1. **OOP (viết tắt của Object Oriented Programming)** – lập trình hướng đối tượng là một phương pháp lập trình dựa trên khái niệm về lớp và đối tượng. OOP tập trung vào các đối tượng thao tác hơn là logic để thao tác chúng.

**Các tính chất chính của OOP:**

- **Tính đóng gói**:Các đối tượng khác nhau của mỗi chương trình sẽ cố gắng tương tác với nhau một cách tự động. Nếu một lập trình viên muốn ngăn cản sự tương tác giữa các đối tượng thì phải đóng gói các đối tượng vào những class riêng biệt. Thông qua đóng gói, các class sẽ không thay đổi và tương tác với những biến hay hàm cụ thể của một đối tượng.

- **Tính trừu tượng**: Tính trừu tượng giống như một phiên bản mở rộng của tính đóng gói vì nó giấu đi những tính chất và phương thức cụ thể để giao thức của các đối tượng đơn giản hơn. Lập trình viên sử dụng tính trừu tượng cho vài lý do có ích khác. Nhìn chung, tính trừu tượng giúp cô lập ảnh hưởng của sự thay đổi mã code. Mục tiêu là nếu có sai sót gì xảy ra, ảnh hưởng của sự thay đổi là không nhiều.

- **Tính kế thừa**: Để tái sử dụng logic chung này và trích những logic cụ thể vào một class riêng, bạn có thể sử dụng tính kế thừa. Có nghĩa là bạn tạo ra một lớp con từ lớp cha. Bằng cách này, chúng ta tạo được một trật tự. Lớp con sẽ sử dụng tất cả các field và method của lớp cha và có thể có thêm những phần riêng biệt của nó.

- **Tính đa hình**: Tính đa hình đưa ra cách sử dụng một lớp con giống hệt như lớp cha để không có sự nhầm lẫn, bối rối nào giữa các dạng khác nhau. Nhưng mỗi lớp con vẫn giữ nguyên method của mình. Điều này thường xảy ra khi tái sử dụng một giao thức lớp cha. Nó đưa ra những method phổ biến, rồi mỗi lớp con thực hiện phiên bản method riêng của nó.

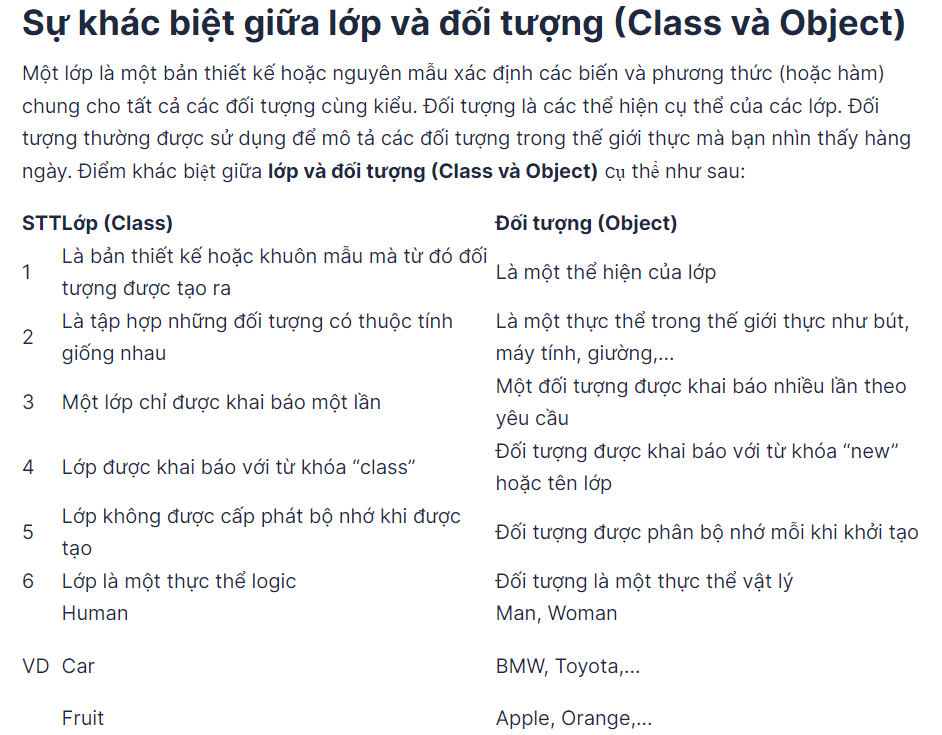
1. **Ưu điểm của lập trình hướng đối tượng:**

