MODULE – 4(Advance PHP)

Q1. What Is Object Oriented Programming?

Ans . Php is a server side scripting language. Php used for website development and software development .

Oops concepts :

1 class

2 object

3 Inheritance

4 Polymorphism

5 Overloading

6 Data Abstraction

7 Constructor

8 Destructor

Q2. What Are Properties Of Object Oriented Systems?

Ans.

Object oriented have many properties are available some properties are:

1. **Encapsulation:**
2. **Abstraction:**
3. **Inheritance:**
4. **Polymorphism:**
5. **Modularity:**
6. **Message Passing:**

Q3. What Is Difference Between Class And Interface?

Ans.

The main difference between a class and an interface is that a class describe of an object. An interface contains the behaviors assigned and defined by the class.

**1** **Class:** A class is a blueprint for creating objects. It defines the attributes (data fields) and behaviors (methods) that objects of that class will have.

Classes can have constructors for initializing object properties, member variables to store data, and member functions (methods) to perform operations on the data.

Objects are instances of classes. When you create an object, you are creating a specific instance of that class with its own set of data and behavior.

**2 Interface:** An interface is a contract that defines a set of methods that a class must implement. It specifies what methods a class should have without providing the implementation details.

Interfaces do not contain any code for method implementation. They only declare the method signatures (names, parameters, and return types) that implementing classes must define.

Classes can implement one or multiple interfaces, which ensures that they provide the required functionality specified by those interfaces.

Q4. What Is Overloading?

Ans . 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.

The overloading methods are invoked when interacting with properties or methods that have not been declared or are not visible in the current scope. The rest of this section will use the terms inaccessible properties and inaccessible methods to refer to this combination of declaration and visibility.

All overloading methods must be defined as public.

\_\_set() is run when writing data to inaccessible (protected or private) or non-existing properties.

\_\_get() is utilized for reading data from inaccessible (protected or private) or non-existing properties.

\_\_isset() is triggered by calling [isset()](https://www.php.net/manual/en/function.isset.php) or [empty()](https://www.php.net/manual/en/function.empty.php) on inaccessible (protected or private) or non-existing properties.

\_\_unset() is invoked when unset() is used on inaccessible (protected or private) or non-existing properties.

The $name argument is the name of the property being interacted with. The \_\_set() method's $value argument specifies the value the $name'ed property should be set to.

Property overloading only works in object context. These magic methods will not be triggered in static context. Therefore these methods should not be declared static. A warning is issued if one of the magic overloading methods is declared static.

Q5. What Is T\_PAAMAYIM\_NEKUDOTAYIM (Scope Resolution Operator (::) with Example

Ans. The Scope Resolution Operator (also called Paamayim Nekudotayim) or in simpler terms, the double colon, is a token that allows access to a constant, static property, or static method of a class or one of its parents. Moreover, static properties or methods can be overriden via late static binding.

When referencing these items from outside the class definition, use the name of the 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).

Paamayim Nekudotayim would, at first, seem like a strange choice for naming a double-colon. However, while writing the Zend Engine 0.5 (which powers PHP 3), that's what the Zend team decided to call it. It actually does mean double-colon - in Hebrew!

Q6 . What are the differences between abstract classes and interfaces?

Ans.   
Abstract classes and interfaces are both important concepts in object-oriented programming, but they serve different purposes and have distinct characteristics. Here are the key differences between abstract classes and interfaces:

|  |  |
| --- | --- |
| abstract classes | interfaces |

|  |  |
| --- | --- |
| 1. Abstract class does not support multiple inheritances. 2. Abstract class does contain a data member. 3. The abstract class supports containers. 4. Abstract class contains both incomplete(i.e. abstract) and complete members. 5. An abstract class can contain access modifiers within subs, functions, and properties. 6. Only a complete member of the abstract class can be static. | 1. Interface class supports multiple inheritance feature 2. This does not contain a data member. 3. The interface does not allow containers. 4. An interface class only contains incomplete members which refer to the signature of the member. 5. Since everything is assumed to be public, an interface class does not have access modifiers by default. 6. Any member of an interface cannot be static. |

Q7. Define Constructor and Destructor?

