**Module 2 – Advanced PHP Excercises**

OOPs Concepts

THEORY EXERCISE:

Define Object-Oriented Programming (OOP) and its four main principles: Encapsulation, Inheritance, Polymorphism, and Abstraction.

### -> ****Definition of OOP****

Object-Oriented Programming (OOP) is a programming paradigm that organizes software design around objects. An object is a real-world entity that has **attributes (data)** and **methods (functions)**.  
OOP makes programs **modular, reusable, maintainable, and easier to understand**.

Four Main Principles of OOP

· **Encapsulation** → Wrapping data and methods together in a class. Data is hidden and accessed only through methods.

· **Inheritance** → One class (child) can reuse properties and methods of another class (parent).

· **Polymorphism** → Same method name can perform different tasks (overloading/overriding).

· **Abstraction** → Hiding implementation details and showing only essential features.

Practical Exercise:

Create a simple class in PHP that demonstrates encapsulation by using private and public properties and methods.

<?Php

Class BankAccount {

Private $balance;

Public function\_construct($amount) {

$this->balance =$amount;

}

Public function desposit($amount){

If ($amount>0) {

$this->balance+=$amount;

Echo”Deposite:$amount<br>”;

}

}

Public function withdrow($amount) {

If ($amount >0&& $amount <=$this->balance) {

$this->balance-=$amount;

Echo”withdrawn:$amount<br>”;

}

}

public function getBalance() {

return $this->balance;

}

}

// Object creation

$account = new BankAccount(1000);

$account->deposit(500);

$account->withdraw(200);

echo "Current Balance: " . $account->getBalance();

?>

**Deposited: 500  
Withdrawn: 200  
Current Balance: 1300 output**

Class

THEORY EXERCISE: 

Explain the structure of a class in PHP, including properties and methods

In php,a class is a blueprint for creating objects.it defines the properties (variables) and methods (functions) that describe the behavior and data of the objects.

**Structure of class in php**

1. Class keyword and Class Name

A class is defined using the class keyword followed by the class name.

1. Properties (Variables)

-> These are variables that hold data.

-> They can be declared as public, private, or protected.

1. Methods (Functions)

-> These are functions defined inside a class.

-> They define the behavior of the object.

1. Objects

An object is created from a class using the new keyword.

**Practical Exercise: **

Write a PHP script to create a class representing a "Car" with properties like make, model, and year, and a method to display the car details.

<?Php

Class car {

// Properties

public $make;

public $model;

public $year;

public function \_\_construct($make, $model, $year) {

$this->make = $make;

$this->model = $model;

$this->year = $year;

}

public function displayCarDetails() {

echo "Car Make: " . $this->make . "<br>";

echo "Car Model: " . $this->model . "<br>";

echo "Car Year: " . $this->year . "<br>";

}

}

// Object creation

$car1 = new Car();

$car1->displayCarDetails();

?>

**OUTPUT**

**Car Make: Toyota**

**Car Model: Corolla**

**Car Year: 2022**

Object

THEORY EXERCISE: 

What is an object in OOP? Discuss how objects are instantiated from classes in PHP.

An **object** is a real-world entity created from a **class**.

While a **class** is only a blueprint or template, an **object** is an actual instance of that blueprint.

Each object has its own **properties (data)** and can use the **methods (functions)** defined in the class.

### ****Object Instantiation in PHP****

In PHP, an object is created from a class using the new keyword.

This process is called **instantiation**.

Once an object is created, we can access class properties and methods using the -> operator.

**Practical Exercise: **

Instantiate multiple objects of the "Car" class and demonstrate how to access their properties and methods.

<?php

class Car {

public $make;

public $model;

public $year;

/ Method to display car details

public function displayCarDetails() {

echo "Car Make: " . $this->make . "<br>";

echo "Car Model: " . $this->model . "<br>";

echo "Car Year: " . $this->year . "<br><br>";

}

}

// First object

$car1 = new Car();

$car1->make = "Toyota";

$car1->model = "Corolla";

$car1->year = 2022;

$car1->displayCarDetails();

// second object

$car1 = new Car();

$car1->make = "Honda";

$car1->model = "Civic";

$car1->year = 2021;

$car1->displayCarDetails();

// Thired object

$car3 = new Car();

$car3->make = "Maruti";

$car3->model = "Swift";

$car3->year = 2023;

$car3->displayCarDetails();

?>

Extends

THEORY EXERCISE: 

Explain the concept of inheritance in OOP and how it is implemented in PHP.

