

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG (OOP) - TRONG PHP

NỘI DUNG BÀI HỌC

- Giới thiệu lập trình hướng đối tượng
- Class và Object
- Thuộc tính và phương thức
- Hàm khởi tạo

MỤC TIÊU BÀI HỌC

- Các khái niệm hướng đối tượng
- Tạo các lớp (class), thuộc tính (attribute), phương thức (method)
- Sử dụng các thuộc tính
- Gọi các phương thức
- Thừa kế
- Thiết kế lớp

Giới thiệu lập trình hướng đối tượng

- **Lập trình hướng đối tượng** (gọi tắt là **OOP** - *object-oriented programming*) là một kỹ thuật lập trình hỗ trợ công nghệ đối tượng.
- Trước kia thì chúng ta lập trình theo hướng thủ tục, hướng modun thì giờ chuyển sang đối tượng để xử lý
- Ở các bài học trước thì mình sẽ chia thành các hàm để xử lý, thì giờ đây khi sử dụng hướng đối tượng thì chúng ta sẽ chia ra thành các đối tượng để xử lý.

Ví dụ:

- **Đây là 2 ví dụ:**
- ✓ Cách 1 thì viết theo hướng thủ tục, xd hàm lấy ra tên và sdt.
- ✓ Cách 2 viết theo hướng đối tượng, cũng lấy ra được tên và sdt nhưng cái chúng ta nhìn được đó là yêu tố bảo mật cao hơn...

```
<?php
// Lập trình hướng thủ tục, tạo ra các hàm
function getInfo_Person(){
    $name = "Nguyen Van A";
    $phone = '989888999';
    return "Name: ".$name." " . "phone: ".$phone;
}

// Lập trình hướng đối tượng
class Info_Person
{
    public $name = "Trinh Khac Tung";
    private $phone = "373263978";

    public function getInfo(){
        return $this->name." ", "$this->phone;
    }
}

?>
```

Ưu điểm của lập trình hướng đối tượng

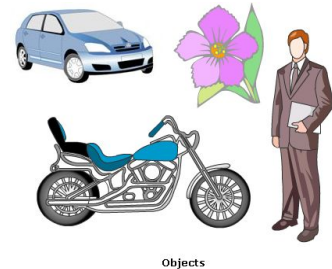
- Dễ dàng quản lý code khi có sự thay đổi về chương trình
- Dễ mở rộng, phát triển khi dự án (project) có sự thay đổi
- Tiết kiệm được tài nguyên đáng kể cho hệ thống.
- Có tính bảo mật cao.
- Có tính tái sử dụng cao.

// OOP được ra đời sau vì vậy nó sẽ khắc phục được các điểm yếu (nhược điểm) của lập trình trước đó

Class và Object

- Trong OOP, có khái niệm về "**class**", được sử dụng để mô hình hóa sang một template dữ liệu (**properties** - các thuộc tính) và chức năng (**methods** - các phương thức).

// Một đối tượng thì bao giờ cũng có: Thuộc tính & phương thức (hành động, method).



Class và Object

- Một "**object**" là giá trị của một class và bạn có thể tạo nhiều giá trị của cùng một class. Nói cách khác thì trong một class có nhiều Object.
- Ví dụ: Trong lớp học có nhiều học viên (person). Vì thế ta sẽ có một class đơn lẻ Person, nhưng nhiều object person có thể là các giá trị của class này

Clas, Lớp

- Cách khai báo một đối tượng (ví dụ Person), đại diện cho đối tượng con người.
- *Cú pháp:*

```
<?php  
  
    class Person  
    {  
        # code...  
    }  
  
?>
```

Thuộc tính

- Thuộc tính (properties) trong class có tác dụng như các biến và hằng trong phương pháp lập trình hướng thủ tục.
- Ví dụ như lớp con người sẽ có các thuộc tính như: tên, mắt, mũi, chân, tay, chiều cao, cân nặng... Và để khai báo thuộc tính trong class chúng ta sử dụng cú pháp:

```
<?php

// Lập trình hướng đối tượng
class Person{

    $name = "Trình Khắc Tung"; // Thuộc tính động
    // hoặc $name = "Trình Khắc Tung", hoặc public $name = "Tung"

    const phone = "373263978"; // Thuộc tính cố định (hằng)

}
```

