Unit-III Apply Object Oriented Concepts in PHP	3a Write constructor and destructor functions for the given problem in PHP. 3b Implement inheritance to extend the given base class. 3c Use overloading / overriding to solve the given problem. 3d Clone the given object.	3.1 Creating Classes and Objects 3.2 Constructor and Destructor 3.3 Inheritance, Overloading and Overriding, Cloning Object. 3.4 Introspection, Serialization

Total Marks:16

3.1 Classes and Objects

What is Object Oriented Programming

Object-Oriented Programming (OOP) is a programming model that is based on the concept of classes and objects. As opposed to procedural programming where the focus is on writing procedures or functions that perform operations on the data, in object-oriented programming the focus is on the creations of objects which contain both data and functions together.Object-oriented programming has several advantages over conventional or procedural style of programming.

The most important ones are listed below:

It provides a clear modular structure for the programs.

It helps you adhere to the "don't repeat yourself" (DRY) principle, and thus make your code much easier to maintain, modify and debug.It makes it possible to create more complicated behavior with less code and shorter development time and high degree of reusability.

Let's assume we have a class named Fruit. A Fruit can have properties like name, color, weight, etc. We can define variables like $name, $color, and $weight to hold the values of these properties.When the individual objects (apple, banana, etc.) are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

Define a Class

A class is defined by using the class keyword, followed by the name of the class and a pair of curly braces ({}). All its properties and methods go inside the braces:

Syntax

```
<?php
class Fruit {
 // code goes here...
```

```
}
?>
```

Below we declare a class named Fruit consisting of two properties ($name and $color) and two methods set_name() and get_name() for setting and getting the $name property:

Example

```
<?php
class Fruit {
  // Properties
  public $name;
  public $color;

  // Methods
  function set_name($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
}
?>
```

Note: In a class, variables are called properties and functions are called methods!

Define Objects

Classes are nothing without objects! We can create multiple objects from a class. Each object has all the properties and methods defined in the class, but they will have different property values.

Objects of a class are created using the new keyword.In the example below, $apple and $banana are instances of the class Fruit:

Example

```
<?php
class Fruit {
  // Properties
  public $name;
  public $color;

  // Methods
  function set_name($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
```

```
}

$apple = new Fruit();
$banana = new Fruit();
$apple->set_name('Apple');
$banana->set_name('Banana');

echo $apple->get_name();
echo "<br>";
echo $banana->get_name();
?>
```

In the example below, we add two more methods to class Fruit, for setting and getting the $color property:

Example

```
<?php
class Fruit {
  // Properties
  public $name;
  public $color;

  // Methods
  function set_name($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
  function set_color($color) {
    $this->color = $color;
  }
  function get_color() {
    return $this->color;
  }
}

$apple = new Fruit();
$apple->set_name('Apple');
$apple->set_color('Red');
echo "Name: " . $apple->get_name();
echo "<br>";
echo "Color: " . $apple->get_color();
?>
```

PHP - The $this Keyword

The $this keyword refers to the current object, and is only available inside methods.

Look at the following example:

Example

```
<?php
class Fruit {
  public $name;
}
$apple = new Fruit();
?>
```

So, where can we change the value of the $name property? There are two ways:

1. Inside the class (by adding a set_name() method and use $this):

Example

```
<?php
class Fruit {
  public $name;
  function set_name($name) {
    $this->name = $name;
  }
}
$apple = new Fruit();
$apple->set_name("Apple");
?>
```

2. Outside the class (by directly changing the property value):

Example

```
<?php
class Fruit {
  public $name;
}
$apple = new Fruit();
$apple->name = "Apple";
?>
```

PHP - instanceof

You can use the instanceof keyword to check if an object belongs to a specific class:

Example

```
<?php
$apple = new Fruit();
var_dump($apple instanceof Fruit);
?>
```

this keyword is used inside a class, generally within the member functions to access non-static members of a class(variables or functions) for the current object.

Let's take an example to understand the usage of $this.

```
<?php
   class Person {
      // first name of person
      private $name;

      // public function to set value for name (setter method)
      public function setName($name) {
         $this->name = $name;
      }

      // public function to get value of name (getter method)
      public function getName() {
         return $this->name;
      }
   }

   // creating class object
   $john = new Person();

   // calling the public function to set fname
   $john->setName("John Wick");

   // getting the value of the name variable
   echo "My name is " . $john->getName();

?>
```

My name is John Wick

In the program above, we have created a private variable in the class with name $name and we have two public methods setName() and getName() to assign a new value to $name variable and to get its value respectively.

Whenever we want to call any variable of class from inside a member function, we use $this to point to the current object which holds the variable.