* OOP mô hình hóa những thứ phức tạp dưới dạng cấu trúc đơn giản.
* Code OOP có thể sử dụng lại, giúp tiết kiệm tài nguyên.
* Giúp sửa lỗi dễ dàng hơn. So với việc tìm lỗi ở nhiều vị trí trong code thì tìm lỗi trong các lớp (được cấu trúc từ trước) đơn giản và ít mất thời gian hơn.
* Có tính bảo mật cao, bảo vệ thông tin thông qua đóng gói.
* Dễ mở rộng dự án.

1. **Khái niệm Đối tượng (object) và lớp (class). Phân biệt sự giống/khác nhau giữa Đối tượng (object) và lớp (class). Lấy ví dụ minh họa.**

- Đối tượng (Object): Một đối tượng bao gồm các phương thức (method) và thuộc tính(property) để tạo thành một kiểu dữ liệu cụ thể hữu ích. Đối tượng xác định hành vi của lớp. Khi bạn gửi tin nhắn đến một đối tượng, bạn đang yêu cầu đối tượng gọi hoặc thực thi một trong các phương thức của nó.

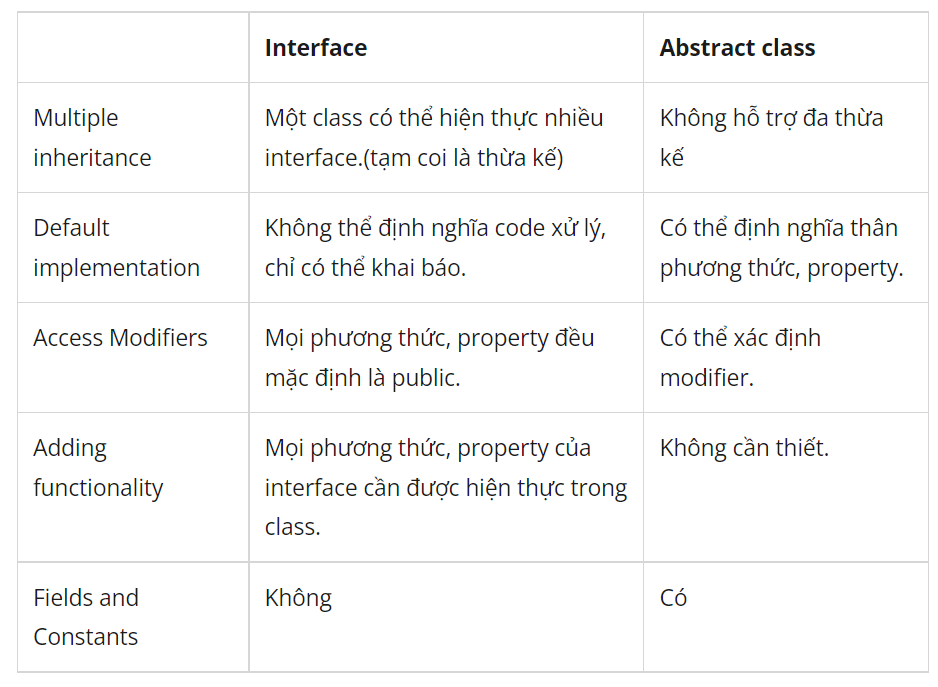
- Lớp (Class): Một lớp là một thực thể xác định cách một đối tượng sẽ hành xử và những gì đối tượng sẽ chứa. Nói cách khác, đó là một kế hoạch chi tiết (blueprint) hoặc một bộ hướng dẫn để xây dựng một loại đối tượng cụ thể.



1. **Khái niệm interface và abstract class? So sánh giữa interface và abstract class? Lấy ví dụ minh họa.**

– **Abstract class:** là một class cha cho tất cả các class có cùng bản chất. Bản chất ở đây được hiểu là kiểu, loại, nhiệm vụ của class. Hai class cùng hiện thực một interface có thể hoàn toàn khác nhau về bản chất. Hiểu đơn giản như một thằng con (child class) chỉ có thể là con của một thằng cha, có tính cách giống cha (abstract class) nó.

