Advance PHP

Q. What Is Object Oriented Programming?

Ans. **OOP** stands for Object-Oriented Programming.

Procedural programming is about writing procedures or functions that perform operations on the data, while object-oriented programming is about creating objects that contain both data and functions.

**Advantages of OOP:-**

Object-oriented programming has several advantages over procedural programming:

1. OOP is faster and easier to execute.
2. OOP provides a clear structure for the programs.
3. OOP provide securities.
4. OOP helps to keep the PHP code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug.
5. OOP makes it possible to create full reusable applications with less code and shorter development time

Q. What Are Properties Of Object Oriented Systems?

Ans. Properties of object oriented programming are:

1. Class
2. Object
3. Encapsulation
4. Inheritance
5. Access Modifier
6. Polymorphism

**Class:-**

Making group of data member (variable) and member function that called class

A class is defined by using the class keyword, followed by the name of the class and a pair of curly braces ({}). All its properties and methods go inside the braces.

**Object:-**

Classes are nothing without objects! We can create multiple objects from a class. Each object has all the properties and methods defined in the class, but they will have different property values.

Objects of a class are created using the new keyword.

Object is also known as the instance of the class.

**Encapsulation:-**

OOPs concept of Encapsulation in PHP means, enclosing the internal details of the object to protect from external sources. It describes, combining the class, data variables and member function that work on data together within a single unit to form an object. Otherwise, it’s the bundling of properties and behavior in a single class unit.

Data is not accessed directly, in fact, they are accessed through functions (GET or SET) written inside the class. Attributes are kept private but getter (GET) and setter (SET) methods are kept public for manipulation of these attributes.

**Inheritance:-**

It is a concept of accessing the features of one class from another class.

We can reuse our code by using Inheritance.

Types:

1. Single-level inheritance
2. Multi-level inheritance
3. Multiple inheritance
4. Hierarchical Inheritance
5. Hybrid Inheritance

**Access Modifier:-**

Properties and methods can have access modifiers which control where they can be accessed.

­

Types:

1. Public
2. Private
3. Protected

**Polymorphism:-**

This word is can from Greek word poly and morphism.

Poly means "many" and morphism means “property” which helps us to assign more than one property.

There are two ways to achieve polymorphism in PHP:

1. Overloading
2. Overriding

**Note:**  PHP does not support Overloading

**Q.** What Is Difference Between Class And Interface?

Ans.

1. “Class” keyword is used to create class, while “interface” keyword is used to create interface in PHP.
2. “Extends” keyword is used to inherit a class into another class, while “implements” keyword is used to inherit interface into a class.
3. In PHP, class does not support multiple inheritance and interface supports multiple inheritance.

Q. What Is Overloading?

­­­­­

Ans.

Function overloading in PHP allows creating multiple functions with the same name but different implementations, using magic methods to dynamically create properties and methods within a class.

**Note: -** We can’t overload method in PHP.

Function signature are based on their names and do not include argument lists, so you can’t have two functions with the same name.

Q. What are the differences between abstract classes and interfaces?

Ans.

1. Interface can’t have properties, while Abstract class can.
2. All interface functions must be pubic, while abstract class functions is public or protected.
3. Abstract class can have abstract functions as well as non-abstract function, while interface can only have abstract method. All methods in an interface are abstract, so they can’t be implemented in code and the abstract keyword is not necessary.
4. Abstract class does not support multiple inheritance, while interface supports multiple inheritance.
5. Abstract keyword is used to create abstract class, while interface keyword is used to create.
6. Abstract class can inherit other class and interface, while interface can inherit only interface.
7. Extends keyword is used to inherit abstract class, while implements keyword is used to inherit interface.

Q. Define Constructor and Destructor?

Ans.

**Constructor: -**

A constructor allows you to initialize an object's properties upon creation of the object. Also called magic function.

If you create a \_\_construct() function, PHP will automatically call this function when you create an object from a class.

Notice that the construct function starts with two underscores (\_\_).

**Example of constructor:-**

<?php

class abc

{

function simple()

{

echo "Simple Function <br>";

}

function \_\_construct()

{

echo "Magic function run automatically <br>";

}

function autocall()

{

$this->simple(); // normal function call in function with this keywords

abc::\_\_construct(); // call by ::(scope resolution)

}

}

$obj=new abc;

$obj->autocall();

?>

**Destructor:-**

In PHP, destructor method is named as \_\_destruct.

During shutdown sequence too, objects will be destroyed.

Destructor method doesn't take any arguments, neither does it return any data type.

**Example of destructor:-**

<?php

class a

{

public function \_\_construct()

{

echo "I'm alive! <br>";

}

public function \_\_destruct()// object() destroy & call in last

{

echo "I'm dead now <br>";

}

public function display()

{

echo "I'm display now <br>";

}

}

$a = new a();

$a->display();

?>

Q. How to Load Classes in PHP?

Ans.

PHP load classes are used for declaring its object etc. in object oriented applications. PHP parser loads it automatically, if it is registered with spl\_auto\_register() function. PHP parse gets the least chance to load class/interface before emitting an error.

