# Thuật ngữ cơ bản

## Class – Lớp

Định nghĩa: [Class là một tập hợp các đối tượng và mô tả của các đối tượng có chung các thuộc tính và hành động. Nó chứa các đặc tính của đối tượng như các thuộc tính, các hành động hoặc các hành vi](https://www.google.com/search?q=class+l%C3%A0+m%E1%BB%99t+t%E1%BA%ADp+h%E1%BB%A3p+c%C3%A1c+%C4%91%E1%BB%91i+t%C6%B0%E1%BB%A3ng(objects)+v%C3%A0+m%C3%B4+t%E1%BA%A3+c%E1%BB%A7a+c%C3%A1c+%C4%91%E1%BB%91i+t%C6%B0%E1%BB%A3ng+c%C3%B3+chung+c%C3%A1c+thu%E1%BB%99c+t%C3%ADnh+v%C3%A0+h%C3%A0nh+%C4%91%E1%BB%99ng.+N%C3%B3+ch%E1%BB%A9a+c%C3%A1c+%C4%91%E1%BA%B7c+t%C3%ADnh+c%E1%BB%A7a+%C4%91%E1%BB%91i+t%C6%B0%E1%BB%A3ng+nh%C6%B0+c%C3%A1c+thu%E1%BB%99c+t%C3%ADnh(attributes)%2C+c%C3%A1c+h%C3%A0nh+%C4%91%E1%BB%99ng+ho%E1%BA%B7c+c%C3%A1c+h%C3%A0nh+vi+(behaviors)&oq=class+l%C3%A0+m%E1%BB%99t+t%E1%BA%ADp+h%E1%BB%A3p+c%C3%A1c+%C4%91%E1%BB%91i+t%C6%B0%E1%BB%A3ng(objects)+v%C3%A0+m%C3%B4+t%E1%BA%A3+c%E1%BB%A7a+c%C3%A1c+%C4%91%E1%BB%91i+t%C6%B0%E1%BB%A3ng+c%C3%B3+chung+c%C3%A1c+thu%E1%BB%99c+t%C3%ADnh+v%C3%A0+h%C3%A0nh+%C4%91%E1%BB%99ng.+N%C3%B3+ch%E1%BB%A9a+c%C3%A1c+%C4%91%E1%BA%B7c+t%C3%ADnh+c%E1%BB%A7a+%C4%91%E1%BB%91i+t%C6%B0%E1%BB%A3ng+nh%C6%B0+c%C3%A1c+thu%E1%BB%99c+t%C3%ADnh(attributes)%2C+c%C3%A1c+h%C3%A0nh+%C4%91%E1%BB%99ng+ho%E1%BA%B7c+c%C3%A1c+h%C3%A0nh+vi+(behaviors)&aqs=chrome..69i57.636j0j7&sourceid=chrome&ie=UTF-8).

class Animal

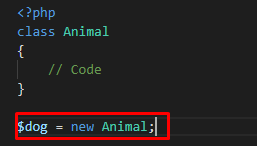
{

    // Code

}

## Object – Đối tượng

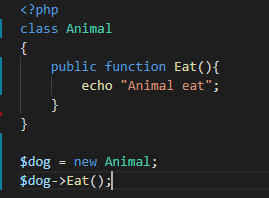
Định nghĩa: Đối tượng trong một ngôn ngữ OOP là các kết hợp giữa mã và dữ liệu mà chúng được nhìn nhận như là một đơn vị duy nhất. Mỗi đối tượng có một tên riêng biệt và tất cả các tham chiếu đến đối tượng đó được tiến hành qua tên của nó.



## Method – Phương thức

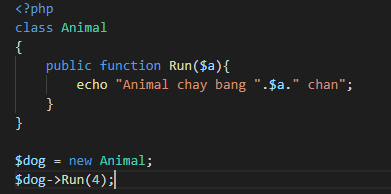
Định nghĩa: Phương thức giống với định nghĩa hàm trong lập trình cấu trúc. Phương thức là các hàm thể hiện hành vi của một đối tượng thuộc lớp. Có các loại phương thức của có đối, phương thức không đối, phương thức trả về, phương thức khởi tạo và phương thức hủy.

