**EXPERIMENT NO. 04**

**Title:** OOP concepts in PHP

**Aim:** To understand and Apply the Object Oriented Concepts in PHP Language.

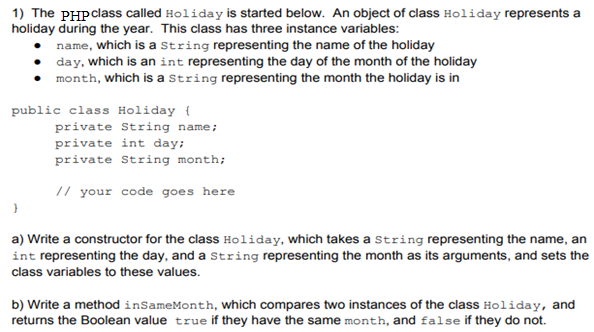
**Objective:** To implement the program on OOP Concepts in PHP

**Contents:**

* **Creating Class and Objects**
* **Accessing member function**
* **Access Specifiers in PHP**
* **Constructor**
* **Encapsulation**
* **Inheritance**
* **Interface**
* **Abstract Class**

**Problem Statements to perform in lab:**

**Problem Stmt1 : Class, Object and constructor**

****

<?php

class Holiday

{

    private String $name;

    private int $day;

    private String $month;

    function \_\_construct($name, $day, $month)

    {

        $this->name = $name;

        $this->day = $day;

        $this->month = $month;

    }

    function insamemonth($holiday2)

    {

        return $this->month == $holiday2->month;

    }

}

$holiday1 = new Holiday('DS', 15, 'Aug');

$holiday2 = new Holiday('ABC', 10, 'Aug');

if ($holiday1->insamemonth($holiday2)) {

    echo "The holidays are in Same month";

} else {

    echo "The holidays are not in same month";

}

?>

**Problem Stmt2 : Inheritance**

Base class Parent with static variable bank\_balance=1000 and two methods i.e Deposit() and Withdraw().

Method implementation is two argument, no return type where the money value is either deposited or withdraw from bank\_balance.

Create child classes i.e Son inherits the Parent class.

Create the 2 object and call methods Deposit() and Withdraw()

Check final bank balance at end. ( as per ur Deposit() and Withdraw)

<?php

class Bank{

    public static $bank\_balance = 1000;

    function Deposit($amt){

        self::$bank\_balance += $amt;

    }

    function withdraw($amt){

        self::$bank\_balance -= $amt;

    }

}

class Son extends Bank{

    // Inherits methods from Parent class

}

$son1 = new Son();

$son1->Deposit(1000);

$son1->withdraw(500);

echo "Final bank balance: " . Bank::$bank\_balance . "\n";

?>

**Problem Stmt3 : Abstract Class**

## Create an abstract class named Fruit, 3 child classes extending the abstract class namely: Apple, Orange, Grape. In these child classes, define the color function so that it prints Apple is red for the Apple class, Orange is orange for the Orange class and Grape is purple for the Grape class.

<?php

abstract class Fruit {

    protected $name;

    public function \_\_construct($name) {

        $this->name = $name;

    }

    abstract public function color();

}

class Apple extends Fruit {

    public function color() {

        echo $this->name . " is red.\n";

    }

}

class Orange extends Fruit {

    public function color() {

        echo $this->name . " is orange.\n";

    }

}

class Grape extends Fruit {

    public function color() {

        echo $this->name . " is purple.\n";

    }

}

$apple = new Apple("Apple");

$orange = new Orange("Orange");

$grape = new Grape("Grape");

$apple->color();

$orange->color();

$grape->color();

?>

**Problem Stmt4 : Interface**

## Write a PHP class called 'Shape' with an abstract method 'calculateArea()'. Create two subclasses, 'Triangle' and 'Rectangle', that implement the 'calculateArea()' method.

<?php

interface Shape

{

    public function calculateArea();

}

class Triangle implements Shape

{

    private $base;

    private $height;

    public function \_\_construct($base, $height)

    {

        $this->base = $base;

        $this->height = $height;

    }

    public function calculateArea()

    {

        return 0.5 \* $this->base \* $this->height;

    }

}

class Rectangle implements Shape

{

    private $width;

    private $height;

    public function \_\_construct($width, $height)

    {

        $this->width = $width;

        $this->height = $height;

    }

    public function calculateArea()

    {

        return $this->width \* $this->height;

    }

}

$triangle = new Triangle(10, 5);

echo "Area of the triangle: " . $triangle->calculateArea() . "<br>";

$rectangle = new Rectangle(10, 5);

echo "Area of the rectangle: " . $rectangle->calculateArea();

**Conclusion:**

Write the crux of the experiment. Here crux means very important point, complicated and again it should be understandable by you. Do not write conclusion like thus we have studied the ……… in PHP programming.