**MODULE – 4(Advance PHP)**

**OOPS**

**1. What Is Object Oriented Programming?**

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.

**2.What Is Object Oriented Programming?**

Object-oriented programming has several advantages over procedural programming:

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

**3. What Are Properties Of Object Oriented Systems?**

### Properties of Object-Oriented Systems

1. Encapsulation: Bundling data (attributes) and methods (functions) that operate on that data within a single unit called an object.
2. Abstraction: Simplifying complex reality by modeling classes on essential properties and behaviors relevant to the problem.
3. Inheritance: Creating new classes (subclasses) that inherit properties and methods from existing classes (superclasses).
4. Polymorphism: The ability of objects of different types to be treated as if they were of the same type.

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

Class: A blueprint for creating objects. It defines properties and methods.

Interface: A contract that defines methods that a class must implement. It doesn't contain implementation details.

**5. What Is Overloading?**

Overloading refers to the ability to have multiple methods with the same name but different parameters. T

**6. What Is T\_PAAMAYIM\_NEKUDOTAYIM (Scope Resolution Operator (::) with Example**

**The scope resolution operator also known as *Paamayim Nekudotayim* or more commonly known as the double colon is a token that allows access to static, constant, and overridden properties or methods of a class.**

**It is used to refer to blocks or codes in context to classes, objects, etc. An identifier is used with the scope resolution operator. The most common example of the application of the scope resolution operator in PHP is to access the properties and methods of the class.**

**7. What are the differences between abstract classes and interfaces?**

* **Abstract Class**:
* Can have both abstract and concrete methods, member variables, and constructors. Single inheritance.
* **Interface**:
* Only abstract methods (until PHP 8.0), constants, no constructors. Multiple inheritance.

**8. Define Constructor and Destructor?**

Constructor:

* A constructor is called automatically when an object is created from a class.
* It initializes the object’s properties (member variables) and sets their initial values.
* Constructors are essential for defining the blueprint of an object.
* In PHP, you define a constructor using the \_\_construct() method.

Destructor:

* A destructor is automatically called when an object is destroyed (e.g., when it goes out of scope or the script ends).
* It is used for cleanup tasks, such as releasing resources (closing files, database connections, etc.).
* In PHP, you define a destructor using the \_\_destruct() method.

**9. How to Load Classes in PHP?**

Manual Loading:

* Traditionally, you include class files using require\_once or include\_once statements.
* For example, if you have a class named Contact defined in a file named Contact.php, you can load it like this:

require\_once 'models/Contact.php';

$contact = new Contact('[john.doe@example.com](mailto:john.doe@example.com)');

Autoloading with spl\_autoload\_register():

* PHP introduced the spl\_autoload\_register() function to automatically load classes when needed.
* Instead of manually including files, you register an autoloading function that PHP calls when a class is not yet loaded.

function load\_model($class\_name) {

$path\_to\_file = 'models/' . $class\_name . '.php';

if (file\_exists($path\_to\_file)) {

require $path\_to\_file;

}

}

spl\_autoload\_register('load\_model');

**10. How to Call Parent Constructor?**

Automatic Parent Constructor Call:

* When you create an instance of a child class, PHP automatically searches for the constructor in the child class.
* If the child class doesn’t have its own constructor, PHP looks for the constructor in the parent class.
* It then invokes the parent class constructor.

Explicitly Calling Parent Constructor:

* If the child class has its own constructor, you can explicitly call the parent constructor using parent::\_\_construct(arguments).

