OOP في لغة الـ PHP

دكتور المادة /

إبراهيم الشامي

اعداد الطالب /

عبدالمنعم فهد السوادي

2025/2024

المقدمة :-

البرمجة الكائنية التوجه (OOP) هي أسلوب برمجي يقوم على تصميم البرامج باستخدام كائنات .

الكائنات تمثل كيانات تحتوي على بيانات خصائص (Properties - وسلوكيات) دوال.(Methods -

PHP بدأت دعم OOP منذ الإصدار 4، وتم تحسينه في PHP 5 بشكل كبير. ومع PHP 8 ، تم تقديم تحسينات جديدة لجعل البرمجة الكائنية أكثر قوة وسهولة.

فوائد الـ OOP في لغة الـ PHP :-

1. **إعادة استخدام الكود** :البرمجة الكائنية تسهل إعادة استخدام الكود من خلال الوراثة (Inheritance) وإعادة استخدام الكائنات.
2. **تحسين التنظيم** :الكائنات والفئات تجعل الكود أكثر تنظيمًا وأسهل في القراءة والصيانة.
3. **القابلية للتوسع** :يمكن إضافة ميزات جديدة بسهولة باستخدام الميزات الكائنية مثل الوراثة والتعدد.(Polymorphism)
4. **تقليل التكرار** :يمكن إنشاء دوال وخصائص عامة تستخدمها جميع الكائنات.
5. **مناسبة للمشاريع الكبيرة** :تسهل إدارة المشاريع الكبيرة باستخدام الكائنات والفئات.

أساسيات الـ OOP في لغة الـ PHP :-

1. الفئات :-

الفئة (Class) هي قالب لإنشاء الكائنات.

الكائن (Object) هو نسخة من الفئة يتم إنشاؤها باستخدام الكلمة المفتاحية.

```php
<?php
class Car {
    public $brand;
    public $color;
    public function start() {
        echo "السيارة تعمل.";
    }
}

$myCar = new Car();
$myCar->brand = "Toyota";
$myCar->color = "Red";
$myCar->start();  // الإخراج: السيارة تعمل.
?>
```

2. **الخصائص :-**

الخصائص تمثل البيانات الخاصة بالكائن.

يتم تعريفها داخل الفئة باستخدام الكلمات المفتاحية public, protected, أوprivate.

3. **الدوال:-**

الدوال تمثل السلوكيات أو الأفعال التي يمكن أن ينفذها الكائن.

كلمات التحكم في الوصول:-

1. **PUBLIC:-**

يمكن الوصول إلى الخاصية أو الدالة من أي مكان

2. **PRODEECTED :-**

يمكن الوصول إليها فقط من داخل الفئة أو الفئات الموروثة

3. **PRIVATE:-**

يمكن الوصول إليها فقط من داخل الفئة نفسها.

```php
<?php
class Car {
    public $brand;
    protected $engine;
    private $serialNumber;

    public function setSerialNumber($serial) {
        $this->serialNumber = $serial;
    }

    public function getSerialNumber() {
        return $this->serialNumber;
    }
}
?>
```

المفاهيم الأساسية في OOP :-

1 - الوراثة :-

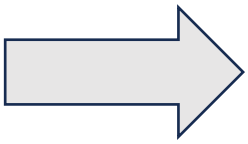
تتيح للفئة الوراثة من فئة أخرى.

```php
<?php
class Vehicle {
    public $type;
    public function move() {
        echo "الطائرة تتحرك.";
    }
}
class Car extends Vehicle {
    public $brand;
    public function honk() {
        echo "السيارة تطلق البوق.";
    }
}
$car = new Car();
$car->move(); // الإخراج: الطائرة تتحرك.
$car->honk(); // الإخراج: السيارة تطلق البوق.
?>
```

2 - الحمل الزائد :-

السماح للكائنات من أنواع مختلفة باستخدام نفس الدوال بطرق مختلفة.

```php
<?php
class Shape {
    public function draw() {
        echo "رسم الشكل الأساسي.";
    }
}
class Circle extends Shape {
    public function draw() {
        echo "رسم دائرة.";
    }
}
```



```php
class Square extends Shape {
    public function draw() {
        echo "رسم مربع.";
    }
}
$shapes = [new Circle(), new Square()];

foreach ($shapes as $shape) {
    $shape->draw();
}
?>
```

