MODULE-4 (ADVANCED PHP)

- **OOPS**

1) What Is Object Oriented Programming?

Object-oriented programming (OOP) is a computer programming model that organizes software design around data,or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior.

OOP focuses on the objects that developers want to manipulate rather than the logic required to manipulate them. This approach to programming is well-suited for programs that are large, complex and actively updated or maintained. This includes programs for manufacturing and design, as well as mobile applications; for example, OOP can be used for manufacturing system simulation software.

The organization of an object-oriented program also makes the method beneficial to collaborative development, where projects are divided into groups. Additional benefits of OOP include code reusability, scalability and efficiency.

2) What Are Properties Of Object Oriented Systems?

- **The characteristics of object oriented programming are class, objects, encapsulation, inheritance, abstraction, and polymorphism.**

- **class:-**

  class is collection of objects.

- **Object:-**
  It is an instance/example/representative entity of class.

- **Encapsulation:-**
  To enclose in or as if in a capsule. a pilot encapsulated inthe cockpit.

- **Inheritance:-**
  Inheritance refers to the assets that an individual bequeaths to their loved ones after they pass away.

- **Abstraction:-**
  Abstraction is the process of hiding the internal details ofan application from the outer world.

- **Polymorphism:-**

Polymorphism is the ability of any data to be processedin more than one form.

3) What Is Difference Between Class And Interface?

CLASS	INTERFACE
The 'class' keyword is used to create a class.	The 'interface' keyword is used to create an interface.
An object of a class can be created.	An object of an interface cannot be created.
Class doesn't support multiple inheritance.	Interface supports multiple inheritance.
A class can inherit another class.	An Interface cannot inherit a class.
A class can be inherited by another class using the keyword 'extends'.	An Interface can be inherited by a class using the keyword 'implements' and it can be inherited by another interface using the keyword 'extends'.
A class can contain constructors.	An Interface cannot contain constructors.
It cannot contain abstract methods.	It consists of abstract methods only.
Variables and methods can be declared using any specifiers like public, protected, default, private.	Variables and methods are declared as public only.

4) What Is Overloading?

- **Method overloading is a form of polymorphism in OOP.**

Polymorphism allows objects or methods to act in different ways, according to the means in which they are used. One such manner in which the methods behave according to their

argument types and number of arguments is method overloading.

5) What are the differences between abstract classes and interfaces?

- **ABSTRACT CLASSES :**

- **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.**
- **Multiple inheritance is not possible.**

- **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.**

6) Define Constructor and Destructor?

- **CONSTRUCTOR:**

  A constructor is a member function of a class that has the same name as the class name. It helps to initialize the object of a class. It can either accept the arguments or not.It is used to allocate the memory to an object of the class. It is called whenever an instance of the class is created. Itcan be defined manually with arguments or without arguments. There can be many constructors in a class. It can be overloaded but it can not be inherited or virtual.
  There is a concept of copy constructor which is used toinitialize an object from another object.

SYNTAX:

```
ClassName()
{
        //Constructor's Body
}
```

DESTRUCTOR:

Like a constructor, Destructor is also a member function ofa class that has the same name as the class name preceded by a tilde(~) operator. It helps to deallocate the memory of an object. It is called while the object of the class is freed or deleted. In a class, there is always a single destructor without any parameters so it can't be overloaded. It is always called in the reverse order of the constructor. if a class is inherited by another class and boththe classes have a destructor then the destructor of the child class is called first, followed by the destructor of the parent or base class.

SYNTAX:

```
ClassName()
{
        //Destructor's Body
}
```

7) How to Load Classes in PHP?

- **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 .**

8) 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 itmay be inherited from the parent class just like a normal class method (if it was not declared as private).**

9) Are Parent Constructor Called Implicitly When Create An Object Of Class?

- **Classes which have a constructor method call this methodon 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.**

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

- **A private constructor in Java is used in restricting object creation. It is a special instance constructor used in static member-only classes. If a constructor is declared as private, then its objects are only accessible from within the declared class.**

11) What are PHP Magic Methods/Functions? List them .

- **List of Magic Methods in PHP**
- **__construct()**
- **__destruct()**
- **__call($fun, $arg)**
- **__callStatic($fun, $arg)**
- **__get($property)**
- **__set($property, $value)**
- **__isset($content)**
- **__unset($content)**
- **__sleep()**
- **__wakeup()**
- **__toString()**
- **__invoke()**

- **__set_state($array)**
- **__clone()**
- **__debugInfo()**

- ## Write program for Static Keyword in PHP?

Definition and Usage

The static keyword is used to declare properties and methodsof a class as static. Static properties and methods can be used without creating an instance of the class. The static keywordis also used to declare variables in a function which keep theirvalue after the function has ended.

Example:-

```
<!DOCTYPE html>
<html>
<body>

<?php
class MyClass {
  public static $str = "Hello World!";

  public static function hello() {
    echo MyClass::$str;
  }
}
```

```
echo  MyClass::$str;
echo "<br>";

echo MyClass::hello();

?>


</body>

</html>
```

Output:-

```
Hello World!Hello
World!
```

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

```
<?php

  /*
```

PHP only supports single inheritance:

a child class can inherit only from one single parent. So, what if a class needs to inherit multiple behaviors?

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();

?>
```

13) 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 1var2
=> value 2var3 =>
value 3

MyClass::iterateVisible:
var1 => value 1
var2 => value 2var3
=> value 3
```

protected => protected var
private => private var

14) 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.

```
<?php
class simple{

        public $k = 9;

        public function display()
        {
                return $this->k;
         }
}

$obj = new simple();
echo $obj->display();

?>
```

Output:-

9