Thuộc tính

- Để khai báo thuộc tính động (biến) thì chúng ta sử dụng từ khóa var, và chúng ta cũng có thể thiết lập giá trị luôn cho biến bằng phép gán. ví dụ: \$name= 'Trinh Khac Tung'.
- hoặc *[public, private, protected]* \$name = "Trinh Khac Tung";

```
<?php

// Lập trình hướng đối tượng
class Person{

    $name = "Trinh Khac Tung"; // Thuộc tính động
    // hoặc $name = "Trinh Khac Tung", hoặc public $name = "Tung"

    const phone = "373263978"; // Thuộc tính cố định (hằng)

}
```

Phương thức

- ✓ Phương thức trong trong class là các hành động, hành vi của class đó.
- ✓ Về cơ bản nó khá giống với hàm ở trong phương pháp lập trình hướng thủ tục.
Cú pháp khai báo như sau:

```
<?php

// Lập trình hướng đối tượng
class Person{

    $name = "Trinh Khắc Tung"; // Thuộc tính động
    $phone = "373263978"; // Thuộc tính cố định (hằng)

    function work(){
        // công việc của người đó
    }

    function go(){
        // phương tiện đi lại
    }

}

?>
```

Phương thức

✓ Phương thức (hành động, hành vi) trong class thì bản chất đó là: function (hàm), mà đã làm hàm thì:

1. *Hàm không có tham số*
2. *Hàm có tham số truyền vào*
3. *Hàm trả về có tham số*
4. *Hàm trả về không có tham số...*

```
<?php

// Lập trình hướng đối tượng
class Person{

    $name = "Trinh Khac Tung"; // Thuộc tính động
    $phone = "373263978"; // Thuộc tính cố định (hằng)

    function work(){
        echo "Công việc làm: IT";
    }

    function go($car){
        return "Xe người này đi là: ".$car;
    }

}
```

?>

Khởi tạo lớp

- ✓ Khai báo đối tượng: có thuộc tính, có phương thức.
- ✓ Lấy giá trị trong đối tượng, chúng ta cần khởi tạo lớp với từ khóa “new”.

Trong đó: **className** là tên của

class các bạn cần khởi tạo

(khuyến khích dùng cách 2).

- Chúng ta cũng có thể gán nó vào một biến với kiểu dữ liệu là object bằng phép gán

```
<?php
```

```
new className; // cách 1  
//hoặc  
new ClassName(); // cách 2  
  
$info = new ClassName();
```

Truy xuất thuộc tính, lấy giá trị của class

- ✓ Để truy xuất thuộc tính của một class chúng ta sẽ chia làm 2 dạng là truy xuất trong class và truy xuất ngoài class.

1. Truy xuất trong class

Để truy xuất các **thuộc tính động** trong class thì chúng ta dùng từ khóa **this** với cú pháp: **`$this->propertyName`**

```
<?php

// Lập trình hướng đối tượng
class Person{

    var $name = "Trinh Khắc Tung"; // Thuộc tính động
    const phone = "373263978"; // Thuộc tính cố định (hằng)

    function getName(){
        echo "Họ tên của bạn là: ".$this->name;
    }

    function Go($car){
        return "A/c: ".$this->name." đi xe ".$car;
    }

}
```

Truy xuất thuộc tính, lấy giá trị của class

1. Truy xuất trong class

Để truy xuất các **thuộc tính cố định**

trong class thì chúng ta dùng từ khóa

với cú pháp: **className::propertyName**

Hoặc **self::propertyName**

<?php

```
// Lập trình hướng đối tượng
class Person{
    var $name = "Trình Khắc Tung"; // Thuộc tính động
    const phone = "373263978"; // Thuộc tính cố định (hằng)

    function getPhone(){
        echo "Số điện thoại của bạn là: ".self::phone;
        // hoặc echo "Số điện thoại của bạn là: ".Person::phone;
    }
}
```


