Bài 3
Lớp và đối tượng trong PHP

Module: **ADVANCED PROGRAMMING WITH PHP 2.0**

Mục tiêu

- Khai báo được lớp trong PHP
- Khởi tạo được đối tượng trong PHP
- Triển khai được getter/setter
- Sử dụng được access modifier
- Sử dụng được static method

Thảo luận

Lớp

Đối tượng

Khai báo lớp

- **Lớp** là đơn vị thực thi cơ bản trong ngôn ngữ PHP
- Lớp quy định hình thức và các khả năng của các đối tượng
- Khai báo lớp đồng thời cũng là khai báo một kiểu dữ liệu mới để có thể khởi tạo các đối tượng thuộc kiểu dữ liệu đó

Cú pháp khai báo lớp

- Cú pháp:

```
class <class_name> {
    // class body
}
```

Trong đó:

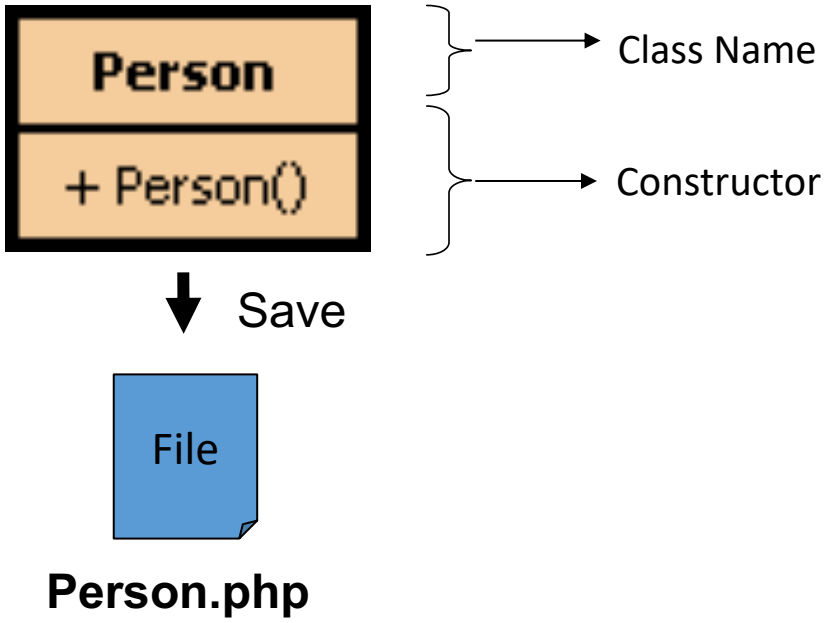
- ***class*** là từ khoá được dùng để khai báo biến
- ***class_name*** là tên của lớp
- ***class body*** là phần thân của lớp: nơi khai báo các thành phần của lớp như các trường (field), các phương thức (method) và các phương thức khởi tạo (constructor)

- Constructor – phương thức khởi tạo: là một phương thức đặc biệt được sử dụng để khởi tạo các đối tượng của một lớp

Đặt tên lớp

- Một số quy ước khi đặt tên lớp:
  - Tên lớp nên là một danh từ
  - Tên lớp nên tuân theo quy tắc Camel
  - Tên lớp nên đơn giản, có nghĩa
  - Tên lớp không thể trùng với các từ khoá trong Java
  - Tên lớp không được bắt đầu bằng chữ số. Có thể bắt đầu bằng ký tự dollar ($) hoặc dấu gạch dưới (_)

Khai báo lớp: Ví dụ




```php
<?php
class Person
{
    public __construct () {
    }
}
?>
```

Khởi tạo đối tượng

- Có thể khởi tạo đối tượng của một lớp sau khi lớp đó được khai báo
- Sử dụng từ khoá ***new*** để khởi tạo đối tượng
- Cú pháp:

```
<$object_name> = new <class_name> ();
```

Trong đó:

- **class_name** là tên của lớp
- ***new*** là từ khoá để khởi tạo đối tượng
- ***object_name*** là tên biến chứa tham chiếu trỏ đến đối tượng

Khởi tạo đối tượng: Ví dụ

- Ví dụ:

```
$personObj = new Person();
```

Trong đó:

- Biểu thức **new Person()** ở phía bên phải cấp phát bộ nhớ tại thời điểm thực thi
- Sau khi vùng nhớ đã được cấp phát thì một tham chiếu đến vùng nhớ đó được trả về và gán cho biến $personObj

