Object-Oriented Programming

Object-Oriented Programming is a type of programming language methodology in this type that helps in building complex, reusable web applications. This feature added in php5 and now has a full object model. Object-oriented or object-orientation is a software engineering concept, in which concepts are represented as "objects" which can contain data, in the form of fields (often known as attributes or properties or data member), and code, in the form of procedures (often known as methods or function). 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.

Basic Concept of Object-Oriented Programming:

Class - This is a programmer-defined data type, which includes local functions as well as local data. You can think of a class as a template or blueprint for making many instances of the same kind (or class) of object. Class is a collection of objects. Object has properties and behavior. First, we have to define a PHP class, where class name should be same as filename Mostly class creates in separate file. A valid class name (excluding the reserved words) starts with a letter or underscore, followed by any number of letters, numbers, or underscores.

Class names usually begin with an uppercase letter to distinguish them from other identifiers.

```
<?php
class ClassName
{
    // Add Data member or property statements here
   // Add the member function or methods here
}
?>
```

Here class is keyword and ***ClassName*** is a user define class name identifier then followed by a set of curly braces ({}) which enclose constants, variables (called "data member", "properties", or "attribute") and functions called "methods" or “behavior ") belonging to the class.

Data Member Variable - These are the variables defined inside a class. This data will be invisible to the outside of the class and can be accessed via member functions. These variables are called attribute of the object once an object is created. Declaring a property in a class is an easy task; use one of the keyword public, protected, or private followed by a normal variable declaration. If declared using ***var*** (compatibility with PHP 4), the property will be defined as public.

public: The data member or property can be accessed from outside the class, either by the script or from another class. Public is a default state when you declare a **variable** or method can be accessed by anything directly to the object.

private: No access is granted from outside the class, either by the script or from another class. Private can be referenced only within the object, not subclasses.

protected: No access is granted from outside the class except a class that's a child of the class with the protected property or method. Protected can be accessed only within the object and subclasses.

Example

```
<?php
class Students
{
    public roll;       // data member or attribute of student roll number
    public name;     // data member or attribute of student name
}
?>
```

Member function - These are the function defined inside a class and are used to access object data and define their behavior.

Constructor - refers to a special type of function or method, which will be called automatically whenever there is an object formation from a class. It is use for data member or attributes initialization when an object is created. The 'construct' method starts with two underscores (__). The constructor is not required if you do not want to pass any property values or perform any actions when the object is created.

Syntax

```
function __construct([argument1, argument2, ...])
{
    /* Initialization code for data members of class */
}
```

Example

```
<?php
  function __construct($roll=0,$name="nothing", $age=0) // default initialization
  {
            $this->roll = $roll;
             $this->name = $name;
             $this->age = $age;
  }
?>
```

Object - An individual instance of the data structure defined by a class. You define a class once and then make many objects that belong to it. Objects are also known as instance of a class. To create an object out of a class, the ***new*** keyword must be used.

Syntax

```
<?php
$objectname = new ClassName();
?>
```

Inheritance – When the properties and the methods of the parent class are accessed by the child class, we call the concept has inheritance. The child class can inherit the parent method and give own method implementation, this property is called overridden method. When the same method of the parent class is inherited we call as inherited method. This is concerned with the relationship between classes. The relationship takes the form of a parent and child.

Inheritance is a well-established programming principle, and PHP makes use of this principle in its object model. This principle will affect the way many classes and objects relate to one another. when you extend a class, the subclass inherits all of the public and protected methods from the parent class. Unless a class overrides those methods, they will retain their original functionality.

The child uses the methods defined in the parent class. The main purpose of inheritance is Re-usability a number of children, can inherit from the same parent. This is very useful when we have to provide common functionality such as adding, updating and deleting data from the database other form/situation. This is useful for defining and abstracting functionality, and permits the implementation of additional functionality in similar objects without the need to re-implement all of the shared functionality.

The class which is inherited is called Parent class (or super class or base class) while the class which is inheriting other class is called as Child class (or sub class or derived class).