* + 1. Phương thức không đối

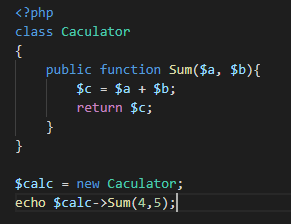




* + 1. Phương thức có đối



* + 1. Phương thức trả về





## Attribute – Thuộc tính

Thuộc tính của một lớp bao gồm các biến, các hằng, hay tham số nội tại của lớp đó. Ở đây, vai trò quan trọng nhất của các thuộc tính là các biến vì chúng sẽ có thể bị thay đổi trong suốt quá trình hoạt động của một đối tượng. Các thuộc tính có thể được xác định kiểu và kiểu của chúng có thể là các kiểu dữ liệu cổ điển hay đó là một lớp đã định nghĩa từ trước.

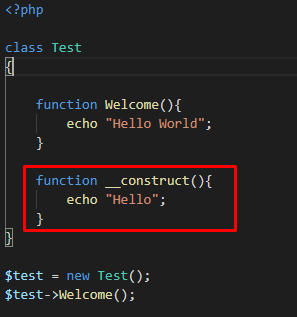
* + 1. Biến
    2. Hằng
    3. Mảng
    4. Object

## Constructor – Destructor

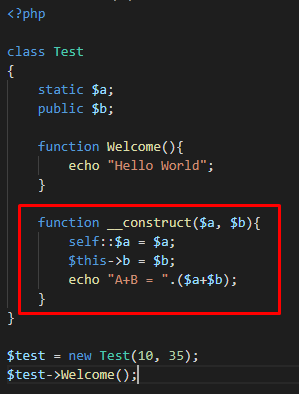
### Construct

* Có hai loại là construct có đối và construct không đối. Và hai phương thức construct này không thể được khai báo cùng lúc.
* Hàm construct là hàm thể hiện cho lớp đó. Và luôn được khai báo mặc định khi ta gọi đối tượng của lớp đó.

VD: Hàm construct không đối số

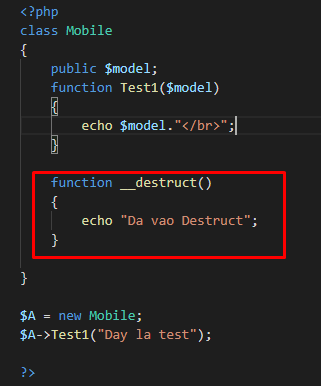


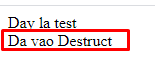
Hàm construct có đối số



### Destruct

* Hàm hủy, hàm này luôn dùng cho đóng collection của database

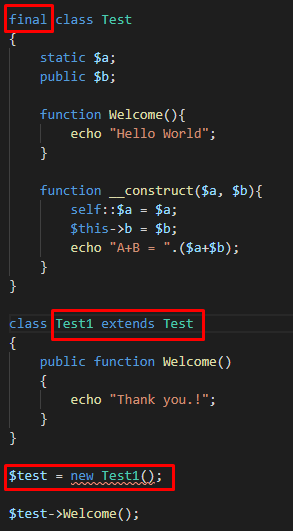




## Final – Static

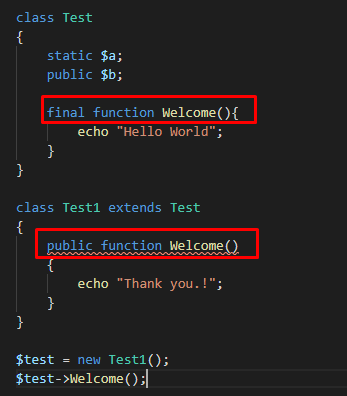
### Final

* Khi khai báo final class thì không lớp nào có thể kế thừa nó mà nó chỉ có thể khởi tạo được thôi





* Tương tự như class khi ta khai báo final cho phương thức thì các phương thức được kế thừa từ lớp con không thể ghi đè lên phương thức đó được.

