# 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

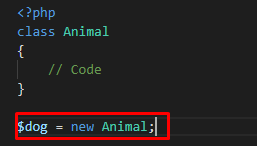
{

    // Code

}

## Object – Đối tượng

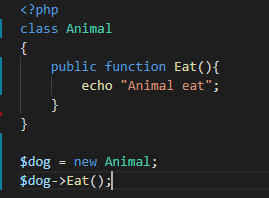
Định nghĩa: Đối tượng trong một ngôn ngữ OOP là các kết hợp giữa mã và dữ liệu mà chúng được nhìn nhận như là một đơn vị duy nhất. Mỗi đối tượng có một tên riêng biệt và tất cả các tham chiếu đến đối tượng đó được tiến hành qua tên của nó.



## Method – Phương thức

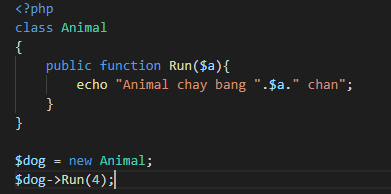
Định nghĩa: Phương thức giống với định nghĩa hàm trong lập trình cấu trúc. Phương thức là các hàm thể hiện hành vi của một đối tượng thuộc lớp. Có các loại phương thức của có đối, phương thức không đối, phương thức trả về, phương thức khởi tạo và phương thức hủy.

* + 1. Phương thức không đối

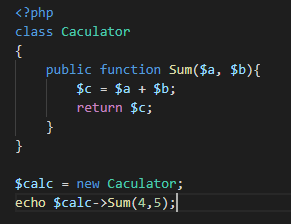




* + 1. Phương thức có đối



* + 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

## Contructor – Destructor

## Final – Static

## This - self

## Ngoại lệ

## Namespace

# Tính đóng gói – Encapsulation

## 2.1. 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

## Zend frameword

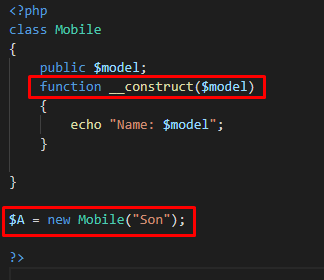
# Các mẫu thiết kế

## SOLID

## MVC

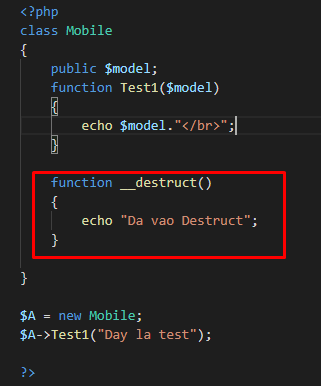
1. Contructor PHP

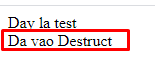
* Nếu khai báo hàm contructor có biến thì khi khai báo đối tượng ta phải truyền luôn giá trị vào đối tượng đó.
* Hàm contructor sẽ được chạy đầu tiên khi ta tạo đối tượng có thể hiện của class



1. Destruct

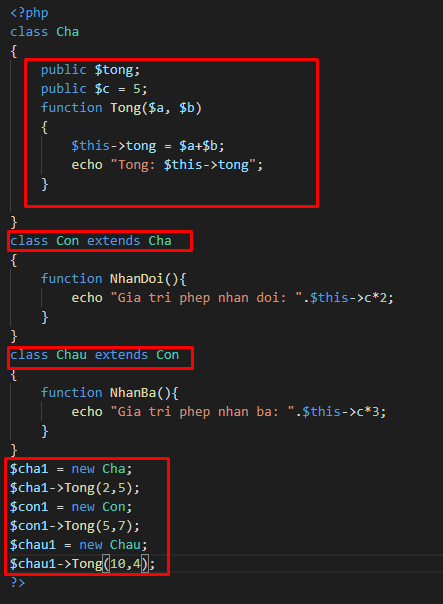
* Hàm hủy, hàm này luôn dùng cho đóng collection của database





1. Kế thừa

* Trong PHP có tính kế thừa nhiều cấp độ



1. Mức độ truy cập

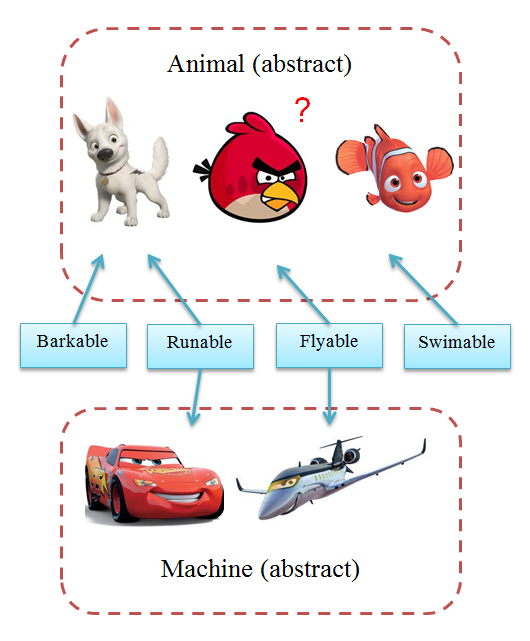
* Public: Các đối tượng bên ngoài class có thể truy cập
* Private: Các đối tượng bên trong class mới có thể truy cập. Và không có tính kế thừa
* Protected: Các đối tượng bên trong class mới có thể truy cập. Và có tính kế thừa tức là lớp con có thể gọi được thuộc tính của lớp cha
* Set và get: Cho phép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