– **Interface**: là một chức năng mà bạn có thể thêm và bất kì class nào. Từ chức năng ở đây không đồng nghĩa với phương thức (hoặc hàm). Interface có thể bao gồm nhiều hàm/phương thức và tất cả chúng cùng phục vụ cho một chức năng.



1. **Khái niệm Traits? Lấy ví dụ minh họa:**

**Trait** có thể hiểu như một class, là nơi tập hợp một nhóm phương thức (method) mà chúng ta muốn sử dụng trong class khác. Cũng giống như **Abstract class**, chúng ta không thể khởi tạo một đối tượng từ Trait.

Graphical user interface, text, application, email

Description automatically generated

1. **Khái niệm Namespaces. Cách sử dụng. Lấy ví dụ minh họa**

Khi một trang web trở nên lớn hơn, việc giữ tất cả các file class trong cùng một thư mục sẽ khó hơn. Ngoài ra, chúng ta phải sử dụng tên rất dài cho các file và class, Đó là một trở ngại trong quản lý mã nguồn. Namespaces được giới thiệu để ngăn chặn vấn đề này.

**Text

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

Ở đây thay vì phải truyền đầy đủ namespace vào từ khóa new khi khởi tạo đối tượng, bạn chỉ cần gõ tên class, PHP sẽ tự biết chỗ nhờ lệnh use phía trên để nạp class Iphone trong thư mục smartphone.

1. **Khái niệm Static Properties. Cách sử dụng. Lấy ví dụ minh họa**

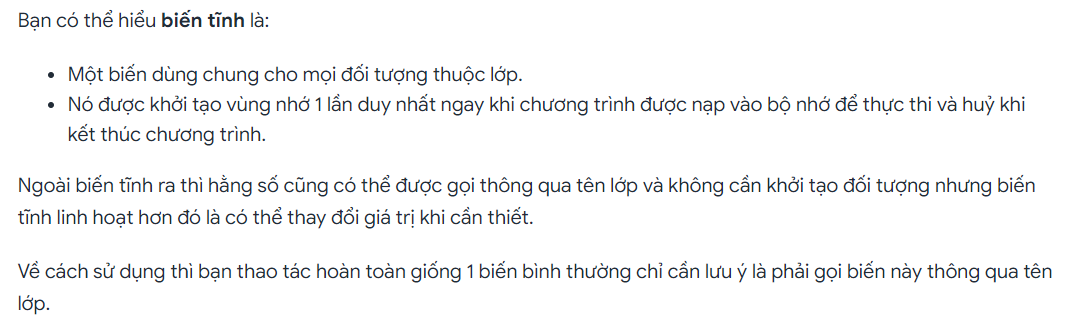
- **Biến Static (Static Variables):** Ta có thể thấy rằng, Static dịch ra là Tĩnh. Chính vì vậy, biến Static hay được gọi là biến tĩnh khi ta khai báo chúng.

- **Phương thức Static (Static Methods):** Khi người dùng tiến hành khai báo một phương thức dưới dạng Static thì nó được gọi là phương thức Static.

- **Khối Static (Static Blocks):** Khi muốn khởi tạo các thành viên thuộc nhóm dữ liệu Static, ta sẽ tiến hành sử dụng khối Static.

- **Lớp Static (Static Class):** Người dùng có thể đặt các lớp lồng nhau thành dạng Static. Các lớp lồng nhau này có thể được truy cập mà không cần đến object của các lớp bên ngoài.

- **Import Static:** Ta được phép tiến hành Import các Static Member của một lớp hoặc một package vào một lớp khác bằng phương thức sử dụng từ khóa Import với các phiên bản Java từ 5 trở lên. Sau đó, người dùng có thể vận dụng Import Static đó như một thành viên của lớp.



Graphical user interface, text, application, email

Description automatically generated

**Text

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

1. **Phân biệt cách dùng từ khoá static::method() với self::method(). Cách sử dụng. Lấy ví dụ minh họa.**

Nhìn chung thì cả self và static đều dùng để gọi các thành phần tĩnh trong đối tượng, nhưng nếu chỉ đơn thuần như trong nội bộ class thì cả 2 keywords này đều cho ra kết quả tốt.

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text

Description automatically generated**