Truy xuất thuộc tính, lấy giá trị của class

2. Truy xuất bên ngoài class.

▣ Để truy xuất các thuộc tính động

ngoài class thì chúng ta cần:

- Khởi tạo đối tượng, truy xuất với cú pháp:

`$newClass = new className();`

`$newClass->propertyName;`

```
<?php

// Lập trình hướng đối tượng
class Person{
    var $name = "Trình Khắc Tung"; // Thuộc tính động
    const phone = "373263978"; // Thuộc tính cố định (hằng)

    function getPhone(){
        echo "Số điện thoại của bạn là: ".self::phone;
        // hoặc echo "Số điện thoại của bạn là: ".Person::phone;
    }
}

$info = new Person();
echo $info->name;
```

Truy xuất thuộc tính, lấy giá trị của class

2. Truy xuất bên ngoài class.

▣ Để truy xuất các **thuộc tính cố định**

ngoài class thì chúng ta cần:

- Khởi tạo đối tượng, truy xuất với cú pháp:

`$newClass = new className();`

`$newClass::propertyName;`

hoặc `className ::propertyName;`

```
<?php
```

```
// Lập trình hướng đối tượng
class Person{
    var $name = "Trinh Khắc Tung"; // Thuộc tính động
    const phone = "373263978"; // Thuộc tính cố định (hằng)

    function getPhone(){
        echo "Số điện thoại của bạn là: ".self::phone;
        // hoặc echo "Số điện thoại của bạn là: ".Person::phone;
    }
}

$info = new Person();
echo $info::phone;

// hoặc echo Person::phone
```

Truy xuất phương thức ngoài class

- Khởi tạo đối tượng, truy xuất với cú pháp:

```
$newClass = new className();
```

```
$newClass->methodName();
```

```
<?php

// Lập trình hướng đối tượng
class Person{
    var $name = "Trình Khắc Tung"; // Thuộc tính động
    const phone = "373263978"; // Thuộc tính cố định (hằng)

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

    function getPhone(){
        echo $this->getName()." có số điện thoại của bạn là: ".self::
            phone;
        // hoặc echo "Số điện thoại của bạn là: ".Person::phone;
    }
}

$info = new Person();
echo $info->getPhone();
```

Ví dụ

- ✓ - Khai báo đối tượng: Động Vật, bao gồm các thuộc tính:
 - Tên động vật
 - Chiều cao
 - Cân nặng
 - Màu lông
- Phương thức (hành động): noiSong()
- XD phương thức hienThi(), show toàn bộ các thông tin về một con vật nào đó như: Chó, Hươu, Cá...

Phạm vi truy cập

- ✓ Trong lập trình hướng đối tượng các thuộc tính và phương thức được ràng buộc về mức độ truy cập, giúp cho dữ liệu được bảo mật hơn.
- ✓ Cụ thể thể là ba phạm vi: **private**, **protected**, **public**

Phạm vi: public của thuộc tính và phương thức

- ✓ Public: Là công khai, là ai cũng có thể nhìn thấy được
- ✓ Thì trong hướng đối tượng public có nghĩa là chúng ta hoàn toàn có thể truy cập được vào thuộc tính hay phương thức.

```
<?php

// Lập trình hướng đối tượng
class Person{

    public $phone = "373 263 978";

    public function getAddress(){
        echo "Tôi đang sống ở Hà Nội, số điện thoại liên lạc của tôi
        là ".$this->phone;
    }

}

$info = new Person();
echo $info->getAddress();
```

Phạm vi: private, không công khai, chỉ một mình

- **Private** là giới hạn hẹp nhất của thuộc tính và phương thức trong hướng đối tượng.
- Khi các thuộc tính và phương thức khai báo với **private**, thì các thuộc tính phương thức đó chỉ có thể sử dụng được trong class đó, bên ngoài class không thể nào có thể sử dụng được.

