تكليف :الخاص ب

OOP in PHP

عمل الطالب :

الخليل عبدالفتاح سالم حسن

* **المفاهيم الأساسية في OOP**

### الكلاس (Class)

**. الكلاس هو نموذج أو قالب يُستخدم لإنشاء الكائنات**

**يحتوي على خصائص و طرق .**

**class Car {**

**public $brand;**

**public $model;**

**public function startEngine() {**

**return "Engine started!";**

**}**

**}**

**الكائن (Object)**

**الكائن هو نسخة من الكلاس .**

**$myCar = new Car();**

**$myCar->brand = "Toyota";**

**$myCar->model = "Corolla";**

**echo $myCar->startEngine(); // Output: Engine started!**

**الخصائص (Properties)**

* **هي المتغيرات التي تنتمي إلى الكلاس.**
* **يمكن أن تكون عامة (public)، خاصة (private)، أو محمية (protected).**
* **المبادئ الأساسية في OOP**

**1. التغليف (Encapsulation)**

* **يعني إخفاء التفاصيل الداخلية للكائن وتعريض واجهة بسيطة للتعامل معه.**
* **يتم تحقيق ذلك باستخدام معدلات الوصول (public, private, protected).**

**class BankAccount {**

**private $balance = 0;**

**public function deposit($amount) {**

**if ($amount > 0) {**

**$this->balance += $amount;**

**}**

**}**

**public function getBalance() {**

**return $this->balance;**

**}**

**}**

**$account = new BankAccount();**

**$account->deposit(100);**

**echo $account->getBalance();**

**2. الوراثة (Inheritance)**

* **تسمح بإنشاء كلاس جديد بناءً على كلاس موجود.**
* **يتم استخدام الكلمة المفتاحية extends.**

**class Animal {**

**public function makeSound() {**

**return "Some sound";**

**}**

**}**

**class Dog extends Animal {**

**public function makeSound() {**

**return "Woof!";**

**}**

**}**

**$dog = new Dog();**

**echo $dog->makeSound();**

**3. تعدد الأشكال (Polymorphism)**

* **يعني أن الكائنات يمكن أن تأخذ أشكالًا متعددة.**
* **يتم تحقيق ذلك عن طريق تجاوز الطرق (Method Overriding) أو استخدام الواجهات (Interfaces).**

**interface Shape {**

**public function area();**

**}**

**class Circle implements Shape {**

**private $radius;**

**public function \_\_construct($radius) {**

**$this->radius = $radius;**

**}**

**public function area() {**

**return pi() \* pow($this->radius, 2);**

**}**

**}**

**class Square implements Shape {**

**private $side;**

**public function \_\_construct($side) {**

**$this->side = $side;**

**}**

**public function area() {**

**return pow($this->side, 2);**

**}**

**}**

**$circle = new Circle(5);**

**$square = new Square(4);**

**echo $circle->area();**

**echo $square->area();**

**4. التجريد (Abstraction)**

* **يعني إخفاء التفاصيل المعقدة وتعريض الوظائف الأساسية فقط.**
* **يتم تحقيق ذلك باستخدام الكلاسات المجردة (abstract classes) أو الواجهات (interfaces).**

**abstract class Vehicle {**

**abstract public function startEngine();**

**}**

**class Car extends Vehicle {**

**public function startEngine() {**

**return "Car engine started!";**

**}**

**}**

**$car = new Car();**

**echo $car->startEngine(); // Output: Car engine started!**

* **معدلات الوصول (Access Modifiers)**
* **Public : يمكن الوصول إلى الخاصية أو الطريقة من أي مكان.**
* **Private :  يمكن الوصول إلى الخاصية أو الطريقة فقط من داخل الكلاس نفسه.**
* **Protected :  يمكن الوصول إلى الخاصية أو الطريقة من داخل الكلاس نفسه و الكلاسات التي ترث منه.**
* **المُنشئ (Constructor) والمُدمِّر (Destructor)**
* **يتم استدعاؤه تلقائيًا عند إنشاء كائن جديد.**
* **يُستخدم لتهيئة الخصائص.**