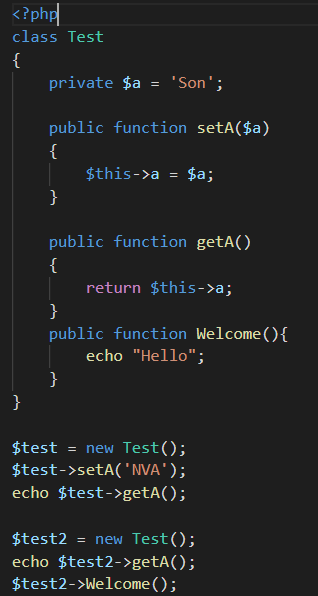




### Static

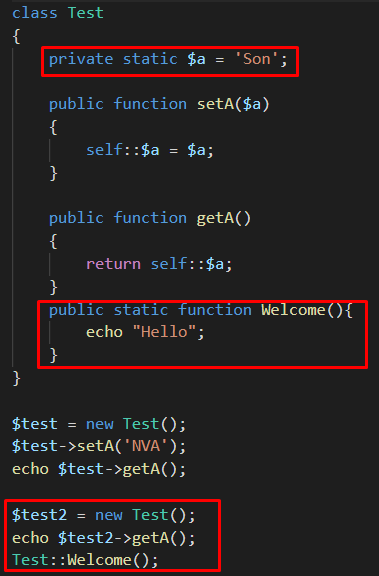
* Là một thành phần tĩnh mà nó hoạt động như một biến toàn cục. Nó lưu lại giá trị cuối cùng mà nó thực hiện

VD: Không khai báo thuộc tính static





VD: Với khai báo static



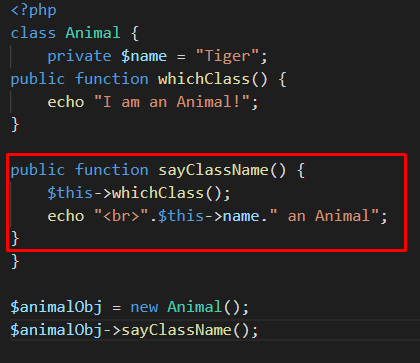


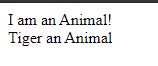
## This - self

### This

* Từ khóa this dùng để tham chiếu đến các thuộc tính, phương thức thường

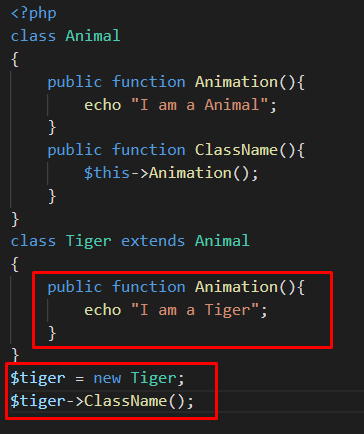
VD:





* Từ khóa this dùng để tham chiếu đến đối tượng (object).

VD:





Trong ví dụ này ta thấy khi gọi phương thức ClassName() bằng đối tượng $tiger mà trong phương thức ClassName() của lớp Animal lại dùng this để gọi đến phương thức Animation() nên nó sẽ hiểu là chuyển đến phương thức Animation() của lớp Tiger. Tức là this luôn luôn tham chiếu đến đối tượng.

* Từ khóa this không sử dụng được bên trong các thành phần tĩnh

VD

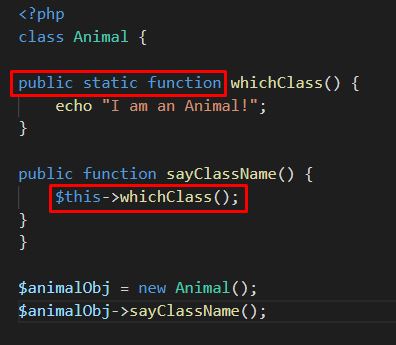




Như chúng thấy nó sẽ báo lỗi. Điều này chứng tỏ nó không có tác dụng khi được khai báo bên trong các phương thức tĩnh.