We can also use $this to call one member function of a class inside another member function.

NOTE: If there is any static member function or variable in the class, we cannot refer it using the $this.

Accessing Properties and Methods

Once you have an object, you can use the -> notation to access methods and properties of theobject:

```
$object->propertyname
$object->methodname([arg, ... ])
```

For example:

```
printf("Ram is %d years old.\n", $ram->age);  // property access
$ram->birthday();                    // method call
$ram->set_age(21);                    // method call with arguments
```

3.2 Constructor and Desctructor

To create and initialize a class object in a single step, PHP provides a special method called as Constructor, which is used to construct the object by assigning the required property values while creating the object.

And for destroying the object, the Destructor method is used.

Syntax for defining Constructor and Destructor

__construct() and __destruct().

```
<?php
   class <CLASS_NAME> {

      // constructor
      function __construct() {
         // initialize the object properties
      }

      // destructor
      function __destruct() {
         // clearing the object reference
      }
   }
?>
```

Constructor can accept arguments, whereas destructors won't have any argument because a destructor's job is to destroy the current object reference.

PHP Constructor:-Let's take the example of a class Person which has two properties, fname and lname, for this class we will define a constructor for initialising the class properties(variables) at the time of object creation.

```
<?php
   class Person {
      // first name of person
      private $fname;
```

```
    // last name of person
    private $lname;

    // Constructor
    public function __construct($fname, $lname) {
       echo "Initialising the object...<br/>";
       $this->fname = $fname;
       $this->lname = $lname;
    }

    // public method to show name
    public function showName() {
       echo "The Legend of India: " . $this->fname . " " . $this->lname;
    }
  }

  // creating class object
  $j = new Person("Bhagat", "Singh");
  $j->showName();

?>
```

While earlier, we were using the -> operator to set values for the variables or used the setter methods, in case of a constructor method, we can assign values to the variables at the time of object creation.If a class has a constructor then whenever an object of that class is created, the constructor is called.

PHP Destructor:-PHP Destructor method is called just before PHP is about to release any object from its memory. Generally, you can close files, clean up resources etc in the destructor method. Let's take an example,

```
<?php
  class Person {
    // first name of person
    private $fname;
    // last name of person
    private $lname;

    // Constructor
    public function __construct($fname, $lname) {
       echo "Initialising the object...<br/>";
       $this->fname = $fname;
       $this->lname = $lname;
    }

    // Destructor
```

```
    public function __destruct(){
      // clean up resources or do something else
      echo "Destroying Object...";
    }

    // public method to show name
    public function showName() {
      echo "The Legend of India: " . $this->fname . " " . $this->lname . "<br/>";
    }
  }

  // creating class object
  $j = new Person("Swami", "Vivekananda");
  $j->showName();

?>
```

As we can see in the output above, as the PHP program ends, just before it PHP initiates the release of the object created, and hence the destructor method is called.The destructor method cannot accept any argument and is called just before the object is deleted, which happens either when no reference exist for an object or when the PHP script finishes its execution.

3.3 Inheritance

PHP - What is Inheritance?
Inheritance in OOP = When a class derives from another class.The child class will inherit all the public and protected properties and methods from the parent class. In addition, it can have its own properties and methods.An inherited class is defined by using the extends keyword.

Syntax for Inheriting a Class
In PHP, extends keyword is used to specify the name of the parent class while defining the child class. For example,

```
<?php
  class Human {
    // parent class code
  }

  class Male extends Human {
    // child class code
  }

  class Female extends Human {
    // child class code
  }
?>
```

Some important points to remember while using inheritance are:

Child class can access and use only the non-private properties and methods on the parent class.Child class can have it's own methods too, which will not be available to the parent class. Child class can also override a method defined in the parent class and provide its own implementation for it.Let's look at an example:

Example

```
<?php
class car {
  public $name;
  public $color;
  public function __construct($name, $color) {
    $this->name = $name;
    $this->color = $color;
  }
  public function intro() {
    echo "The car is {$this->name} and the color is {$this->color}.<br>";
  }

  /*protected function print(){
    echo "Tata motor also famous in World";
  }*/
}

class maruti extends car {
  public function message() {
    echo " Maruti in an Indian company of car<br>";
  }
}
$d = new maruti("swift dzire", "red");
$d->message();
$d->intro();
//$d->print();
?>
```

The maruti class is inherited from the car class.This means that the maruti class can use the public $name and $color properties as well as the public_construct() and intro() methods from the car class because of inheritance.The maruti class also has its own method: message().

3.3.Overloading

1. Function overloading or method overloading is the ability to create multiple functions of the same name with different implementations depending on the type of their arguments.
2. Overloading in PHP provides means to dynamically create properties and methods.

