CACS 205: Script Language

Advance Server Sitde Scripting 7 Hrs.

- Object Oriented Programming in PHP
- Classes and Objects,
- Defining and using properties and method
- Constructions and Destructions,
- Method Overriding,
- Encapsulation,
- Inheritance,
- Polymorphism,
- Static member
- Exception handling

Object Oriented Programming 15 Hrs.

What is Object Oriented Programming

- Object-Oriented Programming (OOP) is a programming model that is based on the concept of classes and objects. As opposed to procedural programming where the focus is on writing procedures or functions that perform operations on the data, in object-oriented programming the focus is on the creations of objects which contain both data and functions together.
- Object-oriented programming has several advantages over conventional or procedural style of programming. The most important ones are listed below:
- It provides a clear modular structure for the programs.
- It helps you adhere to the "don't repeat yourself" (DRY) principle, and thus make your code much easier to maintain, modify and debug.
- It makes it possible to create more complicated behavior with less code and shorter development time and high degree of reusability.

Object Oriented Programming 15 Hrs.

Classes and Objects

- Classes and objects are the two main aspects of object-oriented programming. A class is a self-contained, independent collection of variables and functions which work together to perform one or more specific tasks, while objects are individual instances of a class.
- A class acts as a template or blueprint from which lots of individual objects can be created. When individual objects are created, they inherit the same generic properties and behaviors, although each object may have different values for certain properties.
- For example, think of a class as a blueprint for a house. The blueprint itself is not a house, but is a detailed plan of the house. While, an object is like an actual house built according to that blueprint. We can build several identical houses from the same blueprint, but each house may have different paints, interiors and families inside, as shown in the illustration below.

Object Oriented Programming 15 Hrs.

```php
<?php
class Person
{
  private $name;
  public function getName()
  {
    return $this->name;
  }
  public function setName($name)
  {
    $this->name = $name;
  }
}
$person = new Person();
$person->setName('Krishna Acharya');
echo $person->getName(); // prints 'Krishna Acharya'
//$person->name = 'Krishna Acharya';
//echo $person->name;
?>
```


Person is Class
$person is Object of Class of Person
Member function(getName,setName and member variables($name)

- Syntactically, variables within a class are called properties, whereas functions are called methods. Also class names conventionally are written in PascalCase i.e. each concatenated word starts with an uppercase letter (e.g. MyClass).

Object Oriented Programming 15 Hrs.

```php
<?Php
class Car
{
    private $make;
    function __get($name)
    {
        return $this->$name;
    }
    function __set($name, $value)
    {
        $this->$name = $value;
    }
}
$car = new Car();
$car->make = 'BMWTTC';
echo $car->make;
?>
```

Object Oriented Programming 15 Hrs.

Controlling the Visibility of Properties and Methods

When working with classes, you can even restrict access to its properties and methods using the visibility keywords for greater control. There are three visibility keywords (from most visible to least visible): public, protected, private, which determines how and from where properties and methods can be accessed and modified.

public — A public property or method can be accessed anywhere, from within the class and outside. This is the default visibility for all class members in PHP.
protected — A protected property or method can only be accessed from within the class itself or in child or inherited classes i.e. classes that extends that class.
private — A private property or method is accessible only from within the class that defines it. Even child or inherited classes cannot access private properties or methods.

Object Oriented Programming 15 Hrs.

PHP Constructor

- Constructor function of a class get invoked automatically when an object of a class is created.
- Constructors does not have any return type so constructors does not return anything.
- Constructors can take any number of input values as parameter.
- It just ensures that an object should go through proper initialization process.
- Constructors are used when we want to set any property in class when an object is created or before an object is used.

```php
<?php
class Person
{
  private $name;
  public function __construct(){
     echo 'The class "' . __CLASS__ . '" was initiated!<br>';
     $name="Object Oriented PHP";
    }
  public function getName()
  {
    return $this->name;
  }
  public function setName($name)
  {
    $this->name = $name;
  }
}
$person = new Person();
$person->setName('Krishna Acharya');
echo $person->getName(); // prints 'Krishna Acharya'
//$person->name = 'Krishna Acharya';
//echo $person->name;
?>
```

```php
<?php

class Friend {
    private $born;
    private $name;
    function __construct($name, $born) {
        $this->name = $name;
        $this->born = $born;
    }
    function getInfo() {
        echo "My friend $this->name was born in $this->born\n";
    }
}
$friend = new Friend("Ekraj Adhikari", 1990);
$friend->getInfo();
?>
```