**class Person {**

**private $name;**

**public function \_\_construct($name) {**

**$this->name = $name;**

**}**

**public function getName() {**

**return $this->name;**

**}**

**}**

**$person = new Person("John");**

**echo $person->getName();**

**المُدمِّر (\_\_destruct)**

* **يتم استدعاؤه تلقائيًا عند تدمير الكائن.**
* **يُستخدم لتنظيف الموارد.**

**class Person {**

**public function \_\_destruct() {**

**echo "Object destroyed!";**

**}**

**}**

**$person = new Person();**

* **الواجهات (Interfaces)**
* الواجهة تحدد مجموعة من الطرق التي يجب على الكلاس تنفيذها.
* يتم استخدام الكلمة المفتاحية implements.

**interface Logger {**

**public function log($message);**

**}**

**class FileLogger implements Logger {**

**public function log($message) {**

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

**}**

**}**

**$logger = new FileLogger();**

**$logger->log("Error occurred!");**

* **السمات (Traits)**
* السمات تُستخدم لإعادة استخدام الأكواد في الكلاسات.
* يتم استخدام الكلمة المفتاحية use.

**trait Loggable {**

**public function log($message) {**

**echo "Logging: $message";**

**}**

**}**

**class User {**

**use Loggable;**

**}**

**$user = new User();**

**$user->log("User created!");**

* **الكلاسات المجردة (Abstract Classes)**
* لا يمكن إنشاء كائنات منها مباشرة.
* يجب أن يتم توريثها وتنفيذ الطرق المجردة.

**abstract class Animal {**

**abstract public function makeSound();**

**}**

**class Dog extends Animal {**

**public function makeSound() {**

**return "Woof!";**

**}**

**}**

**$dog = new Dog();**

**echo $dog->makeSound();**

* **الكلاسات النهائية (Final Classes)**
* لا يمكن توريثها.

**class ParentClass {**

**final public function showMessage() {**

**return "This is a final method.";**

**}**

**}**

* **الكلاسات المجهولة (Anonymous Classes)**
* تُستخدم لإنشاء كائنات بدون تعريف كلاس مسبقًا.

**$obj = new class {**

**public function sayHello() {**

**return "Hello!";**

**}**

**};**

**echo $obj->sayHello();**

* **التحميل الزائد (Overloading)**
* يُستخدم للتعامل مع الخصائص والطرق الديناميكية.

**class OverloadExample {**

**private $data = [];**

**public function \_\_set($name, $value) {**

**$this->data[$name] = $value;**

**}**

**public function \_\_get($name) {**

**return $this->data[$name] ?? null;**

**}**

**}**

**$obj = new OverloadExample();**

**$obj->name = "John";**

**echo $obj->name;**

* **التحميل السحري (Magic Methods)**
* هي دوال خاصة يتم استدعاؤها تلقائيًا في ظروف معينة.
* أمثلة: \_\_construct(), \_\_destruct(), \_\_get(), \_\_set(), \_\_toString(), إلخ.

**class MagicMethods {**

**public function \_\_toString() {**

**return "This is a magic method!";**

**}**

**}**

**$obj = new MagicMethods();**

**echo $obj;**

* **التحميل الزائد للطرق (Method Overloading)**
* يتم استخدام \_\_call() و \_\_callStatic() للتعامل مع الطرق الديناميكية.

**class MethodOverloading {**

**public function \_\_call($name, $arguments) {**

**return "Calling method '$name' with arguments: " . implode(', ', $arguments);**

**}**

**}**

**$obj = new MethodOverloading();**

**echo $obj->undefinedMethod("arg1", "arg2");**

* **التحميل الزائد الثابت (Static Overloading)**
* يتم استخدام \_\_callStatic() للتعامل مع الطرق الثابتة الديناميكية.

**class StaticOverloading {**

**public static function \_\_callStatic($name, $arguments) {**

**return "Calling static method '$name' with arguments: " . implode(', ', $arguments);**

**}**

**}**

**echo StaticOverloading::undefinedStaticMethod("arg1", "arg2");**