3 - التجريد :-

إخفاء التفاصيل الداخلية للفئة وتقديم واجهة بسيطة للمستخدم.

```
<?php
abstract class Animal {
    abstract public function sound();
}
class Dog extends Animal {
    public function sound() {
        echo "الكلب ينبح.";
    }
}
class Cat extends Animal {
    public function sound() {
        echo "القطة تموء.";
    }
}
$animals = [new Dog(), new Cat()];
foreach ($animals as $animal) {
    $animal->sound();    }      ?>
```

4 - التغليف :-

تجميع البيانات والدوال معًا داخل الكائن.

```
<?php
class BankAccount {
    private $balance = 0;
    public function deposit($amount) {
        if ($amount > 0) {
            $this->balance += $amount;
        }
    }
    public function withdraw($amount) {
        if ($amount > 0 && $this->balance >=
$amount) {
            $this->balance -= $amount;
        }
    }
```

```
    public function getBalance() {
        return $this->balance;
    }
}
$account = new BankAccount();
$account->deposit(1000);
$account->withdraw(200);
echo $account->getBalance(); // 800 :الإخراج
?>
```

المميزات المتعلقة بـ OOP في PHP 8 :-

1 - الخصائص المعرفية (TYPED PROPERTIES) :-

يمكنك الآن تحديد نوع البيانات للخصائص.

```
<?php
class Product {
    public string $name;
    public float $price;
}
?>
```

2 - الدوال المنشئة (Constructor Promotion):-

تحسين الدوال المنشئة لتقليل التكرار عند تعريف الخصائص.

```
<?php
class User {
    public function __construct(
        public string $name,
        public int $age
    ) {}
}
$user = new User("Ali", 25);
echo $user->name; // الإخراج: Ali
?>
```

3 - الدوال الثابتة (Static Methods):-

يمكن استدعاء الدوال بدون إنشاء كائن.

```
<?php
class Math {
    public static function add($a, $b) {
        return $a + $b;
    }
}
echo Math::add(5, 10); //   15 :الإخراج          ?>
```

4 - التطابق (Match Expressions):-

طريقة محسنة للتعامل مع الحالات الشرطية.

```
<?php
class Order {
   public function getStatusMessage(int $status): string {
      return match($status) {
         1 => "تمت الموافقة",
         2 => "قيد المعالجة",
         3 => "تم الإلغاء",
         default => "حالة غير معروفة",
      };
   }
}
$order = new Order();
echo $order->getStatusMessage(1); // الإخراج: تمت الموافقة
?>
```

5 - القيم الافتراضية الثابتة (Static Variables in Inheritance):-

تحسينات في التعامل مع الخصائص الثابتة عند الوراثة.

```
<?php
class ParentClass {
   protected static string $value = "Parent";
   public static function getValue() {
      return static::$value;
   }
}
class ChildClass extends ParentClass {
   protected static string $value = "Child";
}
echo ChildClass::getValue(); // الإخراج: Child
?>
```

التصميم الهيكلي في الـ OOP :-

1 ـ النمط الأحادي (Singleton Pattern):-

يضمن وجود نسخة واحدة فقط من الكائن في النظام.

```
<?php
class Singleton {
    private static $instance;
    private function __construct() {}
    public static function getInstance() {
        if (self::$instance === null) {
            self::$instance = new Singleton();
        }
        return self::$instance;
    }
}
$instance1 = Singleton::getInstance();
$instance2 = Singleton::getInstance();
var_dump($instance1 === $instance2); // الإخراج: true
?>
```

2 ـ نمط المصنع (Factory Pattern):-

يوفر واجهة لإنشاء كائنات دون تحديد نوعها الدقيق.

```
<?php
class ShapeFactory {
    public static function createShape($type) {
        return match($type) {
            'circle' => new Circle(),
            'square' => new Square(),
            default => throw new Exception("نوع غير معروف"),
        };
    }
}
$circle = ShapeFactory::createShape('circle');
$square = ShapeFactory::createShape('square');          ?>
```