1. Giới thiệu về OOP trong PHP

- Lập trình hướng đối tượng (OOP - Object-Oriented Programming) trong PHP giúp tổ chức code thành các đối tượng (objects) và lớp (classes) để dễ bảo trì và mở rộng.
- Lợi ích của OOP trong PHP:
  ✔ Dễ bảo trì và mở rộng.
  ✔ Tái sử dụng code hiệu quả.
  ✔ Tăng tính bảo mật thông qua các thuộc tính & phương thức private.
  ✔ Hỗ trợ các nguyên tắc lập trình hướng đối tượng như kế thừa (inheritance), đóng gói (encapsulation), và đa hình (polymorphism).

2. Class và Object trong PHP

- Class (Lớp): Là một bản thiết kế (blueprint) cho một đối tượng.
- Object (Đối tượng): Là một thể hiện (instance) của class.

Ví dụ 1: Định nghĩa class và tạo object

```php
class Car {
    public $brand; // Thuộc tính (property)
    public $color;

    // Hàm khởi tạo
    public function __construct($brand, $color) {
        $this->brand = $brand;
        $this->color = $color;
    }

    // Phương thức hiển thị thông tin xe
    public function displayInfo() {
        return "Xe " . $this->brand . " màu " . $this->color;
    }
}
```

```php
// Tạo object từ class Car
$car1 = new Car("Toyota", "Đỏ");
$car2 = new Car("Honda", "Xanh");

// Gọi phương thức
echo $car1->displayInfo();
echo "<br>";
echo $car2->displayInfo();
```

Output:

```
Xe Toyota màu Đỏ
Xe Honda màu Xanh
```

Giải thích:

class Car định nghĩa một lớp với thuộc tính **$brand** và **$color**.

Hàm __construct() giúp khởi tạo đối tượng ngay khi tạo **instance**.

$this->brand = $brand; gán giá trị cho thuộc tính của đối tượng.

3. Các Phạm vi truy cập (Access Modifiers)

PHP hỗ trợ 3 mức truy cập cho **thuộc tính** và **phương thức**:

Phạm vi	Mô tả
`public`	Truy cập từ mọi nơi.
`protected`	Chỉ có thể truy cập từ trong class và class con.
`private`	Chỉ có thể truy cập bên trong class.

Ví dụ 2: Sử dụng private, protected và public

```php
class Person {
    public $name;      // Có thể truy cập ở bất cứ đâu
    protected $age;    // Chỉ có thể truy cập trong class & class con
    private $password; // Chỉ có thể truy cập trong class

    public function __construct($name, $age, $password) {
        $this->name = $name;
        $this->age = $age;
        $this->password = $password;
    }

    public function getInfo() {
        return "Tên: " . $this->name . ", Tuổi: " . $this->age;
    }
}

$person = new Person("Nam", 25, "123456");
```

```php
// Truy cập thuộc tính public
echo $person->name;  // ✅ Được phép
// echo $person->age; // ❌ Lỗi vì protected
// echo $person->password; // ❌ Lỗi vì private

// Truy cập qua phương thức public
echo "<br>" . $person->getInfo();
```

output:

```
Nam
Tên: Nam, Tuổi: 25
```

Lưu ý:

public cho phép truy cập trực tiếp.
protected và **private** chỉ có thể truy cập qua phương thức trong class.

4. Kế thừa (Inheritance)

- Kế thừa giúp một class mở rộng (extend) từ class khác, tái sử dụng code.
- Dùng từ khóa extends để kế thừa.

Ví dụ 3: Class Employee kế thừa từ Person

```php
class Employee extends Person {
    public $salary;

    public function __construct($name, $age, $password, $salary) {
        parent::__construct($name, $age, $password); // Gọi constructor của class cha
        $this->salary = $salary;
    }

    public function getEmployeeInfo() {
        return $this->getInfo() . ", Lương: " . $this->salary;
    }
}

$employee = new Employee("Hải", 30, "abcdef", 5000);
echo $employee->getEmployeeInfo();
```

Output:

```
Tên: Hải, Tuổi: 30, Lương: 5000
```

Employee kế thừa **Person**, nghĩa là có thể sử dụng các thuộc tính và phương thức của Person.
parent::__construct(...) gọi constructor của class cha.

5. Đa hình (Polymorphism)

Đa hình là việc class con có thể ghi đè (**override**) phương thức của class cha.

Ví dụ 4: Ghi đè phương thức

```php
class Animal {
    public function makeSound() {
        return "Tiếng động chung";
    }
}

class Dog extends Animal {
    public function makeSound() {
        return "Gâu gâu!";
    }
}

class Cat extends Animal {
    public function makeSound() {
        return "Meo meo!";
    }
}
```

```php
$dog = new Dog();
$cat = new Cat();

echo $dog->makeSound();
echo "<br>";
echo $cat->makeSound();
```

Output:

```
Gâu gâu!
Meo meo!
```

6. Interface trong PHP

Interface là một tập hợp các phương thức mà class phải triển khai.
Dùng từ khóa **interface**.

Ví dụ 5: Interface AnimalInterface

```php
interface AnimalInterface {
    public function makeSound();
}

class Dog implements AnimalInterface {
    public function makeSound() {
        return "Gâu gâu!";
    }
}

$dog = new Dog();
echo $dog->makeSound();
```

Output:

```
Gâu gâu!
```

Lưu ý:

Interface chỉ chứa khai báo phương thức, không có phần thân.
Class Dog phải triển khai toàn bộ phương thức của **interface**.

7. Abstract Class

Abstract Class là class chứa ít nhất một phương thức trừu tượng (**abstract method**).
Không thể tạo **instance** từ **abstract class**.

Ví dụ 6: Abstract class Animal

```php
abstract class Animal {
    abstract public function makeSound();
}

class Dog extends Animal {
    public function makeSound() {
        return "Gâu gâu!";
    }
}

$dog = new Dog();
echo $dog->makeSound();
```

Output:

```
Gâu gâu!
```

- Sự khác biệt giữa **Abstract Class** và **Interface**:

Tính năng	Interface	Abstract Class
Chứa phương thức có thân	❌ Không	✅ Có thể
Chứa thuộc tính	❌ Không	✅ Có thể
Đa kế thừa	✅ Có thể	❌ Không

8. Kết luận

Khái niệm	Chức năng
Class & Object	Định nghĩa và tạo đối tượng
Constructor	Khởi tạo giá trị ban đầu
Access Modifier	Quy định phạm vi truy cập
Inheritance	Kế thừa từ class cha
Polymorphism	Ghi đè phương thức
Interface	Định nghĩa các phương thức cần triển khai
Abstract Class	Định nghĩa lớp trừu tượng