**Task - Advance PHP**

**OOPS**

* **What Is Object Oriented Programming?**
* Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.
* **What Are Properties Of Object Oriented Systems?**
* 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.

* **What Is Difference Between Class And Interface?**
* Interfaces cannot have properties, while abstract classes can All interface methods must be public, while abstract class methods is public or protected. All methods in an interface are abstract, so they cannot be implemented in code and the abstract keyword is not necessary .Classes can implement an interface while inheriting from another class at the same time.
* **What Is Overloading?**
* Overloading in PHP provides means to dynamically create properties and methods. These dynamic entities are processed via magic methods, one can establish in a class for various action types. All overloading methods must be defined as Public.
* **What Is T\_PAAMAYIM\_NEKUDOTAYIM (Scope Resolution Operator (::) with Example.**
* The Scope Resolution Operator (also called Paamayim Nekudotayim) or in simpler terms, the double colon, is a token that allows access to [static](https://www.php.net/manual/en/language.oop5.static.php), [constant](https://www.php.net/manual/en/language.oop5.constants.php), and overridden properties or methods of a class.
* It's possible to reference the class using a variable. The variable's value can not be a keyword (e.g. self, parent and static).

**Example**:

<?php  
class MyClass {  
    const CONST\_VALUE = 'A constant value';  
}  
  
$classname = 'MyClass';  
echo $classname::CONST\_VALUE;  
  
echo MyClass::CONST\_VALUE;  
?>

* **What are the differences between abstract classes and interfaces?**
* Interface are similar to abstract classes.

The difference between interfaces and abstract classes are:

* Interfaces cannot have properties, while abstract classes can
* All interface methods must be public, while abstract class methods is public or protected
* All methods in an interface are abstract, so they cannot be implemented in code and the abstract keyword is not necessary
* Classes can implement an interface while inheriting from another class at the same time.
* **Define Constructor and Destructor?**

**Constructor** :

* A constructor allows you to initialize an object's properties upon creation of the object.
* 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 (\_\_)!
* Destructor: A destructor is called when the object is destructed or the script is stopped or exited.
* If you create a \_\_destruct() function, PHP will automatically call this function at the end of the script.
* Notice that the destruct function starts with two underscores (\_\_)!
* **How to Load Classes in PHP?**
* 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.
* **How to Call Parent Constructor?**
* 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). $obj = new OtherSubClass();
* **Are Parent Constructor Called Implicitly When Create An ObjectOf Class?**
* **Parent constructors are not called implicitly if the child class defines constructor**. In order to run a parent constructor, a call to parent::\_\_construct() within the child constructor is required.
* **What Happen, If Constructor Is Defined As Private Or Protected?**
* If a constructor is declared as private, then **its objects are only accessible from within the declared class**. You cannot access its objects from outside the constructor class.
* **What are PHP Magic Methods/Functions? List them**
* The following method names are considered magical: \_\_construct(), \_\_destruct(), \_\_call(), \_\_callStatic(), \_\_get(), \_\_set(), \_\_isset(), \_\_unset(), \_\_sleep(), \_\_wakeup(), \_\_serialize(), \_\_unserialize(), \_\_toString(), \_\_invoke(), \_\_set\_state(), \_\_clone(), and \_\_debugInfo().
* **Write program for Static Keyword in PHP?**
* The static keyword is used to declare properties and methods of a class as static. Static properties and methods can be used without creating an instance of the class.
* The static keyword is also used to declare variables in a function which keep their value after the function has ended.
* **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, message2;

}

$obj = new Welcome();

$obj2->msg1();

$obj2->msg2();

* **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();

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

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

}

echo "\n";

$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

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