3. These dynamic entities are processed via magic methods, one can establish in a class for various action types.
4. The overloading methods are invoked when interacting with properties or methods that have not been declared or are not visible in the current scope
5. All overloading methods must be defined as Public.
6. After creating an object for a class, we can access a set of entities that are properties or methods not defined within the scope of the class.
6. Such entities are said to be overloaded properties or methods, and the process is called overloading.
7. For working with these overloaded properties or functions, PHP magic methods are used.
8. Most of the magic methods will be triggered in object context except __call() method which is used in dynamic context.
9. __call() and ___callStatic() are called when somebody is calling a nonexistent object method in object or static context.
public __call (string $name , array $arguments) : mixed
public static __callStatic (string $name , array $arguments) : mixed

10. __call() is triggered when invoking inaccessible methods in an object context.

__callStatic() is triggered when invoking inaccessible methods in a static context.

The $name argument is the name of the method being called. The $arguments argument is an enumerated array containing the parameters passed to the $name'ed method.

Example #2 Overloading methods via the __call()

```
<?php
class Foo {

   public function __call($method, $args) {

      if ($method === 'findSum') {
         echo 'Sum is calculated to ' . $this->getSum($args)."<br>";
      } else {
         echo "Called method $method<br>";
      }
   }
private function getSum($args) {
      $sum = 0;
      foreach ($args as $arg) {
         $sum += $arg;
      }
      return $sum;
   }
}
$foo = new Foo;
```

```
$foo->bar1(); // Called method bar1
$foo->bar2(); // Called method bar2
$foo->findSum(10, 50, 30); //Sum is calculated to 90
$foo->findSum(10.75, 101); //Sum is calculated to 111.75
?>
```

3.3. Overriding

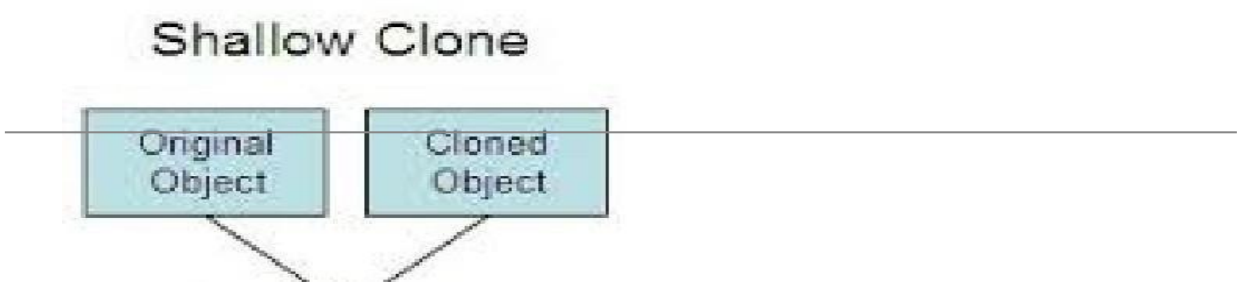
1. It is the same as other OOPs programming languages.
2. In this function, both parent and child classes should have the same function name and number of arguments.
3. It is used to replace the parent method in child class.
4. The purpose of function overriding is to change the behavior of the parent class method.
5. The two functions with the same name and the same parameter are called function overriding.

E.g

```
<?php
class aicte {
function helloWorld() {
echo "Parent"."<br>";
}
}
class msbte extends aicte {
function helloWorld() {
echo "\nChild";
}
}
$p = new aicte;
$c= new msbte;
$p->helloWorld();
$c->helloWorld();
?>
```

Cloning Object

1. The clone keyword is used to create a copy of an object.
2. If any of the properties was a reference to another variable or object, then only the reference is copied.
3. Objects are always passed by reference, so if the original object has another object in its properties, the copy will point to the same object.
4. This behavior can be changed by creating a __clone() method in the class.