Phạm vi: private, không công khai, chỉ một mình!

```
<?php

// Lập trình hướng đối tượng
class Person{

    private $phone = "373 263 978";

    private function getAddres(){
        echo "Tôi đang sống ở Hà Nội, số điện thoại liên lạc của tôi  
là ".$this->phone;
    }

}

$info = new Person();
echo $info->getAddres();
echo $info->phone;
```


Để lấy được số điện thoại, thì chỉ có cách đổi phương thức từ private sang public

```
<?php

// Lập trình hướng đối tượng
class Person{

    private $phone = "373 263 978";

    public function getAddress(){
        echo "Tôi đang sống ở Hà Nội, số điện thoại liên lạc của tôi
            là ".$this->phone;
    }

}

$info = new Person();
echo $info->getAddress();|
```

Cũng có thể set cho \$phone một giá trị nào đó và get giá trị đó ra nếu đang là private

```
<?php

// Lập trình hướng đối tượng
class Person{

    private $phone;

    public function setPhone($phone){
        $this->phone = $phone;
    }

    public function getPhone(){
        return $this->phone;
    }

}

$info = new Person();
$info->setPhone("373263978");
echo $info->getPhone();
```

Ví dụ:

- Để học được môn “Lập trình PHP” yêu cầu người học phải có 3 thông tin sau: Họ tên, Số điện thoại, tuổi (tuổi thì phải lớn hơn 18, nhỏ hơn thì không cho học)

Kế thừa extends.

- Một class kế thừa từ class cha sẽ có được đầy đủ các thuộc tính và phương thức của class cha.
- > **Lưu ý:** Chỉ là có được các thuộc tính và phương thức, còn việc sử dụng được hay không là phụ thuộc vào action của lớp cha.
- Trong PHP để khai báo kế thừa từ một lớp cha chúng ta sử dụng từ khóa extends theo cú pháp:

```
class Info extends Person
{
    // code
}
```

Hàm khởi tạo, construct

- Đây là phương thức (hành động, hàm) có tên trùng với Class hoặc tên hàm là `__construct`
- Hàm này sẽ tự động chạy khi được khởi tạo đối tượng

```
class Person
{
    public $name = 'TungTK';
    public $age = '25';

    function __construct(){
        echo $this->name." ".$this->age;
    }
}
```



Từ khóa **parent** trong class con

- Từ khoá `parent` sẽ giúp bạn gọi đến hàm của class cha, tức class mà class hiện tại của bạn kế thừa (trực tiếp hay gián tiếp).
- Hàm `parent::__construct()` sẽ giúp bạn gọi đến hàm `__construct()` của class cha hoặc `parent::nameFunction()`;

Từ khóa **parent** trong class con

```
class Person
{
    public $name = 'TungTK';
    public $age = '25';

    public function getInfo(){
        echo $this->name." ".$this->age;
    }
}

class Info extends Person
{
    function __construct(){
        $per = new Person();
        $per->getInfo();
    }
}

$info = new Info();
```

Không sử dụng parent::

```
class Person
{
    public $name = 'TungTK';
    public $age = '25';

    public function getInfo(){
        echo $this->name." ".$this->age;
    }
}

class Info extends Person
{
    function __construct(){
        parent::getInfo();
    }
}

$info = new Info();
```

Phạm vi: protected, kế thừa trong class

Khác với **private** một chút thì các phương thức và thuộc tính khi khai là **protected** thì: Ngoài được sử dụng **trong class** đó ra thì **class con kế thừa từ nó** cũng có thể sử dụng được (*nhưng bên ngoài class không có thể sử dụng được*)

