|  |
| --- |
| تطوير ويب 2 |
| **البرمجة كائنية التوجه OOP في لغة PHP** |
| د/إبراهيم الشامي |

|  |
| --- |
| **هديل خالد عبدالله الحزورة**  **علوم حاسوب** |

* **المقدمة:**

البرمجة كائنية التوجه (Object-Oriented Programming - OOP) هي نهج برمجي يعتمد على إنشاء برامج تعتمد على الكائنات.

في PHP، تُعتبر OOP طريقة حديثة ومنظمة لبناء التطبيقات حيث توفر تسهيلات لإنشاء الشيفرة بشكل مرن وقابل لإعادة الاستخدام.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* **مفاهيم أساسية في OOP:**

1. **الكلاس (Class)**

الكلاس هو الهيكل الأساسي الذي يُستخدم لإنشاء الكائنات. يحتوي على الخصائص (Properties) التي تمثل البيانات، والدوال (Methods) التي تمثل السلوكيات.

class Car {

public $brand;

public $color;

public function drive() {

return "Driving the $this->color $this->brand.";

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **الكائن (Object)**

الكائن هو نسخة من الكلاس يتم إنشاؤها لاستخدام البيانات والسلوكيات المحددة في الكلاس.

$myCar = new Car();

$myCar->brand = "Toyota";

$myCar->color = "Red";

echo $myCar->drive(); // "Driving the Red Toyota."

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **الخصائص (Properties)**

تمثل المتغيرات داخل الكلاس التي تحتفظ بالبيانات الخاصة بالكائن.

class Person {

public $name;

public $age;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **الدوال (Methods)**

تمثل الوظائف أو السلوكيات التي يمكن أن يؤديها الكائن.

class Calculator {

public function add($a, $b) {

return $a + $b;

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* **مبادئ OOP الأساسية:**

1. **التغليف (Encapsulation)**

يُستخدم لإخفاء تفاصيل الكود الداخلي للكائن وجعل البيانات قابلة للوصول فقط من خلال واجهة محددة. يتم تحقيق ذلك باستخدام مستويات الوصول:

* **Public**: يمكن الوصول من أي مكان.
* **Private**: يمكن الوصول داخل الكلاس فقط.
* **Protected**: يمكن الوصول داخل الكلاس والكلاسات الموروثة فقط.

class BankAccount {

private $balance;

public function setBalance($amount) {

$this->balance = $amount; }

public function getBalance() {

return $this->balance;}}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **الوراثة (Inheritance)**

تسمح بإنشاء كلاس جديد يرث الخصائص والدوال من كلاس آخر.

class Vehicle {

public $speed;

public function move() {

return "Moving at $this->speed km/h.";

}

}

class Car extends Vehicle {

public $brand;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **تعدد الأشكال (Polymorphism)**

يسمح بإعادة تعريف الدوال في الكلاسات الفرعية لتغيير سلوكها.

class Animal {

public function speak() {

return "Some sound";

}

}

class Dog extends Animal {

public function speak() {

return "Bark";

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **التجريد (Abstraction)**

إخفاء تفاصيل التنفيذ وتوفير واجهة للاستخدام فقط. يتم تحقيق ذلك باستخدام الكلاسات والدوال المجردة.

abstract class Shape {

abstract public function area();

}

class Rectangle extends Shape {

public $width;

public $height;

public function area() {

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

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* **ميزات OOP في: PHP**

1. **الكلاسات النهائية (Final Classes)**

تمنع الوراثة من الكلاس.

final class Database {

// لا يمكن إنشاء كلاس فرعي

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **الواجهات (Interfaces)**

تحدد مجموعة من الدوال التي يجب على الكلاس أن ينفذها.

interface Logger {

public function log($message);}

class FileLogger implements Logger {

public function log($message) {

echo "Logging to file: $message";}}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. **الصفات (Traits)**

تُستخدم لمشاركة الشيفرة بين الكلاسات دون الحاجة إلى الوراثة.

trait LoggerTrait {

public function log($message) {

echo "Log: $message";

}

}

class User {

use LoggerTrait;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* **أهمية استخدام OOP في PHP**

1. **تنظيم الشيفرة**: يسهل تقسيم المشروع إلى وحدات صغيرة ومنظمة.
2. **إعادة الاستخدام**: الكلاسات يمكن إعادة استخدامها في مشاريع مختلفة.
3. **التوسع والصيانة**: يسهل إجراء التعديلات وإضافة ميزات جديدة دون التأثير على بقية المشروع.
4. **قابلية الاختبار**: يسهل اختبار الكود بفضل الفصل بين الكلاسات والوظائف.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* **الخاتمة**

البرمجة كائنية التوجه (OOP) في PHP تُعتبر أساسًا لبناء تطبيقات كبيرة ومعقدة بطريقة مرنة وقابلة للتوسعة. استخدام مفاهيم مثل التغليف، الوراثة، التجريد، وتعدد الأشكال يجعل الشيفرة أكثر تنظيماً وسهولة في الصيانة.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_