Now let us see types of inheritance supported in Object Oriented Programming in PHP.

Single Level Inheritance:

In Single Level Inheritance the Parent class methods will be extended by the child class. All the methods can be inherited. In order to declare that one class inherits the code from another class, we use the ***extends*** keyword

```
<?php
class ParentClasssName
{
  // The parent's class code
}

class ChildClassName extends ParentClassName
{
  // The  child can use the parent's class code
}
?>
```

Multi-Level Inheritance:

In Multi-Level Inheritance, the parent class method will be inherited by child class and again subclass will inherit the child class method but PHP does support Multi-level inheritance.

Parent class – A class that is inherited from by another class. This is also called a base class or super class.

Child Class – A class that inherits from another class. This is also called a subclass or derived class.

Consider a class **Human** with basic methods in it like ***walk (), eat (), hear (), see ()*** etc. Now if we have to create another class for **Animals,** with all the properties and methods of the class Human and some specific features available only Animals, we can do so by inheriting **Human** class in **Animals** class.

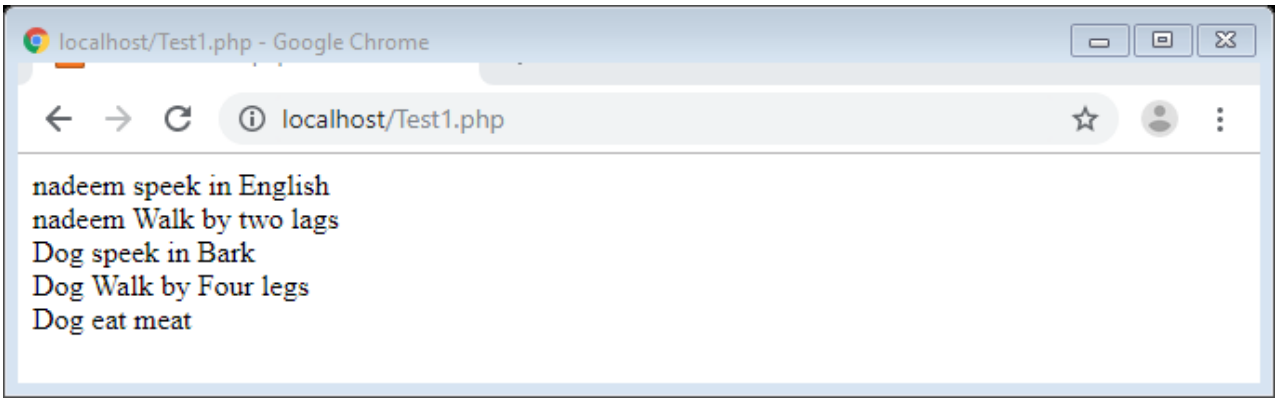
Example

```
<?php
class Human
{
    public $name;
    function __construct($name = "nothing")
   {
        $this->name = $name;
    }
  public function speek($string)
 {
      echo $this->name . " speek in  " . $string .   "<br>";
  }
  public  function walk($string )
 {
     echo  $this->name . " Walk " . $string . "<br>";
  }
}
class Animal extends Human
{
 public function eat($string)
 {
    echo $this->name . " eat  " . $string .   "<br>";
 }
}
```

```
$nadeem = new Human("nadeem");  // call constructor by Human/Super  class!"
$nadeem->speek("English"); //   call speek function by Human/Super  class!"
$nadeem->walk("by two lags");//   call walk function by Human/Super  class!"

$dog = new Animal("Dog");   // call constructor by Human/Super  class!"
$dog->speek("Bark"); //   call speek function by Human/Super  class! "
$dog->walk(" by Four legs");  //call walk function by Human/Super  class!
$dog->eat("meat"); //call walk function by Animal/derived class!

?>
```



Polymorphism – this is concerned with having a single form but many different implementation ways. The main purpose of polymorphism is Simplify maintaining applications and making them more extendable. This is an object oriented concept where same function can be used for different purposes. For example function name will remain same but it takes different number of arguments and can do different task.