* Có thể giọi các hàm static bằng từ khóa this

VD:

* Không thể gọi các thuộc tính static bằng từ khóa this

VD:

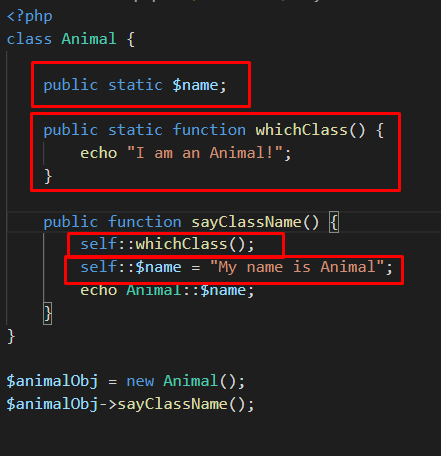




### Self

* Từ khóa self dùng để tham chiếu đến các thuộc tính, phương thức static

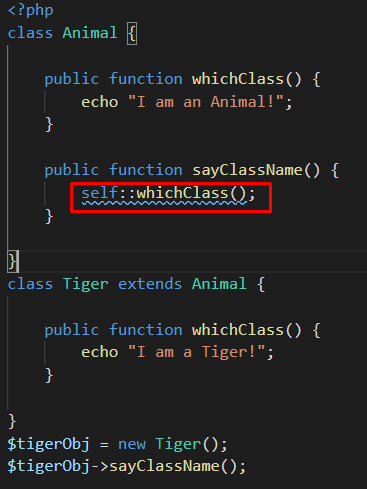
VD:





* Từ khóa self dùng để truy cập đến chính class

VD:

Trong ví dụ này khi đối tượng gọi hàm sayClasName() nhưng class Tiger không có phương thúc này nên nó sẽ lấy ở lớp cha nhưng do trong phương thức này gọi hàm wichClass() bằng từ khóa self nên nó gọi phương thức của chính lớp cha.

* Từ khóa self sử dụng được bên trong các thành phần thường

VD:

* Có thể giọi các hàm static bằng từ khóa self

VD



* Không thể gọi các thuộc tính thường bằng từ khóa self

|  |  |
| --- | --- |
| **This** | **Self** |
| Từ khóa this dùng để tham chiếu đến các thuộc tính, phương thức thường | Từ khóa self dùng để tham chiếu đến các thuộc tính, phương thức static |
| Từ khóa this dùng để tham chiếu đến đối tượng (object) | Từ khóa self dùng để truy cập đến chính class |
| Từ khóa this không sử dụng được bên trong các thành phần tĩnh | Từ khóa self sử dụng được bên trong các thành phần thường |
| Có thể giọi các hàm static bằng từ khóa this | Có thể giọi các hàm static bằng từ khóa self |
| Không thể gọi các thuộc tính static bằng từ khóa this | Không thể gọi các thuộc tính thường bằng từ khóa self |
| Làm việc với các hàm và thuộc tính thường | Làm việc với các hàm và thuộc tính static |

## Ngoại lệ

* Khi có một bài toán nếu như người dùng nhập những phép tính không hợp lệ thì php sẽ ném ra một lỗi và gây ra việc chương trình bị dừng lại nếu lỗi đó không được sửa. Và một lỗi được ném ra phải được sửa, đôi khi người viết code không sửa được lỗi được ném ra mà vẫn muốn chạy chương trình nên ngoại lệ được định nghĩa.
* VD: Một bài toán rút gọn phân số. Nhưng nếu mẫu số bằng 0 thì phân số không hợp lệ. Khi chạy chương trình người dùng cố ý nhập mẫu bằng 0 thì xảy ra lỗi.
* Tham khảo thêm tại: <https://docs.google.com/document/d/1yDuotFu01eJFHOOddfd1pW38Gwa0KcduuM5WA8iwFEQ/edit?usp=sharing> hoặc <https://niithanoi.edu.vn/vi-du-huong-dan-su-dung-try-catch-xu-ly-loi-va-ngoai-le-trong-php.html>

