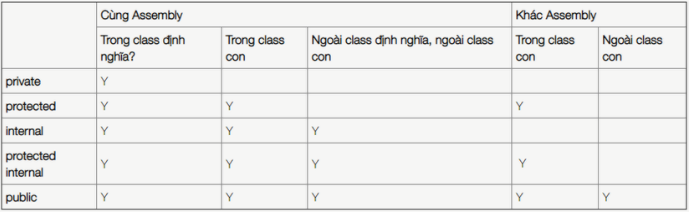
**Lập trình hướng đối tượng(OOP – Object Oriented Programming)**

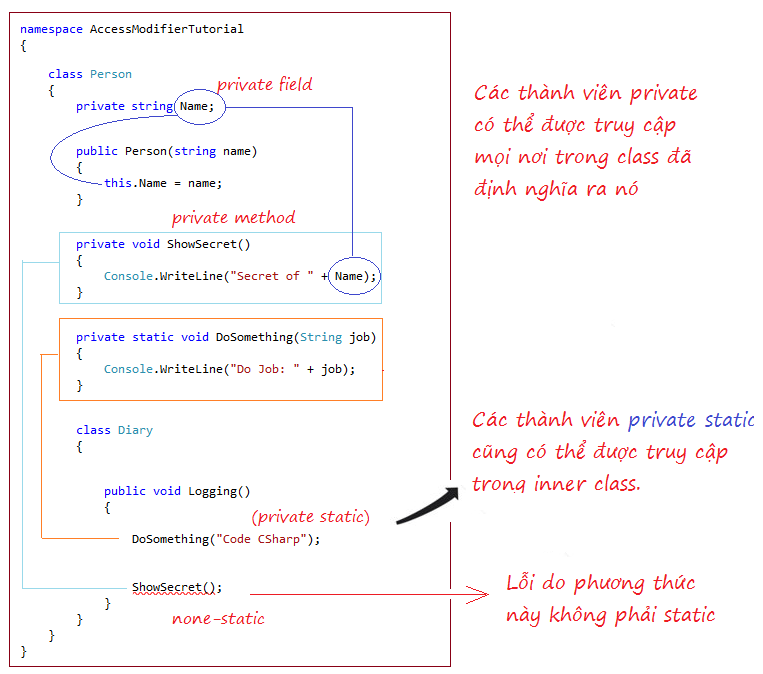
**Tính chất:**

- **Encapsulation**(Đóng gói): Ngăn cản việc truy cập tới chi tiết của trình triển khai(được hiển thị qua 5 access modifier: **private**, **protected**, **internal**, **protected** **internal**, **public**)

**Access modifier:**



1. **Private** : Truy cập hạn chế trong phạm vi định nghĩa của **class**, Nếu không được chỉ định thì đây sẽ là giá trị default (ngoài ra, đối với những **inner class** thì cần sử dụng từ khóa **static**)



2. **Protected**: Truy cập giới hạn trong phạm vi class hoặc class kế thừa từ class này.

3. **Internal**: Truy cập giới hạn trong phạm vi **Assembly** của dự án hiện tại

1. *Tạo 1 project với loại Library Class*
2. *Tạo class có chưa internal*
3. *Build project để tạo file .dll*
4. *Tạo 1 project mới với dạng bất ký như console,*[*asp.net*](http://asp.net/)*,...*
5. *Add file dll vừa tạo vào dự án này và bạn có thể thấy các method loại internal sẽ không thể truy cập ở đây.*

4. **Protected internal**: Là kết hợp giữa **protected** và **internal**. Giới hạn truy cập trong một **Assembly** và class đó hoặc class con kế thừa nó.

5. **Public**: Truy cập không bị giới hạn

- **Inheritance**(Kế thừa): Cho phép định nghĩa một lớp trong điều kiện một lớp khác giúp tái sử dụng code. (C# không hỗ trợ đa kế thừa, có thể sử dụng interface để triển khai đa kế thừa)

+ ***Interface***: Được định nghĩa như là một giao ước có tính chất cú pháp(syntactical contact) mà tất cả lớp kế thừa Interface đó nên theo.

+ ***Các tính chất đặt biệt của Interface***:

* Bắt buộc các **method** trong **interface** là **public** và **abstract**
* Không thể khai báo các **field** trong **interface**
* Các **method** không có thân hàm
* Không có **constructor**
* 1 class có thể **implement** n **Interface**
* Class **implement** **Interface** thì phải chứa đầy đủ các **method** có trong **interface** đó

- **Abstraction**(trừu tượng): Ẩn các tiến trình triển khai và chỉ hiển thị tính năng tới người dùng. Tính trừu tượng cho phép bạn loại bỏ tính chất phức tạp của đối tượng và chỉ đưa ra các thuộc tính và phương thức cần thiết. Thể hiện qua từ khóa **Abstract** và **Interface**

+ ***Các tính chất đặt biệt của Abstract class***:

* Trong **abstract** class có thể khai báo **field**
* 1 class chỉ có thể kế thừa 1 **abstract** class
* Có **constructor**
* Các method có thể chứa các **access modifier**(private, protected, public,…) chứ không bắt buộc phải public như **Interface**
* Ở class con kế thừa có thể có hoặc không có các method cơ sở(chỉ bắt buộc có các **field** cơ sở)
* Không thể tạo 1 **Instance** của **abstract** class
* **Abstract** class không thể được khai báo là **sealed**

- **Polymorphism**(Đa hình): Đối tượng thuộc các class khác nhau có thể hiểu cùng 1 thông điệp theo cách khác nhau

**Abstract class và Interface**

<https://o7planning.org/vi/10431/abstract-class-va-interface-trong-csharp>

4 tính chất của OOP

<https://viblo.asia/p/4-tinh-chat-cua-lap-trinh-huong-doi-tuong-trong-c-RnB5pBEDZPG>