MODULE – 4

TOPIC – ADVANCE PHP

TASK: OOPS

- 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.

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

- What are properties of object-oriented systems?

An object-oriented system revolves around a Class and objects. A class is used to describe characteristics of any entity of the real world. An object is a pattern of the class. An actual object created at runtime is called as an instance. A class, apart from characteristics has some functions to perform called as methods. For example, A class named “Food” has attributes like ‘price’, ‘quantity’. “Food” class has methods like Serve_food(), bill_food().

Object: Objects in Object Oriented Systems interact through messages.

Inheritance: The main class or the root class is called as a Base Class. Any class which is expected to have ALL properties of the base class along with its own is called as a Derived class. The process of deriving such a class is Derived class. For the “Food” class, a Derived class can be “Class Chinesefood”.

Abstraction: Abstraction is creating models or classes of some broad concept. Abstraction can be achieved through Inheritance or even Composition.

Encapsulation: Encapsulation is a collection of functions of a class and object. The “Food” class is an encapsulated form. It is achieved by specifying which class can use which members (private, public, protected) of an object.

Polymorphism: Polymorphism means existing in different forms. Inheritance is an example of Polymorphism. A base class exists in different forms as derived classes. Operator overloading is an example of Polymorphism in which an operator can be applied in different situations.

- What is difference between class and interface?

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

- 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.
  - After creating object for a class, we can access set of entities that are properties or methods not defined within the scope of the class.
  - Such entities are said to be overloaded properties or methods, and the process is called as overloading.
  - For working with these overloaded properties or functions, PHP magic methods are used.

- What is T_PAAMAYIM_NEKUDOTAYIM (scope resolution operator (::) with example)?

The Scope Resolution Operator (also called T_PAAMAYIM_NEKUDOTAYIM) or in simpler terms, the double colon, is a token that allows access to static, constant and overridden properties or methods of a class.

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

Example:

```php
<?php
class MyClass {
    const CONST_VALUE = 'A constant value';
}

$classname = 'MyClass';
echo $classname::CONST_VALUE;

echo MyClass::CONST_VALUE;
?>
```

Output:

```
A constant valueA constant value
```

- What are the differences between abstract classes and interfaces?

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

- Define constructor and destructor?

Constructor:

In object-oriented programming terminology, constructor is a method defined inside a class is called automatically at the time of creation of object. Purpose of a constructor method is to initialize the object. In PHP, a method of special name **__construct** acts as a constructor.

Destructor:

Destructor is a method automatically as soon as garbage collector fins that a particular object has no more references. In PHP, destructor method is named as **__destruct**. During shutdown sequence too, objects will be destroyed. Destructor method doesn't take any arguments, neither does it return any data type.

- How to load classes in PHP?

  - PHP can load class files automatically on demand (No explicit require statements are needed).
  - The file name must match the case of the terminating class name (each class in a separate file).
  - The directory name must match the case of the namespace names.
  - __autoload() has been DEPRECATED as of PHP 7.2.0. Relying on this feature is highly discouraged.

- How to call parent constructor?

We will face two cases while calling the parent constructor method in child class.

Case 1:

We can't run directly the parent class constructor in child class 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.

Example:

```
<?php
   class grandpa{
      public function __construct(){
         echo "I am in Tutorials Point"."<br>";
      }
   }
   class papa extends grandpa{
      public function __construct(){
         parent::__construct();
         echo "I am not in Tutorials Point";
      }
   }
$obj = new papa();
?>
```

Output:

```
I am in Tutorials Point
I am not in Tutorials Point
```

Case 2:

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).

Example:

```
<?php
   class grandpa{
      public function __construct(){
         echo "I am in Tutorials point";
      }
   }
   class papa extends grandpa{
   }
   $obj = new papa();
?>
```

Output:

```
I am in Tutorials point
```

- Are parent constructor called implicitly when create an object of class?
  - 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).

- What happens, if constructor is defined as private or protected?
  - The constructor may be made private or protected to prevent it from being called externally.
  - If so, **only a static method will be able to instantiate the class**.
  - Because they are in the same class definition they have access to private methods, even if not of the same object instance.
  - The private constructor is optional and may or may not make sense depending on the use case.

- What are PHP magic methods/functions? List them.

Magic methods in PHP are **special methods that are aimed to perform certain tasks**. These methods are named with double underscore (__) as prefix. All these function names are reserved and can't be used for any purpose other than associated magical functionality. Magical method in a class must be declared public. These methods act as interceptors that are automatically called when certain conditions are met.

