
    }
}
$a = new test("ankur" , "techflirt");
$b = $a; //Copy of the object
$a->a = "no Ankur";
print_r($a);
print_r($b);
```

Following is output of above code:

```
test Object
( [
    [a] => no Ankur
    [b:test:private] => techflirt
] )
test Object
( [
    [a] => no Ankur
    [b:test:private] => techflirt
] )
```

So here in object $a after copy it to $b, we have changed its object value a to "no Ankur", and then we have print both object value and found that $b has changes made in $a. This is one of the very useful when you want object by reference. But when you need object copy by value this is show stopper. So to overcome with this limitation php has provided feature of object cloning separably.

Implementation of Object Cloning in PHP:

As we have seen that typical copy of object in php is by reference. We can copy two objects by value in php by using method of object cloning. Object cloning in php can be implemented using clone keyword. Following is example of object cloning:
class test
{
    public $a;
    private $b;
    function __construct($a, $b)
    {
        $this->a = $a;
        $this->b = $b;
    }
}
$a = new test("ankur", "techflirt");
$b = $a;  //Copy of the object
$c = clone $a;  //clone of the object
$a->a = "no Ankur";
print_r($a);
print_r($b);
print_r($c);

In above example $c is clone of $a variable. So $c is completely separate from object $a and $b. Following is output of above code.

test Object
(
    [a] => no Ankur
    [b:test:private] => techflirt
)
test Object
(
    [a] => no Ankur
    [b:test:private] => techflirt
)
test Object
(
    [a] => ankur
    [b:test:private] => techflirt
)
test Object
(
    [a] => no Ankur
    [b:test:private] => techflirt
)

Object cloning with magic method __clone:

Suppose you want to change value of your property $a of the test class in case of cloning of object in php. We can change behaviour of the clone object in php using magic method __clone. Magic method clone executes when object cloning is performed. As soon as php
execute statement $c = clone $a, __clone method invoked. Following is example of the object cloning and __clone magic method in php

class test
{
    public $a;
    private $b;
    function __construct($a, $b)
    {
        $this->a = $a;
        $this->b = $b;
    }
    function __clone()
    {
        $this->a = "c";
    }
}
$a = new test("ankur", "techflirt");
$b = $a; //Copy of the object
$c = clone $a; //clone of the object
$a->a = "no Ankur";
print_r($a);
print_r($b);
print_r($c);
print_r($a);

So here $c->a will be "c". Because __clone method will be invoked and only for object $c it will set value of $this->a to 'c'. Following is output of above example :

test Object
(
    [a] => no Ankur
    [b:test:private] => techflirt
)
test Object
(
    [a] => no Ankur
    [b:test:private] => techflirt
)
test Object
(
    [a] => ankur
    [b:test:private] => techflirt
)
test Object
(
    [a] => no Ankur
For more details about object cloning in php you can read:

http://en.wikipedia.org/wiki/Object_copy