PHP Destructor

- Destructors are responsible for destroying objects.
- Destructor function invokes automatically when an object is no-longer in use in the application.
- It automatically closes all the available resources of an object.
- It automatically cleans the object from the memory.
- Destructor function is also useful if you want to write some clean codes like closing data connection, unset some class properties, clearing memory, closing socket connections and so on.
- In terms of managing the memory resources it is better to destroy an object when no longer needed.

Object Oriented Programming 15 Hrs.

PHP Destructor

```php
<?php
class Person
{
   private $name;
   public function __construct(){
      echo 'The class "' . __CLASS__ . '" was initiated!<br>';
      $name="Object Oriented PHP";
    }
   public function getName()
   {
     return $this->name;
   }
   public function setName($name)
   {
     $this->name = $name;
   }
   public function __destruct(){

        echo "<br> Destructor :: Terminating ";
        echo 'The class "' . __CLASS__ . '" was destroyed.<br>';
    }
}
$person = new Person();
$person->setName('Krishna Acharya');
echo $person->getName(); // prints 'Krishna Acharya'
//$person->name = 'Krishna Acharya';
//echo $person->name;
?>
```

Object Oriented Programming 15 Hrs.

Method overloading(polymorphism)

Method overloading allows the creation of several methods with the same name which differ from each other in the type of the input parameter.

```php
<?php

class Sum {

    public function getSum() {
        return 0;
    }

    public function getSum($x) {
        return $x;
    }

    public function getSum($x, $y) {
        return $x + $y;
    }
}

$s = new Sum();
echo $s->getSum() . "\n" ;
echo $s->getSum(5) . "\n" ;
echo $s->getSum(3, 4) . "\n" ;

?>
```

```php
class shape {
  function __call($name_of_function, $arguments) {
    if($name_of_function == 'area') {
      switch (count($arguments)) {
        case 1:
          return 3.14 * $arguments[0];
        case 2:
          return $arguments[0]*$arguments[1];
      }
    }
  }
}

$s = new Shape;
echo($s->area(2));
echo "<br>";
echo ($s->area(4, 2));
?>
```

```php
<?php
class Sum {
  public function getSum() {
    $sm=0;
    $args = func_get_args();
    if (empty($args))
      return 0;
    else
      foreach ($args as $arg) {
        $sm += $arg;
      }
        return $sm;
    }
}
$s = new Sum();
echo $s->getSum() . "<br>" ;
echo $s->getSum(5) . "<br>" ;
echo $s->getSum(3, 4) . "<br>" ;
echo $s->getSum(3, 4, 7) . "<br>" ;

?>
```

Object Oriented Programming 15 Hrs.

Function Overriding:

- Function overriding is same as other OOPs programming languages.
- In function overriding, both parent and child classes should have same function name with and number of arguments.
- It is used to replace parent method in child class.
- The purpose of overriding is to change the behavior of parent class method.
- The two methods with the same name and same parameter is called overriding.

```php
<?php
class P {
 function hello() {
    echo "Parent<br>";
 }
}
class C extends P {
  function hello() {
      // P::hello();
        echo "<br> Child";
  }
}
$p=new P;
$p->hello();
$c=new C;
$c->hello();

?>
```

In the example code p->hello() method refers to P class function where as c->hello function refer to c->hello function.
To avoid such confusion we have option p::hello() call from child class function (hello) in first line

Object Oriented Programming 15 Hrs.