Following magical methods are currently available in PHP.

__sleep

```
public __sleep (void): array
```

serialize () method in class checks if it has a function name __sleep (). If so, that function is executed prior to any serialization. It is supposed to return an array with the names of all variables of that object that should be serialized.

__wakeup

```
public __wakeup (void): void
```

unserialize () method checks there exists a function with the magic name __wakeup (). If present, this function can reconstruct any resources that the object may have.

__serialize

```
public __serialize (void): array
```

serialize () method also checks if the class has __serialize () method. If so, it is executed prior to any serialization. It must construct and return an associative array of key/value pairs that represent the serialized form of the object.

__unserialize

```
public __unserialize (array $data): void
```

unserialize () also checks for if __unserialize () is present, and it will be passed the restored array that was returned from __serialize (). It may then restore the properties of the object from that array as appropriate

__toString

```
public __toString (void): string
```

The __toString () method describes string representation of object. For example, what echo $obj; will print. This method must return a string

__invoke

```
__invoke ([$...]): mixed
```

This method is called when a script tries to call an object as a function.

__set_state

```
static __set_state (array $properties): object
```

This static method is called for classes exported by var_export (). It receives one parameter which is an array containing exported properties in the form array ('property' => value, ...).

__debugInfo

```
__debugInfo (void): array
```

This method is automatically called when var_dump () is executed for dumping an object to get the properties that should be shown. If it isn't defined, all public, protected and private properties will be shown.

__set

```
public __set (string $name, mixed $value): void
```

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

__get

```
public __get (string $name): mixed
```

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

__isset

```
public __isset (string $name): bool
```

__isset () is triggered by calling isset () or empty () on inaccessible (protected or private) or non-existing properties.

__unset

```
public __unset (string $name): void
```

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

- 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.

Example:

```php
<?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!
```

- Create multiple traits and use it in to a single class?

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). Traits are declared with the trait keyword.

Example:

```php
<?php
trait message1 {
  public function msg1() {
    echo "OOP is fun! "."<br>";
  }
}

trait message2 {
  public function msg2() {
    echo "OOP reduces code duplication!";
  }
}

class Welcome {
  use message1, message2;
}


$obj = new Welcome();
$obj->msg1();
$obj->msg2();

?>
```

Output:

```
OOP is fun!
OOP reduces code duplication!
```

- Write PHP script of object iteration?

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.

Example:

```php
<?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:<br>";
       foreach ($this as $key => $value) {
           print "$key => $value<br>";
       }
    }
}

$class = new MyClass();

foreach($class as $key => $value) {
    print "$key => $value<br>";
}
echo "<br>";


$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.

Example:

```php
<?php
class simple{

    public $k = 9;

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

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

?>
```

Output:

9

- Consider the exercise1 and add edit link near delete link e.g. Clicking up on edit button a particular row should be open in.
    1. Editing mode.
    2. 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.

Code:

Folder Name: MVC

Sub Folder Name: Model

- **File Name: CommonModel.php**

```php
<?php
class CommonModel{
    function __construct(){
        $this->conn = new mysqli('localhost','root','','project');
    }
    function select_all($tbl){
        $sw="SELECT * FROM $tbl";
        $run=$this->conn->query($sw);
        if($run->num_rows>0){

        while ($course = $run->fetch_object()) {
            $row[]=$course;
        }
        return $row;
        }else{
            return [];
        }

    }
    function select_where($tbl,$data){
        $sw="SELECT * FROM $tbl WHERE ";

        foreach ($data as $key => $value) {
            $sw.=$key.'='."'$value'";
        }

        $run=$this->conn->query($sw);
        $r=$run->fetch_object();
        return $r;
    }
```

```php
function updateData($tbl,$data,$id)
{
    $upd="UPDATE $tbl SET ";
    $count=count($data);
    $i=1;
    foreach($data as $key => $value){
        if($count>$i){
        $upd.=$key.' = '."'$value'".',';
        }else{
        $upd.=$key.' = '."'$value'";
        }$i++;
    }
    $WHERE=' WHERE ';
    foreach($id as $key=>$value){
        $WHERE.=$key.'='.$value;
    }
    $this->conn->query($upd.$WHERE);
}

function save($tbl,$data){
    foreach ($data as $key => $value) {
        $k[]=$key;
        $v[]=$value;
    }
    $key=implode(',',$k);
    $val=implode("','",$v);
    $ins="INSERT INTO $tbl($key) VALUES('$val')";