**Inheritance** is an OOP concept where one class (child/derived class) acquires the **properties** and **methods** of another class (parent/base class).

**Types of inheritance**

· **Single Inheritance:** One child inherits from one parent.

· **Multilevel Inheritance:** A class inherits from a child class which is itself derived from another class.

· **Hierarchical Inheritance:** Multiple child classes inherit from the same parent.

**Practical Exercise:** 

Create a "Vehicle" class and extend it with a "Car" class. Include properties and methods inboth classes, demonstrating inherited behavior.

<?Php

// parent class

Class vehicle {

Public $brand;

Public $year;

// Method in parent class

public function start() {

echo "The vehicle is starting...<br>";

}

public function stop() {

echo "The vehicle is stopping...<br>";

}

}

// Child class

class Car extends Vehicle {

public $model;

// Method in child class

public function displayDetails() {

echo "Car Brand: " . $this->brand . "<br>";

echo "Car Model: " . $this->model . "<br>";

echo "Manufacturing Year: " . $this->year . "<br>";

}

}

// Object of Car class

$car1 = new Car();

$car1->brand = "Toyota";

$car1->year = 2022;

$car1->model = "Corolla";

// Calling methods

$car1->start();

$car1->displayDetails();

$car1->stop();

?>

**Ouput:**

**The vehicle is starting...**

**Car Brand: Toyota**

**Car Model: Corolla**

**Manufacturing Year: 2022**

**The vehicle is stopping...**

Overloading

THEORY EXERCISE:

Discuss method overloading and how it is implemented in PHP

**Method Overloading** in OOP means having multiple methods in the same class with the **same name but different parameters** (number or type).

It allows a class to handle different kinds of data inputs with the same method name.

In languages like **Java**, method overloading is directly supported by defining

multiple methods with the same name but different parameter lists.

**Method Overloading in PHP**

PHP **does not support traditional method overloading** (like Java or C++).

But PHP provides a way to achieve similar behavior using **magic methods**:

\_\_call($name, $arguments) → Handles calls to inaccessible or undefined methods.

\_\_callStatic($name, $arguments) → Handles calls to inaccessible or undefined static methods.

**Practical Exercise: **

Create a class that demonstrates method overloading by defining multiple methods withthesame name but different parameters.

<?php

class Calculator {

// Magic method for method overloading

public function \_\_call($name, $arguments) {

if ($name == "add") {

$count = count($arguments);

if ($count == 2) {

return $arguments[0] + $arguments[1];

} elseif ($count == 3) {

return $arguments[0] + $arguments[1] + $arguments[2];

} else {

return "Invalid number of arguments!";

}

}

}

}

// Object creation

$calc = new Calculator();

// Demonstrating method overloading

echo "Sum of 10 and 20: " . $calc->add(10, 20) . "<br>";

echo "Sum of 10, 20 and 30: " . $calc->add(10, 20, 30) . "<br>";

echo "Test with 4 numbers: " . $calc->add(10, 20, 30, 40);

?>

**Abstraction Interface**

THEORY EXERCISE: 

Explain the concept of abstraction and the use of interfaces in PHP

· **Definition:** Abstraction in OOP is the process of hiding the internal details of implementation and showing only the essential features to the user.

· It allows the programmer to **focus on what an object does** instead of how it does it.

· In PHP, abstraction can be achieved using **abstract classes** and **interfaces**.

### ****Abstract Classes****

An **abstract class** cannot be instantiated directly.

It can have both **abstract methods (without body)** and **regular methods (with body)**.

Child classes must **implement all abstract methods**.

**Practical Exercise: **

Define an interface named VehicleInterface with methods like start(), stop(), andimplement this interface in multiple classes.

<?php

// Interface definition

interface VehicleInterface {

public function start();

public function stop();

// Class Car implementing the interface

}

class Car implements VehicleInterface {

public function start() {

echo "Car is starting...<br>";

}

public function stop() {

echo "Car is stopping...<br>";

}

}

// Class Bike implementing the interface

class Bike implements VehicleInterface {

public function start() {

echo "Bike is starting...<br>";

}

public function stop() {

echo "Bike is stopping...<br>";

}

}

// Object of Car

$car = new Car();

$car->start();

$car->stop();

// Object of Bike

$bike = new Bike();

$bike->start();

$bike->stop();

?>

**Constructor**

THEORY EXERCISE:

 What is a constructor in PHP? Discuss its purpose and how it is used.