**11. Are Parent Constructor Called Implicitly When Create An ObjectOf Class?**

**<?php**

**class ParentClass {**

**public function \_\_construct() {**

**echo "Parent constructor called.\n";**

**}**

**}**

**class ChildClass extends ParentClass {**

**public function \_\_construct() {**

**parent::\_\_construct(); // Explicitly call parent constructor**

**echo "Child constructor called.\n";**

**}**

**}**

**$childObj = new ChildClass();**

**?>**

**12.What Happen, If Constructor Is Defined As Private Or Protected?**

**<?php**

**class A{**

**public function \_\_construct(){**

**echo 'Instance of A created';**

**}**

**}**

**class B{**

**protected function \_\_construct(){**

**echo 'Instance of B created';**

**}**

**}**

**class C{**

**private function \_\_construct(){**

**echo 'Instance of C created';**

**}**

**}**

**?>**

**Output:**

new A; // OK

new B; // Fatal error: Call to protected B::\_\_construct() from invalid context

new C; // Fatal error: Call to private C::\_\_construct() from invalid context

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

**MAGIC METHODS:**

**\_\_construct()**

**This method gets called automatically every time the object of a particular class is created.**

**The function of this magic method is the same as the constructor in any OOP language.**

**\_\_destruct()**

**As the name suggests this method is called when the object is destroyed and no longer in use.**

**Generally at the end of the program and end of the function.**

**\_\_call($name,$parameter)**

**\_\_toString()**

**\_\_get($name)**

**\_\_set($name , $value)**

**\_\_debugInfo()**

**Static keyword program:-**

**<?php**

**class MyClass {**

**public static $str = "Hello World!";**

**public static function hello() {**

**echo MyClass::$str;**

**}**

**}**

**echo MyClass::$str;**

**echo "<br>";**

**echo MyClass::hello();**

**?>**

**Output:**

**Hello World!**

**Hello World!**

**14.Create multiple Traits and use it in to a single class?**

**<?php**

**trait message1 {**

**public function msg1() {**

**echo "OOP is fun! ";**

**}**

**}**

**trait message2 {**

**public function msg2() {**

**echo "OOP reduces code duplication!";**

**}**

**}**

**class Welcome {**

**use message1;**

**}**

**class Welcome2 {**

**use message1, message2;**

**}**

**$obj = new Welcome();**

**$obj->msg1();**

**echo "<br>";**

**$obj2 = new Welcome2();**

**$obj2->msg1();**

**$obj2->msg2();**

**?>**

**Output:**

OOP is fun!

OOP is fun! OOP reduces code duplication!

**15.Write PHP Script of Object Iteration?**

**<?php**

**class MyClass {**

**public $var1 = 'value 1';**

**public $var2 = 'value 2';**

**public $var3 = 'value 3';**

**protected $protected = 'protected var';**

**private $private = 'private var';**

**function iterateVisible() {**

**echo "MyClass::iterateVisible:\n";**

**foreach ($this as $key => $value) {**

**print "$key => $value\n";**

**}**

**}**

**}**

**$class = new MyClass();**

**// Iterate through visible properties**

**foreach ($class as $key => $value) {**

**print "$key => $value\n";**

**}**

**// Call custom method for visible properties**

**$class->iterateVisible();**

**?>**

**Output:**

**var1 => value 1**

**var2 => value 2**

**var3 => value 3**

**MyClass::iterateVisible:**

**var1 => value 1**

**var2 => value 2**

**var3 => value 3**

**protected => protected var**

**private => private var**

**16.Use of The $this keyword**

In PHP, $this is a special keyword that represents the instance of the class where it is used. It allows access to the properties and methods of the current object within the class context. $this is only available within methods of a class, and attempting to use it outside of a class context will result in an error.

**• Consider the exercise11 and add a edit link near delete link e.g. Clicking up on edit button a particular row should be open in**

**• editing mode**

**• e.g. on the Particular row there should be filled text box with data and on the option column there should be a confirm button clicking upon it arrow should be updated.**

**• Create Hotel Room Booking System User can book room by 3 ways • Full day**

**• Half day**

**• Custom**

**• If user select for the full day than user only have selection for the checking checkout date**

**• If user select Half day than user have option of date and slot option(like user want to book room for first half – Morning (8AM to 6PM) if user select for second halfit‟s for evening (7PM to Morning 7AM)). Do proper validation like if user can book only available slot. (have touse jQuery -> Ajax, validation, Json passing).**