## 1, Static.

### Static là gì?

-Static trong lập trình hướng đối tượng là một thành phần tĩnh (có thể là thuộc tính hoặc phương thức) mà nó hoạt động như một biến toàn cục, dù cho nó có được xử lý ở trong bất kỳ một file nào đi nữa (trong cùng một chương trình) thì nó đều lưu lại giá trị cuối cùng mà nó được thực hiện vào trong lớp.

### Khai báo static

-Để khai báo một thuộc tính hay một phương thức là static thì chúng ta chỉ việc thêm từ khóa static sau [vibsility](http://toidicode.com/cac-gioi-han-quyen-cua-thuoc-tinh-phuong-thuoc-trong-huong-doi-tuong-100.html).

**Cú Pháp:**

class ClassName

{

//khai báo thuộc tính tĩnh

visibility

static $propertyName;

//Khai báo phương thức tĩnh

visibility static function methodName()

{

//

}

}

**VD**: Mình sẽ khai báo lớp **ConNguoi** có thuộc tính tĩnh là **name** và phương thức tĩnh là **getName**,**setName**.

class ConNguoi

{

private static $name;

public static function setName($name)

{

//code

}

public static function getName()

{

//code

}

}

### Ví dụ chứng minh

-Để chứng minh những điều tôi nói phía trên là đúng thì tôi sẽ cho các bạn xem  2 ví dụ sau:

+Khi không khai báo static.

class ConNguoi

{

private $name = 'amonymouse';

public function setName($name)

{

$this->name = $name;

}

public function getName()

{

return $this->name;

}

}

//khởi tạo đối tượng con người

$chuBlog = new ConNguoi();

//set name cho đối tượng

$chuBlog->setName('Vũ Thanh Tài');

//in ra name của đối tượng

echo $chuBlog->getName();

//kết quả: Vũ Thanh Tài

//khởi tạo đối tượng Con người

$nguoixem = new ConNguoi();

//hiển thị ra tên người xem

echo $nguoixem->getName();

//Kết quả: amonymouse

+Khi khai báo thuộc tính là static.

class ConNguoi

{

private static $name = 'amonymouse';

public function setName($name)

{

self::$name = $name;

}

public function getName()

{

return self::$name;

}

}

//khởi tạo đối tượng con người

$chuBlog = new ConNguoi();

//set name cho đối tượng

$chuBlog->setName('Vũ Thanh Tài');

//in ra name của đối tượng

echo $chuBlog->getName();

//kết quả: Vũ Thanh Tài

//khởi tạo đối tượng Con người

$nguoixem = new ConNguoi();

//hiển thị ra tên người xem

echo $nguoixem->getName();

//Kết quả: Vũ Thanh Tài

### Gọi thành phần tĩnh

Khi bạn khai báo một thuộc tính hay một phương thức ở dạng **static** thì bạn sẽ không thể gọi bằng cách thông thường là dùng từ khóa this được nữa mà sẽ có các cách gọi khác như sau:

**Gọi phương thức và thuộc tính tĩnh trong class**

-Để gọi phương thức và thuộc tính tĩnh trong class thì chúng ta có thể sử dụng cú pháp selft::ten hoặc ClassName::ten hoặc static::ten.

**VD**:

class ConNguoi

{

private static $name = 'Vũ Thanh Tài';

public static function getName()

{

//gọi thuộc tính tĩnh

return self::$name;

//hoặc

return ConNguoi::$name;

}

public function showAll()

{

//gọi phương thức tĩnh

return self::getName();

//hoặc

return ConNguoi::getName();

}

}

**Gọi phương thức và thuộc tính tĩnh ngoài class**

-Để **gọi phương thức tĩnh** ở bên ngoài class thì chúng ta có thể gọi bằng cách [truyền thống](http://toidicode.com/lop-thuoc-tinh-phuong-thuc-trong-php-98.html#load-h3-3) hoặc gọi theo cú pháp ClassName::tenPhuongThuc() , còn đối với **thuộc tính tĩnh** thì chúng ta không thể sử dụng cách truyền thống để gọi được nữa mà phải sử dụng cú pháp ClassName::$tenthuoctinh.

**VD**:

class ConNguoi

{

public static $name = 'Vũ Thanh Tài';

public static function getName()

{

//gọi thuộc tính tĩnh

return self::$name;

//hoặc

return ConNguoi::$name;

}

public static function showAll()

{

//gọi phương thức tĩnh

return self::getName();

//hoặc

return ConNguoi::getName();

}

}

//gọi thuộc tính tĩnh

ConNguoi::$name;

//gọi phương thức tĩnh

ConNguoi::showAll();

//hoặc

$connguoi = new ConNguoi();

$connguoi->showAll();

**Kế thừa phương thức tĩnh từ lớp cha**

-Để kế thừa phương thức tĩnh từ lớp cha thì chúng ta vẫn sử dụng như cách truyền thống.

**VD**:

class ConNguoi

{

private static $name = 'Vũ Thanh Tài';

public static function getName()

{

return ConNguoi::$name;

}

}

class NguoiLon extends ConNguoi

{

public function getName()

{

parent::getName();

}

}

### Nhược điểm của static

-Như ở trên mình có nói 'static nó hoạt động như một biến toàn cục' và cũng vì điều này mà khi sử dụng static trong chương trình thì nó sẽ chiếm nhiều tài nguyên hơn các thành phần thường.

## 2, Final.

-**Final** dịch ra tiếng việt có nghĩa là sau cùng và trong lập trình hướng đối tượng thì **final** được ứng dụng vào class và phương thức.

### Final class

-Khi một class được khai báo là **final** thì không lớp nào có thể kế thừa nó và nó chỉ có thể khởi tạo được thôi.

**Cú pháp**: Khai báo **final** class.

final class ClassName

{

//

}

**VD:**khai báo **final** class ConNguoi.

final class ConNguoi

{

//

}

**VD**: **final** class không thể kế thừa.

final class ConNguoi

{

//

}

//Sai vì không thể kế thừa final class

class NguoiLon extends ConNguoi

{

}

//Fatal error: Class NguoiLon may not inherit from final class (ConNguoi)

### Final Phương thức

-Khi chúng ta khai báo một phương thức là **final** thì không có một phương thức nào có thể override(ghi đè lại được).

**Cú pháp**: khai báo một phương thức **final**.

class ClassName

{

final public function methodName()

{

//

}

}

**VD**: khai báo lớp con người có phương thức getSoChan() là **final**.

class ConNguoi

{

private $soChan = 2;

final public function getSoChan()

{

return $this->soChan;

}

}

**VD**: không thể override lại phương thức **final**.

class ConNguoi

{

private $soChan = 2;

final public function getSoChan()

{

return $this->soChan;

}

}

//Sai vì không thể override lại final phương thức

class NguoiLon extends ConNguoi

{

public function getSoChan()

{

}

}

//Fatal error: Cannot override final method ConNguoi::getSoChan()