Encapsulation – The mechanism that binds together data and functions are called encapsulation**.** Encapsulation means that is concerned with hiding the implementation details and only exposing the methods where we encapsulate all the data and member functions together to form an object. The main purpose of encapsulation Reduce software development complexity and protect the internal state of an object.

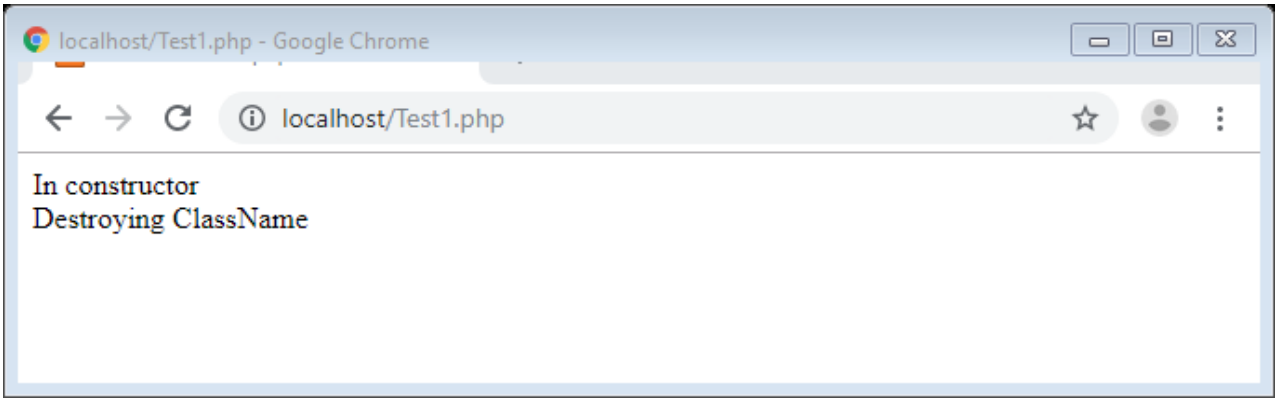
Destructor - refers to a special type of function, which will be called automatically whenever an object is deleted or goes out of scope. PHP 5 introduces a destructor concept similar to that of other object-oriented languages, such as C++. The destructor method will be called as soon as there are no other references to a particular object or in any order during the shutdown sequence. The *'destruct'* method starts with two underscores (__)

Example

```
<?php
class ClassName
{
   function __construct()
   {
      echo "In constructor <br>";
   }

   function __destruct()
  {
      echo "Destroying " . __CLASS__ . "<br>";
   }
}

$obj = new ClassName();
?>
```



One more example of simple program using by OOP coding style in PHP. A create class name students, data member is roll number , name and age with one constructor and six method/ function getter and setter.

Example

```
<?php
class Students {
    // Member variables
          private  $roll;
           public $name;
          public $age;
```