Ans. PHP allows developers to declare constructor methods for classes. 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.

\_\_construct()

Constructors are ordinary methods which are called during the instantiation of their corresponding object. As such, they may define an arbitrary number of arguments, which may be required, may have a type, and may have a default value. Constructor arguments are called by placing the arguments in parentheses after the class name.

PHP possesses a destructor concept similar to that of other object-oriented languages, The destructor method will be called as soon as there are no other references to a particular object, or in any order during the shutdown sequence.

\_\_destruct()

Like constructors, parent destructors will not be called implicitly by the engine. In order to run a parent destructor, one would have to explicitly call parent::\_\_destruct() in the destructor body. Also like constructors, a child class may inherit the parent's destructor if it does not implement one itself.

The destructor will be called even if script execution is stopped using exit(). Calling exit() in a destructor will prevent the remaining shutdown routines from executing.

Q8. How to Load Classes in PHP?

Ans.

Many developers writing object-oriented applications create one PHP source file per class definition. One of the biggest annoyances is having to write a long list of needed includes at the beginning of each script (one for each class).

The spl\_autoload\_register() function registers any number of autoloaders, enabling for classes and interfaces to be automatically loaded if they are currently not defined. By registering autoloaders, PHP is given a last chance to load the class or interface before it fails with an error.

Any class-like construct may be autoloaded the same way. That includes classes, interfaces, traits, and enumerations.

Q9. How to Call Parent Constructor?

Ans. If we want to run a parent constructor , to call a parent::\_\_construct() within the child constructor is required. If the child does not define a constructor then it may be inherit from parent class just like a normal class method .$obj = new OtherSubClass();

Parent constructors are not called implicitly 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. 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).

Q10. Are Parent Constructor Called Implicitly When Create An ObjectOf Class?

Ans. Parent constructors are not called implicitly 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.

If we want to run a parent constructor , to call a parent::\_\_construct() within the child constructor is required. If the child does not define a constructor then it may be inherit from parent class just like a normal class method .$obj = new OtherSubClass();

Q11. What are PHP Magic Methods/Functions? List them Write program for Static

Keyword in PHP?Ans. Magic methods are special methods which override PHP's default's action when certain actions are performed on an object.

The following method names are considered magical:

\_\_construct()

\_\_destruct()

\_\_call()

\_\_callStatic()

\_\_get()

\_\_set()

\_\_isset()

\_\_unset()

\_\_sleep()

\_\_wakeup()

\_\_serialize()

\_\_unserialize()

\_\_toString()

\_\_invoke()

\_\_set\_state()

\_\_clone()

\_\_debugInfo()

Q12. Create multiple Traits and use it in to a single class?

Traits are a mechanism for code reuse in single inheritance languages such as PHP. A Trait is intended to reduce some limitations of single inheritance by enabling a developer to reuse sets of methods freely in several independent classes living in different class hierarchies. The semantics of the combination of Traits and classes is defined in a way which reduces complexity, and avoids the typical problems associated with multiple inheritance and Mixins.

A Trait is similar to a class, but only intended to group functionality in a fine-grained and consistent way. It is not possible to instantiate a Trait on its own. It is an addition to traditional inheritance and enables horizontal composition of behavior; that is, the application of class members without requiring inheritance.

Q13. Write PHP Script of Object Iteration?

Ans. PHP provides a way for objects to be defined so it is possible to iterate through a list of items, with, for example a foreach statement. By default, all visible properties will be used for the iteration.

Q14. Use of The $this keyword

Ans. The pseudo-variable $this is available when a method is called from within an object context. $this is the value of the calling object.

The $this keyword refers to the calling object. It’s essentially a temporary variable that is replaced with the object name to refer to itself.

$this is used with the -> operator to reference a member. The member cannot have a $ symbol as prefix when being referenced.