**Experiment 9**

**Title :** Experiment Based on advanced OOP concepts in PHP

**Aim :** To make student understand the advanced OOP concepts using PHP and usage of the same

**Advanced OOP concepts**

Following are the advanced Object Oriented Concepts in PHP:-

* **Inheritance -** When a class derives from another class it is called as inheritance. The child class will inherit all the 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.
* **Access Modifiers -** Properties and methods can have access modifiers which control where they can be accessed.

There are three access modifiers:

**public** - the property or method can be accessed from everywhere. This is default

**protected** - the property or method can be accessed within the class and by classes derived from that class

**private** - the property or method can ONLY be accessed within the class

* **Constants -** Class constants can be useful if you need to define some constant data within a class. A class constant is declared inside a class with the const keyword. A constant cannot be changed once it is declared. Class constants are case-sensitive. However, it is recommended to name the constants in all uppercase letters. We can access a constant from outside the class by using the class name followed by the scope resolution operator (::) followed by the constant name.
* **Abstract Classes -** Abstract classes and methods are when the parent class has a named method, but need its child class(es) to fill out the tasks. An abstract class is a class that contains at least one abstract method. An abstract method is a method that is declared, but not implemented in the code. An abstract class or method is defined with the abstract keyword.
* **Interfaces -** Interfaces allow you to specify what methods a class should implement. Interfaces are declared with the interface keyword. A class that implements an interface must implement all of the interface's methods.

*The difference between interfaces and abstract classes are:*

* Interfaces cannot have properties, while abstract classes can
* All interface methods must be public, while abstract class methods is public or protected
* All methods in an interface are abstract, so they cannot be implemented in code and the abstract keyword is not necessary
* Classes can implement an interface while inheriting from another class at the same time
* **Static methods -** Static methods can be called directly - without creating an instance of the class first.
* Static methods are declared with the static keyword
* To access a static method use the class name, double colon (::), and the method name
* A static method can be accessed from a method in the same class using the self keyword and double colon (::)
* Static methods can also be called from methods in other classes. To do this, the static method should be public
* To call a static method from a child class, use the parent keyword inside the child class. Here, the static method can be public or protected
* **Static properties -** Static properties can be called directly - without creating an instance of a class.
* Static properties are declared with the static keyword
* A static property can be accessed from a method in the same class using the self keyword and double colon
* To call a static property from a child class, use the parent keyword inside the child class

<?php

// Abstract class

abstract class Media {

protected $title;

protected $author;

const MEDIA\_TYPE = 'General';

public function \_\_construct($title, $author) {

$this->title = $title;

$this->author = $author;

}

abstract public function getDetails();

public static function getMediaType() {

return static::MEDIA\_TYPE;

}

}

// Interface

interface Borrowable {

public function borrow($borrower);

}

// Inheritance

class Book extends Media implements Borrowable {

private $isbn;

public static $totalBooks = 0;

const MEDIA\_TYPE = 'Book';

public function \_\_construct($title, $author, $isbn) {

parent::\_\_construct($title, $author);

$this->isbn = $isbn;

self::$totalBooks++;

}

public function getDetails() {

return "Title: {$this->title}, Author: {$this->author}, ISBN: {$this->isbn}, Type: " . self::getMediaType();

}

public function borrow($borrower) {

return "{$borrower} has borrowed the book: {$this->title}";

}

}

class Magazine extends Media implements Borrowable {

private $issueNumber;

public static $totalMagazines = 0;

const MEDIA\_TYPE = 'Magazine';

public function \_\_construct($title, $author, $issueNumber) {

parent::\_\_construct($title, $author);

$this->issueNumber = $issueNumber;

self::$totalMagazines++;

}

public function getDetails() {

return "Title: {$this->title}, Author: {$this->author}, Issue: {$this->issueNumber}, Type: " . self::getMediaType();

}

public function borrow($borrower) {

return "{$borrower} has borrowed the magazine: {$this->title}";

}

}

// Usage

$libraryItems = [

new Book("1984", "George Orwell", "123456789"),

new Magazine("National Geographic", "Various", "2023-09")

];

foreach ($libraryItems as $item) {

echo $item->getDetails() . PHP\_EOL;

echo $item->borrow("John Doe") . PHP\_EOL;

}

// Accessing static properties

echo "Total Books: " . Book::$totalBooks . PHP\_EOL;

echo "Total Magazines: " . Magazine::$totalMagazines . PHP\_EOL;

?>

**Conclusion:-**

Thus, we have studied, understood and practically checked Advanced OOP concepts in php.