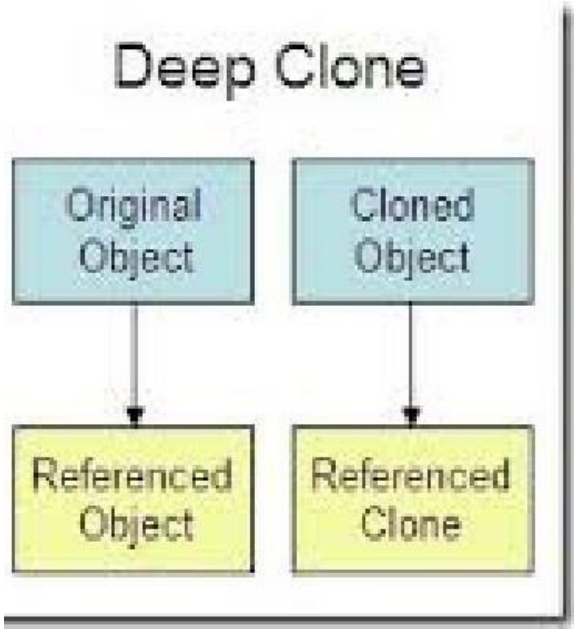
In **shallow copy** a new object is created.
The new object is an exact copy of the value in the original object.
It calls the object's "_clone()" method.
It simply makes a copy of the reference to A to B.
It is copy of A's address.
The addresses of A and B will be same ie. they will be pointing to the same memory location.

```
<?php
class MyClass {
  public $amount;
}

// Create an object with a reference
$value = 5;
$obj = new MyClass();
$obj->amount = &$value;

// Clone the object
$copy = clone $obj;
/ Change the value in the original object
$obj->amount = 6;

// The copy is changed
print_r($copy);
?>
```



In this the data is actually completely copies.
In this everything is duplicated and all values are copies into a new instances.
Advantage of deep copy is that the A & B do not depend on each other but the process is relatively slower and more expensive.
In shallow copy B points to object A's memory location whereas in deep copy all things in object A's memory location get copied to object B's location.

```
<?php
class MyClass {
  public $amount;
  public function __clone() {
    $value = $this->amount;
    unset($this->amount); // Unset breaks references
    $this->amount = $value;
  }
}

// Create an object with a reference
$value = 5;
$obj = new MyClass();
$obj->amount = &$value;
// Clone the object
$copy = clone $obj;

// Change the value in the original object
$obj->amount = 6;

// The copy is not changed
print_r($copy);
echo "<br>";
```

```
print_r($obj);
?>
```

3.4. Introspection

Introspection is the ability of a program to examine an object characteristics such as its name,parent class,properties and method.
Introspection allow us to:

1. Obtain the name of the class to which an object belongs as well as its member properties and method
2. write generic debuggers,serializers,profilers
3. Introspection in PHP offers the useful ability to examine classes, interfaces, properties and method with introspection we can write code that operates on any class or object
4. To examining classes the introspective function provided by PHP are class_exits(),get_class_method(),get_class_vars() etc

1. class_exists():

This function is used to determine whether a class exists.It takes a string and return a boolean value.

Syntax-$yes_no=class_exists(classname);

This function returns TRUE if classname is a defined class, or FALSE

2. get_class_method()

This function returns the names of the class methods.

3. get_parent_class():return the class name of an object parent class

4.is_subclass_of():check whether an object has parent class.

```
<?php

if(class_exists('cwipedia'))
{
 $ob=new  cwipedia();
 echo "This is cwipedia.in";
}
else
{
 echo "Not exist";
}

?>
```

output:Not exist

```
<?php
class vesp
{
  function a()
  {
 echo "hey CO6I";
} }
if(class_exists('vesp'))
{
 $ob=new vesp();
 echo "This is ves.ac.in";
 echo "<br>";
 echo $ob->a();
}
else
{
 echo "Not exist";

}

?>
```

Output:

hey CO6I

"This is ves.ac.in

3.4. serialize

The serialize() function converts a storable representation of a value.

Syntax

serialize(*value*);

To serialize data means to convert a value to a sequence of bits, so that it can be stored in a file, a memory buffer, or transmitted across a network.

```
<?php

  Example

// Complex array
```

```
$myvar = array(
   'hello',
   42,
   array(1, 'two'),
   'apple'
);

// Convert to a string
$string = serialize($myvar);

// Printing the serialized data
echo $string;

?>
```

Output:

```
a:4:{i:0;s:5:"hello";i:1;i:42;i:2;a:2:{i:0;i:1;i:1;s:3:"two";}i:3;s:5:"apple";}
```

Unserialize()

Unserialize() Function: The unserialize() is an inbuilt function php that is used to unserialize the given serialized array to get back to the original value of the complex array, $myvar.

Syntax:

```
unserialize( $serialized_array )
```

Below program illustrate both serialize() and unserialize() functions:

Program:

```
<?php
  // Complex array
$myvar = array(
   'hello',
   42,
   array(1, 'two'),
   'apple'
);
// Serialize the above data
$string = serialize($myvar);
  // Unserializing the data in $string
$newvar = unserialize($string);
  // Printing the unserialized data
print_r($newvar);
  ?>
```

Output:

```
Array
(
   [0] => hello
   [1] => 42
```

```
    [2] => Array
        (
            [0] => 1
            [1] => two
        )

    [3] => apple
)
```