MODULE – Advance PHP

• What Is Object Oriented Programming?

Ans:

The word object-oriented is the combination of two words i.e. object and oriented.

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.

• What Are Properties Of Object Oriented Systems?

Ans:

The three main features of OOP are **encapsulation**, **inheritance**, and **polymorphism**.

Encapsulation allows bundling data and methods within a class,

Inheritance enables the creation of new classes based on existing ones,

and polymorphism allows objects of different classes to be treated uniformly through a common interface.

• What Is Difference Between Class And Interface?

Ans:

- **Class :**

  A class is a collection of properties or methods shared by all objects of the same type.

  A class is a blueprint for creating objects. It encapsulates data for the object and methods to manipulate that data. A class can have variables and functions.

❖ **Interface:**

An interface is a contract that defines the methods a class must implement. It does not contain any implementation of these methods, only their signatures.

• What Is Overloading?

Ans:

Function overloading in PHP is used to dynamically create properties and methods.

Function overloading contains same function name and that function performs different task according to number of arguments.

Like other OOP languages function overloading cannot be done by native approach. In PHP function overloading is done with the help of magic function __call().

This function takes function name and arguments.

• What Is T_PAAMAYIM_NEKUDOTAYIM (Scope Resolution Operator (::) with Example?

Ans :

'T_PAAMAYIM_NEKUDOTAYIM' is the token name used internally by the PHP parser for the scope resolution operator :: .

The term "Paamayim Nekudotayim" is derived from Hebrew and literally means "double colon".

Exp :

```php
class MyClass {
    public static $myStaticProperty = "Hello, World!";

    public static function myStaticMethod() {
        echo "Hello from static method!";
    }
}

// Accessing static property
echo MyClass::$myStaticProperty; // Output: Hello, World!

// Accessing static method
MyClass::myStaticMethod(); // Output: Hello from static method!
```

• What are the differences between abstract classes and interfaces?

Ans :

❖ **Abstract class:**

- Abstract class comes under partial abstraction.
- Abstract classes can maintain abstract methods and non abstract methods.
- In abstract classes, we can create the variables.
- In abstract classes, we can use any access specifier.
- By using 'extends' keyword we can access the abstract class features from derived class.

❖ **Interface :**

- Interface comes under fully abstraction.
- Interfaces can maintain only abstract methods.
- In interfaces, we can't create the variables.
- In interface, we can use only public access specifier.
- By using 'implement' keyword we can get interface from derived class.

- By using interfaces multiple inheritance is possible.

• Define Constructor and Destructor?

Ans:

Constructor :

The constructor method inside a class is called automatically on each newly created object.

You can pass as many as arguments you like into the constructor function. The __construct() function doesn't have any return value.

Destructor :

The __destruct() function doesn't have any parameters, neither does it have any return value. The fact that the __destruct() function is automatically called when any object goes out of scope, can be verified by putting var_dump($this) inside the function.

As mentioned above, $this carries the reference to the calling object, the dump shows that the member variables are set to NULL.

• 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_autoload_register() function. PHP parser gets the least chance to load class/interface before emitting an error.

Syntax :

```
spl_autoload_register(function ($class_name) {
  include $class_name . '.php';
});
```

• How to Call Parent Constructor?

Ans:

In order to run a parent constructor, a call to parent::__construct() within the child constructor is required.

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).

• Are Parent Constructor Called Implicitly When Create An ObjectOf Class?

Ans:

Classes which have a constructor method call this method on each newly-created object, so it is suitable for any initialization that the object may need before it is used.

Note: Parent constructors are not called implicitly if the child class defines a constructor.

• What Happen, If Constructor Is Defined As Private Or Protected?

Ans :

Private constructors allow us to restrict the instantiation of a class. Simply put, they prevent the creation of class instances in any place other than the class itself.

Public and **private** constructors, used together, allow control over how we wish to instantiate our classes – this is known as constructor delegation.

• What are PHP Magic Methods/Functions? List them Write program for Static Keyword in PHP?

Ans :

Magic methods in PHP are special methods that start with a double underscore (__) and are automatically invoked in certain situations. They are not explicitly called in the code but are triggered by specific actions.

Here is a list of common magic methods:

- __construct() - Called when an object is created.
- __destruct() - Called when an object is destroyed.
- __get($name) - Called when getting the value of an inaccessible property.
- __set($name, $value) - Called when setting the value of an inaccessible property.
- __isset($name) - Called when calling isset() or empty() on inaccessible properties.
- __unset($name) - Called when unset() is used on inaccessible properties.
- __toString() - Called when an object is treated as a string.

EXP:

```
<?php
class Counter {
    public static $count = 0;
  public static function increment() {
     self::$count++;
  }
```

```
}

// Accessing static property directly through the class
echo "Initial count: " . Counter::$count . "\n";

// Calling static method to increment the count
Counter::increment();
Counter::increment();

// Accessing static property again
echo "Updated count: " . Counter::$count . "\n";
?>
```

• Create multiple Traits and use it in to a single class?

Ans:

```
<?php
/*
PHP only supports single inheritance: a child class can inherit only from one single parent.
OOP traits solve this problem.
Traits are declared with the trait keyword: as class
To use a trait in a class, use the use keyword: // for inheritance Traits are used to declare methods that can be used in multiple classes.
Traits can have methods and abstract methods that can be used in multiple classes, and the methods can have any access modifier (public, private, or protected).
*/
trait first // use trait insted of class
```

```
{
function method1()
{
echo "This is method1.<br>";
}
function method2()
{
echo "This is method2";
}
}
class sample
{
use first; // here use word (use) for inheritance of class first
}
$obj= new sample;
$obj->method1();
$obj->method2();
?>
```

• Write PHP Script of Object Iteration?

Ans :

Object iteration in PHP allows you to loop through an object's properties using a foreach loop. To achieve this, you can implement the Iterator interface or the simpler IteratorAggregate interface.

```php
<?php
class MyClass {
    public $property1 = "Value 1";
    public $property2 = "Value 2";
    public $property3 = "Value 3";
}

$obj = new MyClass();

// Iterate over the object's properties
foreach ($obj as $property => $value) {
    echo "$property => $value\n";
}
?>
```

O/P:

property1 => Value 1

property2 => Value 2

property3 => Value 3

• 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.

<?php

class simple{

public $num = 9;

public function display()

{

```
return $this-> num;
}
}
$obj = new simple();
echo $obj->display();
?>
```

Output:-

9