Polymorphism

The word polymorphism is derive from Greek word poly and morphism. Ploy means many and morphism is form which means a function can be use as many form. Or one function but different signatures is called polymorphism.

```php
<?php
class Message
{
  public function formatMessage($message)
  {
    return printf("<i>%s</i>", $message);
  }
}

class BoldMessage extends Message
{
  public function formatMessage($message)
  {
    return printf("<b>%s</b>", $message);
  }
}

$message = new Message();
$message->formatMessage('Hello World'); // prints '<i>Hello World</i>'

$message = new BoldMessage();
$message->formatMessage('Hello World'); // prints '<b>Hello World</b>'
?>
```

```php
<?php
interface Shape {
  public function calcArea();
}
class Circle implements Shape {
  private $radius;
  public function __construct($radius)
  {
    $this -> radius = $radius;
  }

  public function calcArea()
  {
    return $this -> radius * $this -> radius * pi();
  }
}
class Rectangle implements Shape {
  private $width;
  private $height;
  public function __construct($width, $height)
  {
    $this -> width = $width;
    $this -> height = $height;
  }
  public function calcArea()
  {
    return $this -> width * $this -> height;
  }
}
$circ = new Circle(3);
echo $circ -> calcArea();
echo "<br>";
$rect = new Rectangle(3,4);
echo $rect -> calcArea();
?>
```

Object Oriented Programming 15 Hrs.

Encapsulation

- Encapsulation is a concept of wrapping up or binding up related data members and methods in a single class known as encapsulation.
- **Data abstraction** is related to hide the essential internal property of that class


```php
<?php
class Person
{
  private $name;
  public function getName()
  {
    return $this->name;
  }
  public function setName($name)
  {
    $this->name = $name;
  }
}
$person = new Person();
$person->setName('Krishna Acharya');
echo $person->getName(); // prints 'Krishna Acharya'
//$person->name = 'Krishna Acharya';
//echo $person->name;
?>
```

Object Oriented Programming 15 Hrs.

Inheritance:

Inheritance is one of the core concepts of object-oriented programming (OOP) languages. It is a mechanism where we can to derive a class from another class for a hierarchy of classes that share a set of attributes and methods.

In Object Oriented Programming, when a class derives its properties and methods from another class then it is called Inheritance. The class that derives properties and methods is called the child-class or the sub-class. The class from which inheritance takes place is called the super-class or the parent-class. We use the extends keyword to inherit from a parent class.

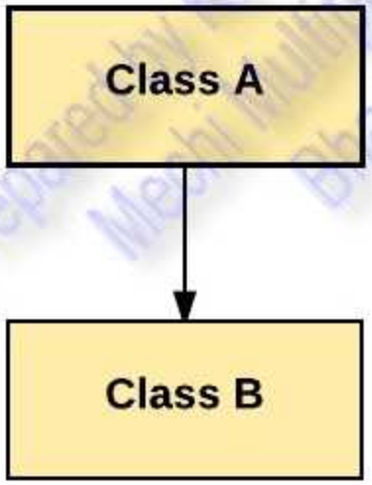
Advantage of inheritance:

- Save time.
- It help to increase code reusability.
- Reduces the chance of errors
- Makes it easier to understand the inherited class
- Makes it easier to understand the inherited class
- Make programs easier to write.

Object Oriented Programming 15 Hrs

Single Inheritance:

In Single Inheritance one class extends another class (one class only).




```php
<?php
class Person{
  private $id;
  private $name;
  public function __construct($para1,$para2){
      $this->id=$para1;
      $this->name=$para2;
  }
  public function getName(){
    return $this->name;
  }
  public function getId(){
     return $this->id;
  }
}
class Employee extends Person{
  private $salary;
  public function __construct($para1,$param2,$param3){
  parent::__construct($para1,$param2);
    $this->salary=$param3;
  }
  public function getSalary(){
    return $this->salary;
  }
}
$person = new Employee(1000,"Krishna Acharya",78000);
echo $person->getId()."<br>";
echo $person->getName()."<br>";
echo $person->getSalary()."<br>";
?>
```

Object Oriented Programming 15 Hrs.