Quá trình khởi tạo đối tượng - 1

- Bước 1: Khai báo biến:

```
$personObj;
```

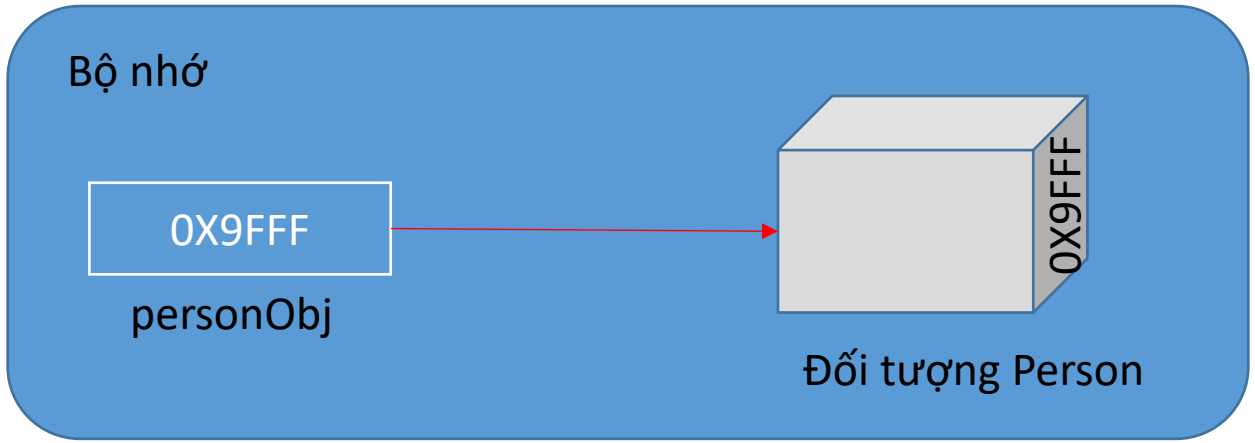
  - Biến personObj được khai báo và không trỏ đến bất kỳ đối tượng nào
  - Biến personObj có giá trị null
  - Nếu sử dụng biến personObj để truy cập các phương thức hoặc thuộc tính của lớp Person tại thời điểm này, sẽ có lỗi xảy ra

Quá trình khởi tạo đối tượng - 2

- Bước 2: Khởi tạo đối tượng:

```
$personObj = new Person();
```

  - Một đối tượng của lớp Person được khởi tạo và lưu vào một vùng nhớ (chẳng hạn có địa chỉ là 0X9FFF)
  - Địa chỉ vùng nhớ được gán cho biến personObj



__construct() vs __destruct()

- Để can thiệp vào quá trình khởi tạo đối tượng, chúng ta sử dụng hàm __construct()
- Để can thiệp vào quá trình huỷ đối tượng, chúng ta sử dụng hàm __destruct()

Thứ tự thực hiện các phương thức của object

1. __constructor()

2. other_method()

3. __destructor()

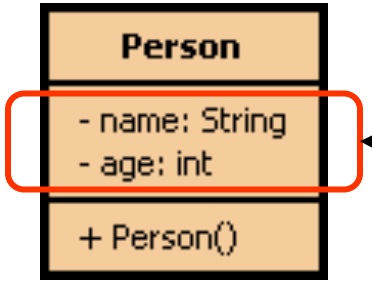
```php
class MyClass
{
    function __construct()
    {
        echo 'Calling constructor<br/>';
    }

    function some_method(){
        echo 'Calling a method<br/>';
    }

    function __destruct()
    {
        echo 'Calling destructor<br/>';
    }
}
```

Khai báo thuộc tính

- Các thuộc tính mô tả các đặc điểm của đối tượng
- Thuộc tính còn được gọi là instance variable (biến của đối tượng)
- Cú pháp:

```
access_modifier dataType $property_name
```

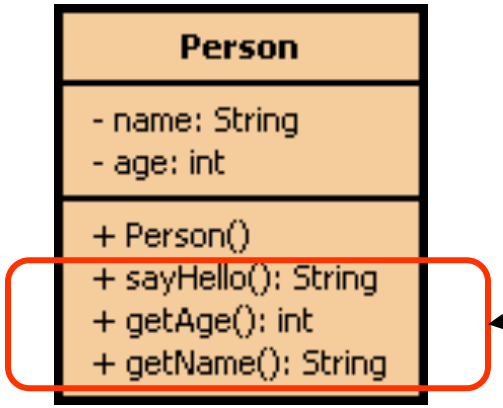
- Ví dụ:




```php
<?php
public class Person
{
    public string $name;
    public int $age;
    public function __construct () {
    }
}
?>
```