## Magic Method

* Magic methods là các phương thức đặc biệt được tạo ra nhằm giải quyết các vấn đề về sự kiện trong chương trình (cụ thể là với class), và đối với PHP cũng thế.
* Nó gồm có các ưu điểm như sau:
  + Giúp cho chúng ta tùy biến được các hành vi.
  + Nó giúp cho chúng ta có thể khởi tạo một đối tượng theo cách mình muốn.
* Các magic method cơ bản:
  + \_\_construct(): gọi khi khởi tạo đối tượng.
  + \_\_destruct(): goij khi hủy đối tượng.
  + \_\_set(): gọi khi ta truyền dữ liệu cho một thuộc tính không được phép truy cập.
  + \_\_get(): khi đọc dữ liệu từ một thuộc tính không được phép truy cập.
  + \_\_isset(): được gọi khi gọi hàm isset() hoặc empty() trên một thuộc tính không được phép truy cập.
  + \_\_unset(): được gọi khi hàm unset() được sử dụng trong một thuộc tính không được phép truy cập.
  + \_\_call():được gọi khi ta gọi một phương thức không được phép truy cập trong phạm vi của một đối tượng.
  + \_\_callstatic(): được kích hoạt khi ta gọi một phương thức không được phép truy cập trong phạm vi của một phương thức tĩnh.
  + \_\_toString(): phương thức này giúp class chỉ định xem sẽ in ra cái gì khi nó được dùng.
  + \_\_invoke():phương thức này được gọi khi một lệnh cố gắng gọi một đối tượng như một hàm.
  + \_\_sleep(): được gọi khi serialize() một đối tượng.
  + \_\_wakeup: được gọi khi unserialize() đối tượng.
  + \_\_set\_state():
  + \_\_clone(): được sử dụng khi chúng ta clone một object.
  + \_\_debugInfo(): được gọi khi chúng ta sử dụng hàm vardump().

## Namespace

* Namespace giúp tạo ra một không gian tên cho hàm và lớp trong lập trình nói chung và trong PHP nói riêng.
* Cú pháp: namespace tenNamespace;
* Mỗi file chứa 1 namespace phải được khai báo trên cùng trước các code khác trong file. Cách thực hành tốt nhất là mỗi file chứa 1 class với tên class trùng với tên file đồng thời có 1 namespace mô tả được đường dẫn tới file đó
* Cú pháp sử dụng: use tenNamespace\tenClass
* Ta cũng có thể định danh lại cho namespace nếu tên của nó quá dài: use tenNamespace as tenMoi;

VD: Các bước thực hiện autoload bằng composer sử dụng namespace:

Bước 1: Tạo thư mục OOP

├── composer.json

├── Ex1.php

└── src

   ├── SP1

   │   └── SP1.php

   ├── SP2

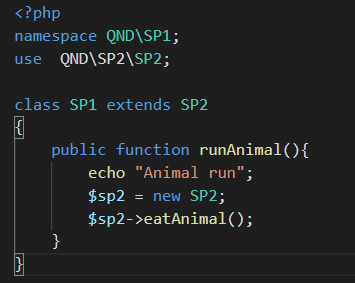
   │   └── SP2.php

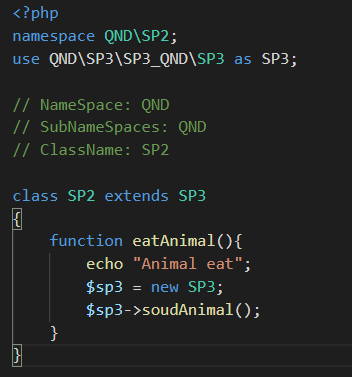
   └── SP3

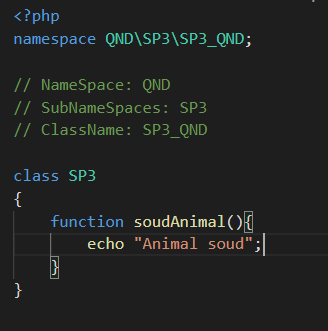
└──SP3\_QND

   └── SP3.php

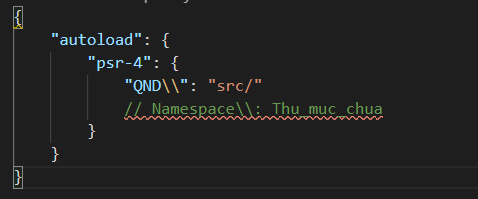
Bước 2: Tạo các file SP1.php, SP2.php, SP3.php