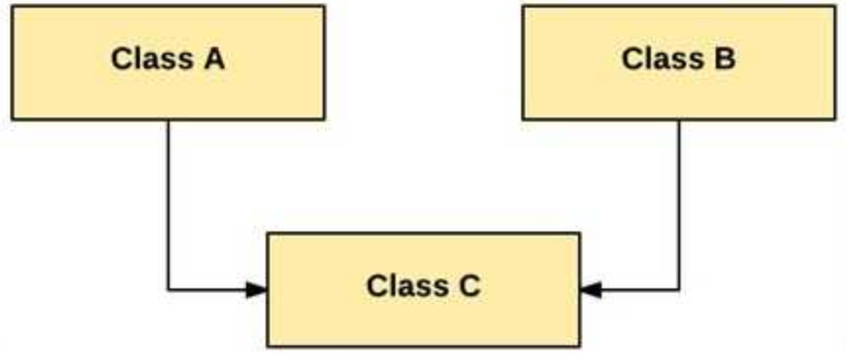
Multiple Inheritance:

- In Multiple Inheritance, one class extending more than one class. PHP does not support multiple inheritance directly but it can possible with use of interface.
- PHP doesn't support multiple inheritance but by using Interfaces in PHP or using Traits in PHP instead of classes, we can implement it.

```php
<?php
class A{
    public function insideA(){
     echo "I am in class A";
    }
}
interface B {
    public function insideB();
}

class Multiple extends A implements B {
     function insideB() {
         echo "<br>I am in interface";
     }
     public function insidemultiple() {
     echo "<br>I am in inherited class";
     }
}
$obj = new Multiple();
$obj->insideA();
$obj->insideB();
$obj->insidemultiple();
?>
```

```php
<?php
Trait Hello {
    public function hello() {
        echo "Hello";
    }
}
Trait World {
    public function world() {
        echo "World";
    }
}
class MyClass {
    use Hello, World;
}
$obj = new MyClass();
$obj -> hello();
$obj -> world();
?>
```

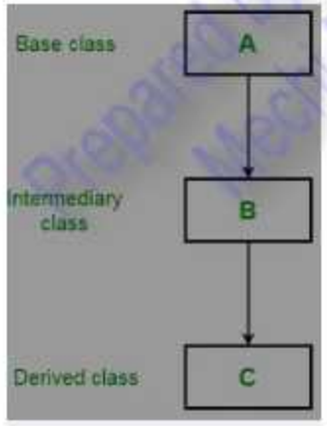


Object Oriented Programming 15 Hrs.

Multilevel Inheritance:

When a class is derived from a class which is also derived from another class, i.e. a class having more than one parent classes, such inheritance is called Multilevel Inheritance.

The level of inheritance can be extended to any number of level depending upon the relation.




```php
<?php
class GrandFather{
  public function gfAge(){
    return  'Age is 80<br>';
  }
}

class Father extends GrandFather{
  public function fAge(){
    return  ' Age is 50<br>';
  }
}

class Son extends Father{
  public function sAge(){
    return  'Age is 20<br>';
  }
  public function myHistory(){
      echo "my grand father ".parent::gfAge();
      echo "my father ".parent::fAge();
      echo "my ".$this->sAge();
      }
}

$son = new Son();
$son->myHistory();
?>
```

Object Oriented Programming 15 Hrs.

Static variable:

it is very handy to access methods and properties in terms of a class rather than an object. This can be done with the help of static keyword. Any method declared as static is accessible without the creation of an object. Static functions are associated with the class, not an instance of the class. They are permitted to access only static methods and static variables. To add a static method to the class, static keyword is used.

```php
<?php

// Class definition
class abc {
    public static $data = 'Mechi Multiple Campus ';
    public static function xyz() {
        echo "A computer science portal";
    }
}
echo abc::$data;
abc::xyz();
?>
```

```php
<?php
class solution {
    static $count;
    public static function getCount() {
        return self::$count++;
    }
}
solution::$count = 1;
for($i = 0; $i < 5; ++$i) {
    echo 'The next value is: '.
    solution::getCount() . "<br>";
}

?>
```

Static Members: Static variables or static functions are directly related with class. It can access static variables or functions directly with class name. Use 'static' keyword to declare static function or static variable.