**EXPERIMENT NO. 04**

**Title:** OOP concepts in PHP

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

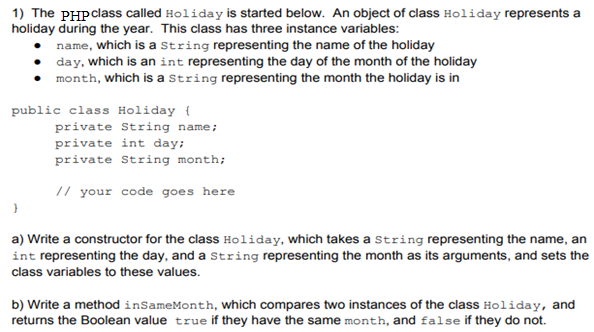
**Objective:** To implement the program on OOP Concepts in PHP

**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 $name;

    private $day;

    private $month;

*// Constructor to initialize the Holiday object*

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

*$this*->name = $name;

*$this*->day = $day;

*$this*->month = $month;

    }

*// Method to compare if two holidays are in the same month*

    public function inSameMonth($otherHoliday) {

        return *$this*->month === $otherHoliday->month;

    }

*// Getter methods for the class variables*

    public function getName() {

        return *$this*->name;

    }

    public function getDay() {

        return *$this*->day;

    }

    public function getMonth() {

        return *$this*->month;

    }

}

*// Example usage:*

$holiday1 = new Holiday("Christmas", 25, "December");

$holiday2 = new Holiday("New Year", 1, "January");

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

    echo "The holidays are in the same month.";

} else {

    echo "The holidays are in different months.";

}

?>

**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)

*<!-- 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 ParentClass {

    protected static $bank\_balance = 1000;

    public function deposit($amount) {

        if ($amount > 0) {

            self::$bank\_balance += $amount;

            echo "Deposited: $amount, New Balance: " . self::$bank\_balance . "\n";

        } else {

            echo "Invalid deposit amount.\n";

        }

    }

    public function withdraw($amount) {

        if ($amount > 0 && $amount <= self::$bank\_balance) {

            self::$bank\_balance -= $amount;

            echo "Withdrawn: $amount, New Balance: " . self::$bank\_balance . "\n";

        } else {

            echo "Invalid withdraw amount or insufficient balance.\n";

        }

    }

    public static function getBankBalance() {

        return self::$bank\_balance;

    }

}

class Son extends ParentClass {

*// Inherits deposit and withdraw methods from ParentClass*

}

*// Create two objects of Son class*

$son1 = new Son();

$son2 = new Son();

*// Perform deposit and withdraw operations*

$son1->deposit(500);

$son2->withdraw(300);

$son1->deposit(200);

$son2->withdraw(400);

*// Check final bank balance*

echo "Final Bank Balance: " . ParentClass::getBankBalance() . "\n";

?>

**Problem Stmt3 : Abstract Class**

## Create an abstract class named Fruit, with a constructor function that gets the name of the fruit and an abstract function, color, that prints the color of the fruit. Create 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.

*<!-- Create an abstract class named Fruit, with a constructor function*

*that gets the name of the fruit and an abstract function, color, that prints the color of the fruit.*

*Create 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 \_\_construct() {

        parent::\_\_construct("Apple");

    }

    public function color() {

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

    }

}

class Orange extends Fruit {

    public function \_\_construct() {

        parent::\_\_construct("Orange");

    }

    public function color() {

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

    }

}

class Grape extends Fruit {

    public function \_\_construct() {

        parent::\_\_construct("Grape");

    }

    public function color() {

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

    }

}

*// Create objects of each fruit class*

$apple = new Apple();

$orange = new Orange();

$grape = new Grape();

*// Call the color method for each object*

$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.

*<!-- Write a PHP class called 'Shape' with an abstract method 'calculateArea()'.*

*Create two subclasses, 'Triangle' and 'Rectangle', that implement the 'calculateArea()' method. -->*

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Shape Area Calculator</title>

</head>

<body>

    <h1>Calculate Area of Shapes</h1>

    <form method="post" action="">

        <h2>Triangle</h2>

        <label for="triangleBase">Base:</label>

        <input type="number"  name="triangleBase" required>

        <label for="triangleHeight">Height:</label>

        <input type="number"  name="triangleHeight" required>

        <h2>Rectangle</h2>

        <label for="rectangleWidth">Width:</label>

        <input type="number"  name="rectangleWidth" required>

        <label for="rectangleHeight">Height:</label>

        <input type="number"  name="rectangleHeight" required>

        <br><br>

        <input type="submit" value="Calculate">

    </form>

    <?php

    if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

        interface Shape {

            abstract 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;

            }

        }

*// Get user input*

        $triangleBase = $\_POST['triangleBase'];

        $triangleHeight = $\_POST['triangleHeight'];

        $rectangleWidth = $\_POST['rectangleWidth'];

        $rectangleHeight = $\_POST['rectangleHeight'];

*// Create shape objects*

        $triangle = new Triangle($triangleBase, $triangleHeight);

        $rectangle = new Rectangle($rectangleWidth, $rectangleHeight);

*// Display the calculated areas*

        echo "<h2>Results</h2>";

        echo "Triangle area: " . $triangle->calculateArea() . "<br>";

        echo "Rectangle area: " . $rectangle->calculateArea() . "<br>";

    }

    ?>

</body>

</html>

**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.