Khai báo phương thức

- Phương thức mô tả các hành vi mà đối tượng có thể thực hiện
- Phương thức còn được gọi là instance method (phương thức của đối tượng)
- Ví dụ:




```php
<?php
public class Person
{
    private string $name;
    private int $age;
    public function __construct () {
    }
    public function sayHello(): string
    {
        return “Hello World”;
    }
    public function getAge(): int
    {
        return $this->age;
    }
    public function getName(): string
    {
        return $this->name;
    }
}
?>
```

Hàm tạo (constructor)

- Dùng để **tạo và khởi tạo** các trạng thái ban đầu của đối tượng
- Mặc định sẽ có một ***Hàm tạo không có đối số*** được thêm vào cho lớp.

```
class BaseClass
{
  private dataType $variableValue;

  function __construct($variableValue)
 {
     $this-> propertyName = $variableValue;
  }
}
```

Sử dụng constructor

- Có thể lựa chọn sử dụng các constructor khác nhau bằng cách truyền vào tham số khác nhau
- Ví dụ:

```
public __constructor() {
      $this->name = "No name";
      $this->age = 10;
}
```

→ $personObj = **new** Person();

```
public constructor($s, $n) {
      $this->name = $s;
      $this->age = $n;
}
```

→ $personObj = **new** Person("John", 20);

Truy xuất các thành phần của đối tượng

Truy xuất các thuộc tính

Gọi các phương thức

Truy xuất thuộc tính của đối tượng

- Có thể truy xuất các thành phần của đối tượng thông qua biến trỏ đến đối tượng
- Sử dụng dấu (->) để truy xuất thuộc tính của đối tượng
- Ví dụ:

```
$person1 = new Person();
$person1->name = "Nam";
$name = $person1.name;
```

Lưu ý: Quyền truy xuất đến các thành phần của đối tượng được quy định bởi access modifier (public/private/protected/default), sẽ được đề cập đến sau.

Gọi phương thức

- Sử dụng dấu (->) để gọi phương thức của đối tượng
- Ví dụ:

```
$person1 = new Person();
$person1->sayHello();
```

Access modifier

Access modifier

- Access modifier là các từ khoá được sử dụng để quy định mức độ truy cập đến lớp và các thành phần của lớp.
- Các mức truy cập:
  - *public*: có thể truy cập từ bất cứ đâu.
  - *private*: các phương thức và thuộc tính chỉ được phép truy xuất trong cùng một lớp.
  - *protected*: các phương thức và thuộc tính được phép truy xuất trong cùng một lớp và ở các lớp con (kế thừa).

Access modifier: Ví dụ private

```php
<?php
class Customer {
    private string $name;
    public function setName($name) : void
    {
        $this->name = $name;
    }
    public function getName(): string
   {
        return $this->name;
    }
}

$c = new Customer();
$c->setName("Stuart Broad");
echo $c->name; //error, $name cannot be accessed from outside the
class
//$name can only be accessed from within the class

echo $c->getName(); //this works, as the methods of the class have
access
//to the private data members or methods
```

private

Access modifier: Ví dụ public

```
class Customer {
    public string $name;
    public function setName($name): void
   {
        $this->name = $name;
    }

    public function getName(): string
   {
        return $this->name;
    }
}

$c = new Customer();
$c->setName("Stuart Broad");
echo $c->name;    // this will work as it is public.
$c->name = "New Name" ; // this does not give an error.
```

public

Access modifier: Ví dụ protected

```php
<?php
class Customer {

    protected string $name;

    public function setName($name): void
    {
        $this->name = $name;
    }

    public function getName(): string
    {
        return $this->name;
    }
}
```

```php
class DiscountCustomer extends Customer {

    private int $discount;

    public function setData($name, $discount): void
    {
        $this->name = $name;
        $this->discount = $discount;
    }
}

$dc = new DiscountCustomer();
$dc->setData("Stuart Broad",10);
echo $dc->name; // this does not work as $name is protected
and hence
// only available in Customer and DiscountCustomer class
```

protected

Access modifier - bảng tổng hợp

Access Modifier	Class	Sub-class	World
public	Y	Y	Y
proteced	Y	Y	N
private	Y	N	N

Getter và Setter

Truy cập trực tiếp vào các trường dữ liệu

- Sử dụng từ khoá public khi khai báo thuộc tính sẽ cho phép truy cập trực tiếp vào các thuộc tính đó
- Ví dụ:

  Khai báo lớp Person sau cho phép truy cập trực tiếp vào trường name

```php
class Person{
    public string $name;
}
```

```php
$person = new Person();
$person->name = "John";
```

- Nhược điểm:
  - Không kiểm soát được truy cập vào thuộc tính
  - Gây khó khăn cho việc duy trì, dễ phát sinh bug

Data field encapsulation

- Data field encapsulation (bao gói trường dữ liệu) là hình thức hạn chế quyền truy cập trực tiếp vào các thuộc tính của đối tượng bằng cách sử dụng từ khoá private
- Khai báo các phương thức để kiểm soát việc truy cập vào các thuộc tính của đối tượng
- Các phương thức cho phép thay đổi giá trị của thuộc tính được gọi là setter, các phương thức cho phép lấy về giá trị của thuộc tính được gọi là getter
- Ví dụ getter: getName(), getAge(), getDate(), isAvailable()…
- Ví dụ setter: setName(), setAge(), setAddress()…

Khai báo getter/setter

- Cú pháp khai báo getter:

```
public function getPropertyName()
```

  - Đối với các thuộc tính kiểu *boolean* thì tên getter bắt đầu bằng chữ *is*:

```
public function isPropertyName()
```

- Cú pháp khai báo setter:

```
public function setPropertyName($propertyValue)
```

Getter/setter: Ví dụ

```php
class Person
{
    private  string $name;

    public function setName($name): void
    {
        $this->name = $name;
    }

    public function getName(): string
    {
        return $this->name;
    }
}
```

```php
$person = new Person();
$person->setName("John");
echo "My name is: " + $person->getName();
```

Từ khoá $this

- Từ khoá *$this* được sử dụng để đại diện cho đối tượng hiện tại
- Có thể sử dụng từ khoá **$this** để truy cập đến các thành phần của đối tượng hiện tại
- Ví dụ, sử dụng từ khoá ***$this*** để phân biệt 2 biến có cùng tên:

```
class Person{
    private string $name;

    public function setName($name): void
   {
        $this->name = $name;
    }
}
```

Biến name của lớp Person

Tham số name được truyền vào

Từ khoá static

Static property

Static method

Từ khoá static

- Từ khoá ***static*** được sử dụng để khai báo các thuộc tính và phương thức của lớp (khác với thuộc tính và phương thức của đối tượng)
- Các thành phần static trực thuộc lớp, thay vì trực thuộc đối tượng
- Biến static còn được gọi là biến của lớp (class variable)
- Phương thức static còn được gọi là phương thức của lớp (class method)
- Có thể truy xuất các thành phần static bằng cách sử dụng lớp hoặc đối tượng
- Không cần khởi tạo đối tượng vẫn có thể sử dụng các thành phần static

Static property

- Cú pháp khai báo *static property*:

```
modifier static $ariable_name;
```

- Ví dụ:

  Khai báo biến static:

```
class Application{
    public static $language = "english";
}
```

  Truy xuất biến static:

```
echo "Current language: " + Application::$language;
```

Static method

- Cú pháp khai báo static method:

```
modifier static function method_name(){
   //body
}
```

- Ví dụ:
  - Khai báo phương thức static

```
class Application{
   public static function getVersion(){
      return "1.0";
   }
}
```

  - Gọi phương thức static

```
echo "Current version: " + Application::getVersion();
```

Tóm tắt bài học

- Lập trình hướng đối tượng (OOP) là mô hình lập trình phổ biến hiện nay
- OOP mô phỏng các đối tượng trong thế giới thực vào trong thế giới lậptrình
- Từ khoá class được sử dụng để khai báo lớp
- Từ khoá new được sử dụng để khởi tạo đối tượng
- Thuộc tính mô tả các đặc điểm của đối tượng
- Phương thức mô tả các hành vi của đối tượng
- Phương thức khởi tạo (constructor) là phương thức giúp khởi tạo các đối tượng
- Nếu không khai báo constructor thì mặc định các lớp đều có một constructor không tham số
- Có thể mô tả lớp bằng các ký hiệu UML có thể truy xuất các thành phần của lớp thông qua dấu chấm (.)

Hướng dẫn

- Hướng dẫn làm bài thực hành và bài tập
- Chuẩn bị bài tiếp: ***Access Modifier và Từ khóa static***