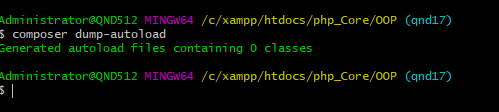


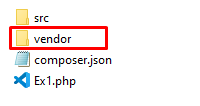




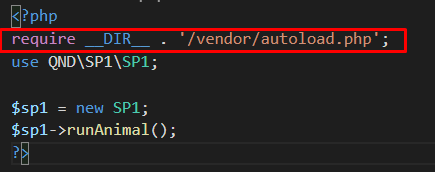
Bước 3: Tạo composer.json thực hiện autoload sau đó chạy lệnh: composer dump-autoload







Bước 4: Trong file Ex1.php gọi tới autoload để load các namespace



Bước 5: Chạy chương trình



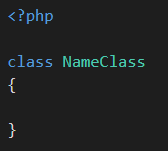
## Quy chuẩn code

### PSR-1

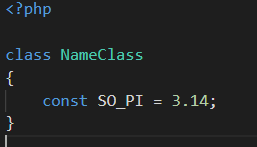
* Dùng thẻ <?php hoặc <?=

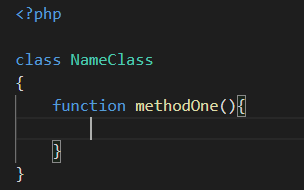
* Các file phải dùng duy nhất UTF-8 không BOM cho code PHP
* Các file nên hoặc định nghĩa kí hiệu (lớp, hàm, hằng số, vv.) hoặc đưa ra tác dụng phụ (e.g. sinh ra output, thay đổi .ini settings, vv.) nhưng không nên làm cả 2
* Các namespace và các class phải tuân theo các chuẩn PSR-4
* Tên class phải được định nghĩa ở dạng StudlyCaps



* Hằng số phải được định nghĩa bằng chữ cái in hoa và cách nhau bằng dấu gạch dưới.

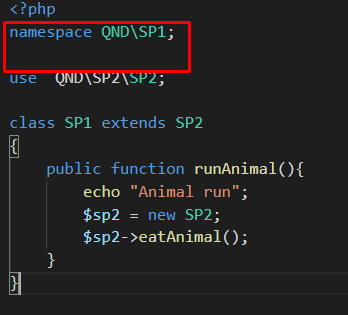


* Các phương thức phải đặt tên dạng camelCase

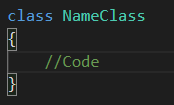


### PSR-2

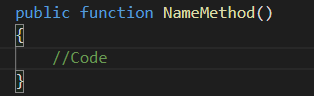
* Tuân thủ PSR-1
* Code phải sử dụng 4 ký tự space để lùi khối
* Mỗi dòng code phải dưới 120 ký tự, Nên dưới 80 ký tự
* Phải có 1 dòng trắng sau namespace, và phải có một dòng trắng sau mỗi khối code



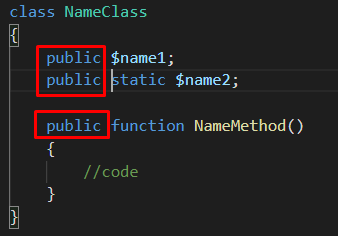
* Ký tự mở lớp { phải ở dòng tiếp theo, và đóng lớp } phải ở dòng tiếp theo của thân class