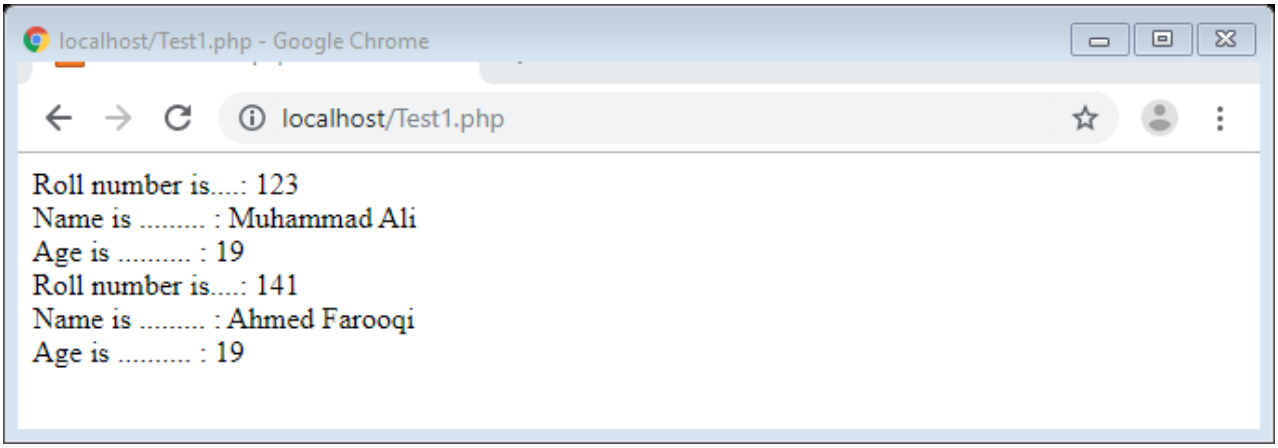
```
  //  Constructor of class
        function __construct($roll=0,$name="nothing", $age=0)
      {
            $this->roll = $roll;
              $this->name = $name;
              $this->age = $age;
      }

    // Member functions
    function setRoll($roll){
      $this->roll = $roll;
    }
    function getRoll(){
      return $this->roll ;
    }
        function setName($name){
      $this->name = $name;
    }
    function getName(){
      return $this->name ;
    }
    function setAge($age){
      $this->age = $age;
    }
    function getAge(){
      return $this->age ;
    }
  }
$Student1 = new Students();
$Student2 = new Students();

$Student1->setRoll(123);
$Student1->setName("Muhammad Ali");
$Student1->setAge(19);
```

```
$Student2->setRoll(141);
$Student2->setName("Ahmed Farooqi");
$Student2->setAge(19);

//#Now you call another member functions to get the values set by in above example
  echo "Roll number is....:  " . $Student1->getRoll() ."<br>";
  echo "Name is ..... :  " . $Student1->getName() ."<br>";
  echo "Age is ..... :  " . $Student1->getAge()  ."<br>";
  echo "Roll number is....:  " . $Student2->getRoll() ."<br>";
  echo "Name is ..... :  " . $Student2->getName() ."<br>";
  echo "Age is ..... :  " . $Student2->getAge()  ."<br>";
?>
```



Exercise

Theory Questions

1. Discuss the benefit of Object-Oriented Programming.
2. Identify benefits of encapsulating.
3. Define the term “instance of a class”.
4. What operator is used to access the methods and properties contained in an Object?
5. Illustrate the syntax used to define a PHP class.
6. Explain the term “garbage collection”
7. Describe the concept of information hiding.
8. List the three levels of access specifiers in PHP.
9. Differentiate between a public access specifier and private access specifier.
10. Describe the purpose of a destructor function.
11. Explain the purpose of the $this reference.
12. What two names may be assigned to a constructor function?

Objective MCQ’s

1. Reusable software object are often referred to as _____
   a) Methods
   b) Components
   c) Widgets
   d) Functions

2. The function associated with an object are called _____.
   a) Properties
   b) Field
   c) Methods
   d) Attributes

3. The term “Black Box” refer to _____
   a) A property
   b) Debugging
   c) Encapsulation
   d) An interface

4. A(n) _____ is an object that has been created from an existing class.
   a) Pattern
   b) Structure
   c) Replica
   d) Instance

5. Which of the following operators is used in member selection notation.
   a) >
   b) ->
   c) =>
   d) .

6. Which of the following functions return the name of the class upon which an object is based?
   a) class_of()
   b) instancceof()
   c) class_name()
   d) get_class()

7. A class that inherits from another class this is also called a _____.
   a) Small class
   b) derived class
   c) Inner class
   d) Outer class

8. This is concerned with having a single form but many different implementation ways.
   a) Polymorphism
   b) Encapsulation
   c) Object
   d) Inheritance

9. When the properties and the methods of the parent class are accessed by the child class, this is called _____.
   a) Polymorphism
   b) Encapsulation
   c) Object
   d) Inheritance

10. These are the function defined inside a class, are used to access object data, and define their behavior that is called _____.
    a) Object
    b) Class
    c) Constructor
    d) Member functions