    $run=$this->conn->query($ins);
    return $run;
```

```php
    }
    function deleteData($tbl,$id){
        $del="DELETE FROM $tbl WHERE ";

        foreach($id as $key=>$value){
            $del.=$key.'='.$value;
        }

        $this->conn->query($del);

    }

}
?>
```

Sub Folder Name: View

- **File Name: Home.php**

```
<center>
<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-6">
            <h3 class="w3-border-bottom w3-border-light-grey w3-padding-16"><strong>Course Information</strong></h3>

             <a href="<?php echo $this->url.'Controller/AdminController.php/addcourse' ?>" class="w3-bar-item w3-button w3-black">Add
             Course</a> <br><br>
            <table border class="table">
                <thead>
                    <tr>
                        <th scope="col">Id</th>
                        <th scope="col">Name</th>
                        <th scope="col">Image</th>
                        <th scope="col">Action</th>
                    </tr>
                </thead>
                <tbody>
                    <?php foreach($course as $p) { ?>
                        <tr>
                        <th scope="row">><?php echo $p->id ?></th>
                        <th scope="row">><?php echo $p->name ?></th>
                        <th scope="row"><img src="<?php echo $this->url.'Uploads/'.$p->image ?>" width="50"></th>
                        <td>
                            <a href="<?php echo $this->url.'Controller/AdminController.php/deleteCourse?eid=
                            '.$p->id ?>" class="w3-bar-item w3-button w3-red">DELETE</a>
                            <a href="<?php echo $this->url.'Controller/AdminController.php/editCourse?eid=
                            '.$p->id ?>" class="w3-bar-item w3-button w3-blue">EDIT</a>
                        </td>

                        </tr>
                    <?php } ?>
                </tbody>
            </table>
        </div>
    </div>
</div>
</center>
```

- **File Name: addcourse.php**

```
<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-6">
        <form method="POST" enctype="multipart/form-data">
            <h3 class="w3-border-bottom w3-border-light-grey w3-padding-16"><strong><center>
            Course</center></strong></h3>
            <div class="mb-3">
                <label for="exampleInputEmail" class="form-label">Name</label>
                <input type="text" class="w3-input" name="name" aria-describedby="emailHelp">
            </div>
            <div class="mb-3">
                <label for ="exampleInputPassword1" class="form-label">Image</label>
                <input type="file" class="w3-input" name="image">
            </div><br>
            <div class="mb-3">
                <input name="<?php if(isset($_GET['eid'])){echo 'update';}else{echo 'save';} ?>" type="submit" class="w3-button w3-right
                w3-black w3-theme" value="<?php if(isset($_GET['eid'])){echo 'Update';}else{echo 'Save';} ?>">
            </div>
        </form>
    </div>
</div>
</div>
```

Sub Folder Name: Controller

- **File Name: AdminController.php**

```php
<?php

include('../Model/CommonModel.php');

class AdminController extends CommonModel{
    function __construct(){
        parent::__construct();
        session_start();
        $this->url='http://localhost:7882/project/MVC/';
    }

    function Home(){
        $course=$this->select_all('db_course');
        include('../View/Admin/Home.php');
    }

    function DeleteCourse(){
        if(isset($_GET['eid'])){
            $d=$_GET['eid'];
            $this->deleteData('db_course',['id'=>$d]);
            header('location:'.$this->url.'Controller/AdminController.php');
        }
    }


    function EditCourse(){
        if(isset($_GET['eid'])){
            $e=$_GET['eid'];
            $course=$this->select_where('db_course',['id'=>$e]);

            if(isset($_POST['update'])){
```

```
            $fname=$_FILES['image']['name'];
            $temp=$_FILES['image']['tmp_name'];
            $path='../Uploads/'.$fname;
            move_uploaded_file($temp, $path);
            $name=$_POST['name'];
            $data=array('name'=>$name,'image'=>$fname);

            $this->updateData('db_course',$data,['id'=>$e]);
            }
        }
        include('../View/Admin/addcourse.php');

    }


    function AddCourse(){
        if(isset($_POST['save'])){
            $fname=$_FILES['image']['name'];
            $temp=$_FILES['image']['tmp_name'];
            $path='../Uploads/'.$fname;
            move_uploaded_file($temp, $path);
            $name=$_POST['name'];
            $data=array('name'=>$name,'image'=>$fname);
            $this->save('db_course',$data);
        }
        include('../View/Admin/addcourse.php');
    }
    function PageNotFound(){
        include('../View/PageNotFound.php');
    }