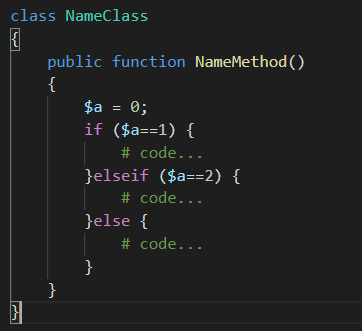
* Ký tự { cho hàm phải ở dòng tiếp theo, và ký tự } kết thúc hàm phải ở dòng tiếp theo của thân hàm



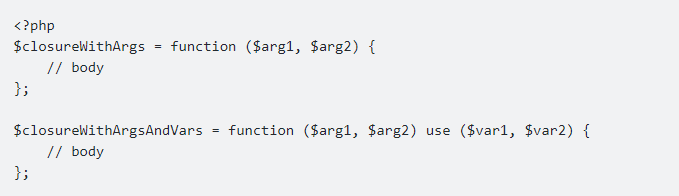
* Các visibility (public, private, protected) phải được khai báo cho tất cả các hàm và các thuộc tính của lớp



* Các từ khóa điều khiển khối(if, elseif, else) phải có một khoảng trống sau chúng; hàm và lớp thì không được làm như vậy.



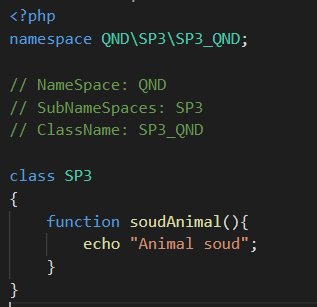
* Mở khối { cho cấu trúc điều khiển phải trên cùng một dòng; và đóng khối này với } ở dòng tiếp theo của thân khối.
* Hằng số true, false, null phải viết với chữ thường.
* Từ khóa extends và implements phải cùng dòng với class Implements nhiều lớp, thì mỗi lớp trên một dòng Keyword và không được dùng sử dụng khai báo property.
* Tên property không nên có tiền tố \_ nhằm thể hiện thuộc tính protect hay private.
* Tham số cho hàm, phương thức: Không được thêm space vào trước dấu ‘,’ và phải có một space sau ‘,’.



* Các tham số có thể trên nhiều dòng, nếu làm như vậy thì phải mỗi dòng 1 tham số. Abstract, final PHẢI đứng trước visibility, còn static phải đi sau

### PSR-4

* Cú pháp: \<NamespaceName>(\<SubNamespaceNames>)\*\<ClassName>
* Trong đó: Mỗi lớp phải xây dựng sao cho có thể được tham chiếu đến bởi dòng code ba thành phần: Namespace, Các SubNamespaceNames con, và tên lớp ClassName
  + NameSpace : Tiền đố đầu tiên bắt buộc phải có - được hiểu là tên vendor. Tên này do bạn tự đặt, sao cho không xung đột tên các thư viện khác.
  + SubNameSpaces: Các namespace con. Có một hoặc nhiều tùy bạn. Nhưng bắt đầu từ SubNameSpace thì nó tương ứng với cấu trúc thư mục lưu trữ code.
  + ClassName: Bắt buộc phải có và phải có tên file PHP trùng tên ClassName ở thư mục tương ứng với namespace cuối cùng (ClassName.php), trong file đó sẽ định nghĩa nội dung của code của lớp.