**Syntax:**

spl\_automated\_register(function ($class\_name){

include $class\_name ‘.php’;

});

Q. How to Call Parent Constructor?

Ans.

**Case 1:**

 We can't run directly the parent class constructor in child class if the child class defines a constructor. In order to run a parent constructor, a call to parent::\_\_construct() within the child constructor is required.

**Example:-**

<?php

class my\_constructor

{

public function \_\_construct()

{

echo “Hello Sir I’m Constructor”;

}

}

class abc extends my\_constructor

{

public function \_\_construct()

{

patrent::\_\_construct(); //or my\_constructor::\_\_construct();

echo “I am not constructor sir”;

}

}

$obj = new abc();

?>

**Case 2:**

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

**Example:-**

<?php

class dadaji

{

Public function \_\_construct ()

{

echo “Sir… you are the best”;

}

}

class papa extends dadaji

{

function display ()

{

echo “display function called”;

}

}

$obj =new papa ();

$obj->display();

?>

Q. Are parent constructor called implicitly when creating an object of class?

Ans.

Yes, parent constructor called implicitly when creating an object of class, if the child class does not define a constructor.

But in case, if we create a constructor in child class, then we have to call patrent constructor explicitly by using scope resolution operator (::).

**Example:-**

<?php

class my\_constructor

{

public function \_\_construct()

{

echo “Hello Sir I’m Constructor”;

}

}

class abc extends my\_constructor

{

public function \_\_construct()

{

patrent::\_\_construct(); //or my\_constructor::\_\_construct();

echo “I am not constructor sir”;

}

}

$obj = new abc();

?>

Q. What happen, if constructor is defined as private or protected?

Ans.

**Case 1:-**

If we create constructor as private and create object of child class, then it will show fatal error, because protected data member can be accessed only in own class. They can’t be accessed out of the class.

**Case 2:-**

If we create constructor as protected, then it will be visible for own class and child/inherit class. It will not be accessed out of the class.

If we want to access protected constructor in child class, then we have to create a function or constructor in child and call it with creating instance (object) of child class. In function or constructor we of child class we have to call parent constructor explicitly by using scope resolution operator (::).

**Example:-**

<?php

class parent

{

protected function \_\_construct()

{

echo “Parent constructor called”;

}

}

class child extends parent

{

function \_\_construct()

{

parent::\_\_construct();

echo “Child constructor called”;

}

}

$obj=new child();

?>

Q. What are PHP magic methods/functions? list them. Write program for static keyword in PHP?

Ans. **Magic methods:-**Magic methods are special methods which override PHP's default action when certain actions are performed on an object.

\_\_construct(),

\_\_destruct(),

\_\_call(),

\_\_callStatic(),

\_\_get(),

\_\_set(),

\_\_isset(),

\_\_unset(),

\_\_sleep(),

\_\_wakeup(),

\_\_serialize(),

\_\_unserialize(),

\_\_toString(),

\_\_invoke(),

\_\_set\_state(),

\_\_clone(), and \_\_debugInfo()

**Static keyword:-**

Static properties can be called directly without creating an instance(object) of a class. Static properties are declared with the static keyword.

To access a static property use the class name, scope resolution operator (::), and the property name. Visibility must be public all time.

**Example:-**

class abc

{

public static $my\_static="I am static";

public $simple\_var="I am simple ";

public function static\_fool()

{

echo $this->simple\_var; // normal variable call by $this

echo abc::$my\_static;

//echo self::$my\_static; // fool::$my\_static;

}

}

class xyz extends abc

{

public function static\_bar()

{

echo $this->$simple\_var;

echo abc::$my\_static;

//echo parent::$my\_static;

}

}

echo abc::$my\_static;

?>

Q. Create multiple traits and use it in to a single class?

Ans. Multiple traits are used as multiple inheritance in PHP.

**Example:-**

<?php

trait abc

{

public function test()

{

echo "This is test method";

}

}

trait xyz

{

public function sample()

{

echo "this is sample method";

}

}

class c

{

use abc,xyz; // multiple inheritance

}

$obj=new c();

$obj->test();

$obj->sample();

?>

Q. Write PHP script of object iteration?

Ans. Iteration=Looping

<?php

class myclass {

private $var;

protected $var1;

public $x, $y, $z;

public function \_\_construct() {

$this->var="Hello World";

$this->var1=array(1,2,3);

$this->x=100;

$this->y=200;

$this->z=300;

}

}

$obj = new myclass();

foreach($obj as $key => $value) {

print "$key => $value <br>";

}

?>

Q. Use of the $this keyword.

Ans.   
$this is a reserved keyword in PHP that refers to the calling object. It is usually the object to which the method belongs, but possibly another object if the method is called statically from the context of a secondary object. This keyword is only applicable to internal methods.

Example:-

<?php

Class MyClass

{

Public $mySample=”Hello !”;

Public function printSample()

{

Echo $this->mySample;

}

}

$obj=new MyClass();

$obj->printSample();

?>