}
```

```php
$obj=new AdminController;
if (isset($_SERVER['PATH_INFO'])) {
    $p=$_SERVER['PATH_INFO'];
}
else{
    $p='/home';

}

switch ($p) {
    case '/home':
    $obj->Home();
    break;

    case '/addcourse':
    $obj->AddCourse();
    break;

    case '/deleteCourse':
    $obj->DeleteCourse();
    break;

    case '/editCourse':
    $obj->EditCourse();
    break;

    default:
    $obj->PageNotFound();
    break;
}


?>
```

Output:

When we click on Add Course button.

Now we can add courses in table and database.

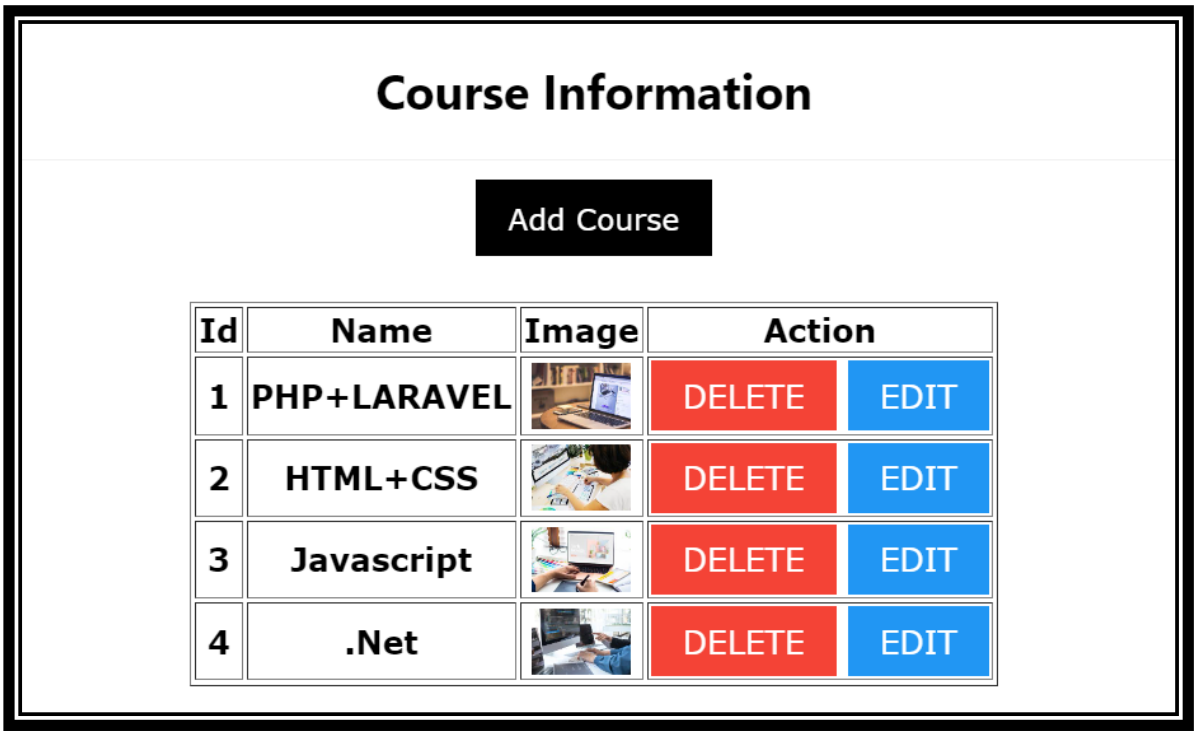

Table: db_course

id	name	image
1	PHP+LARAVEL	i1.jpg
2	HTML+CSS	i2.jpg
3	Javascript	i7.jpg
4	.Net	i5.jpg

When you click on EDIT button.

Now we can edit course.

Course Information

Add Course

Id	Name	Image	Action	
1	PHP+LARAVEL		DELETE	EDIT
2	CSS		DELETE	EDIT
3	Javascript		DELETE	EDIT
4	.Net		DELETE	EDIT

Records at 2nd id gets updated

When you click on DELETE button records will be deleted.

Course Information

Add Course

Id	Name	Image	Action	
1	PHP+LARAVEL		DELETE	EDIT
2	CSS		DELETE	EDIT
4	.Net		DELETE	EDIT

Records at 3rd id gets deleted.

Table: db_course

id	name	image
1	PHP+LARAVEL	i1.jpg
2	CSS	l4.jpg
4	.Net	i5.jpg