# Tính đóng gói – Encapsulation

## Định nghĩa

## 2.2. Access Modifier

## 2.3. Geter – Seter

# Tính kế thừa – Inheritance

## Định nghĩa

## Đa kế thừa

## Ghi đè

## Traint

# Tính đa hình – Polymorphism

## Định nghĩa

## Nạp chồng

# Tính trừu tượng – Abstraction

## Định nghĩa

## Abstract class

## Intereface

# Website PHP

## Debug PHP

## PDO

### CRUD

# Các mẫu thiết kế

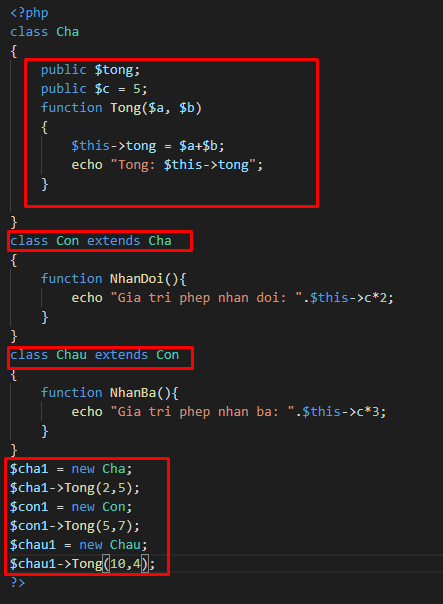
## SOLID

## MVC

# Zend frameword

1. Kế thừa

* Trong PHP có tính kế thừa nhiều cấp độ



1. Mức độ truy cập

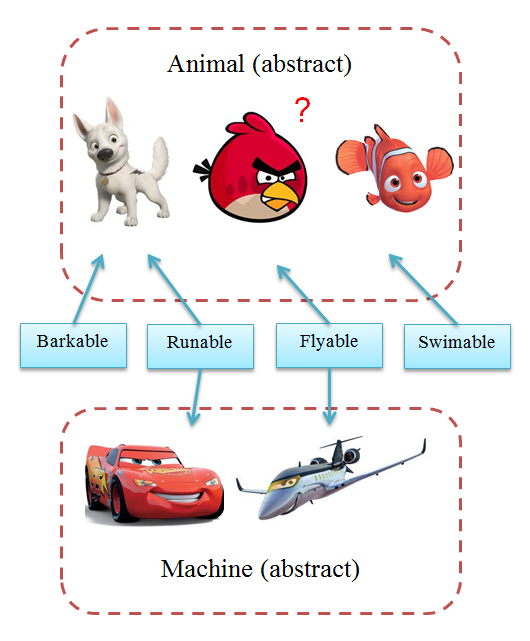
* Public: Các đối tượng bên ngoài class có thể truy cập
* Private: Các đối tượng bên trong class mới có thể truy cập. Và không có tính kế thừa
* Protected: Các đối tượng bên trong class mới có thể truy cập. Và có tính kế thừa tức là lớp con có thể gọi được thuộc tính của lớp cha
* Set và get: Cho phép các đối tượng bên ngoài class xử lý được các thuộc tính có mức độ truy cập protected và private

1. This và self

* This là từ khóa dùng để gọi các biến thuộc các mức độ truy cập public, private, protected. VD: $this->a
* Self là từ khóa dùng để gọi các biến tĩnh dạng static. VD: self::a

1. Abstract Class và Interface

|  |  |
| --- | --- |
| Interface | Abstract |
| **Interface là giao diện** được đưa ra cho user biết được sẽ làm việc với method nào | **Abstract class là một class** và nó có các tính chất giống với 1 class thông thường |
| User sẽ không biết được cụ thể method đó sẽ làm được những cái gì |  |
| **Interface chỉ có abstract method** có nghĩa là method trống không có thực thi | **Có thêm abstract method** |
| Trong Interface chỉ được phép có thêm const (hằng) |  |
| 1 class phải kế thừa abstract method từ interface | 1 class phải kế thừa abstract method từ abstract class |
| 1 class có thể implement được nhiều interface | 1 class chỉ extent được một class |
| Các phương thức trong interface luôn luôn là public vì nó là giao diện đưa ra cho các user dùng chung | Các phương thức trong abstract class có thể là public, protected hoặc private vì nó mang tính chất là một lớp |
| Khi nhóm đối tượng không cùng bản chất nhưng cùng một hành động thì sử dụng interface | Khi nhóm đối tượng cùng bản chất, cùng hành động thì dùng abstract class |



* Trong đó: Barkable, Runable, Flyable, Swimable, là các interface

1. Ghi đè

* Khi class con kế thừa từ lớp cha, nó có thể ghi đè lên phương thức mà lớp cha đã khai báo.
* Phương thức final và static sẽ không thể ghi đè