Phạm vi: protected, kế thừa trong class

```
<?php
// Lập trình hướng đối tượng
class Person{
    public $name;
    public $phone;
    protected $age;
}
class Pr_Nam extends Person{
    private $marri = "Đã kết hôn!";

    public function setPer($name, $phone, $age){
        $this->name = $name;
        $this->phone = $phone;
        $this->age = $age;
    }

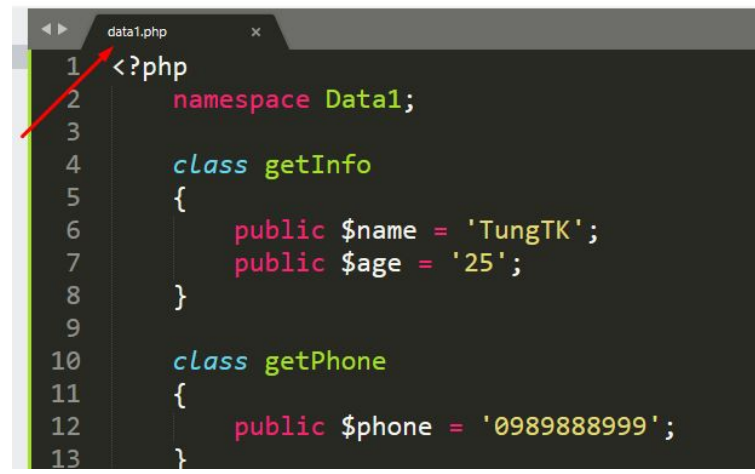
    public function getPer(){
        return $this->name.$this->phone.$this->age;
    }
}
```

Namespace trong php

- Namespace giúp tạo ra một không gian tên cho hàm và lớp trong PHP.
- Nó giải quyết các vấn đề như trong một project sẽ có lúc bạn tổ chức code bị trùng tên class và function



```
data.php
1 <?php
2 class getInfo
3 {
4     public $name = 'Thomas Muller';
5     public $age = '25';
6 }
7
8 class getPhone
9 {
10     public $phone = '3589598999';
11 }
12
```




```
data1.php
1 <?php
2 namespace Data1;
3
4 class getInfo
5 {
6     public $name = 'TungTK';
7     public $age = '25';
8 }
9
10 class getPhone
11 {
12     public $phone = '0989888999';
13 }
```

Khai báo namespace

- Khi khai báo **namespace** thì chúng ta phải đặt nó ở **phía trên cùng** của file

```
namespace Data;  
  
class getInfo  
{  
    public $name = 'Thomas Muller';  
    public $age = '25';  
}
```



Khai báo namespace

- Đặt tên **namespace** theo các cấp.

```
namespace App\Data; // trong đó app là thư mục, Data là tên namespace

class getInfo
{
    public $name = 'Thomas Muller';
    public $age = '25';
}
```



Gọi namespace

- Sử dụng từ khóa **new** và gọi tới tên **namespace**.

`new tenNamespace\tenClass();` // Lưu ý sử dụng **new** luôn thì class phải có **()**;

```
<?php
    include_once 'data-info.php';

    $info = new DataInfo\getInfo();
    echo $info->name;
```

Gọi (nạp) namespace bằng **use**

- Sử dụng từ khóa **use** và gọi tới tên **namespace**;
`new tenNamespace\tenClass; // lưu ý sử dụng use thì class không có ();`

```
<?php
    include_once 'data-info.php';

    use DataInfo\getInfo;
    $info = new getInfo();
    echo $info->name;
```

Gọi (nạp) namespace bằng **use** mà trong file có namespace

- Cách gọi cũng phải theo thứ tự, Ưu tiên đặt namespace lên đầu file và sau đó tới việc gọi file để use namespace

```
<?php
```

```
namespace App\Data; // trong đó app là thư mục, Data là tên namespace
```

```
class getInfoOne
```

```
{
```

```
    public $name = 'Thomas Muller';
```

```
    public $age = '25';
```

```
}
```

```
class getInfoTwo
```

```
{
```

```
    public $name = 'Thomas Muller 22222222';
```

```
    public $age = '25';
```

```
}
```

```
<?php
```

```
namespace Home;
```

```
include_once 'app/data.php';
```

```
use App\Data\getInfoOne; // class 1
```

```
use App\Data\getInfoTwo; // class 2
```

```
$info = new getInfoTwo();
```

```
echo $info->name;
```

Định danh namespace

`use tenNamespace as tenNamespaceMoi;`

```
<?php
    namespace Home;

    include_once 'app/data.php';
    use App\Data as data; // class 1

    $info = new data\getInfoTwo();
    echo $info->name;
```


BÀI TẬP

Tính chu vi và diện tích hình vuông,
hình chữ nhật bằng lập trình
hướng đối tượng.