# Inversion of Control

# I.Mục lục

# 1. Định nghĩa về Inversion of Control DP

# 3. Những trường hợp cần sử dụng Inversion of Control DP

# 2. Các giải pháp/ hiện thực Inversion of Control DP

# II.Nội dung

# Định nghĩa về Inversion of Control DP

# Inversion of Control DP là một design pattern được dùng để code tuân theo nguyên tắc chữ D trong 5 nguyên tắc SOLID, nguyên tắc Dependency Inversion Principle (DIP).

# Giới thiệu về DIP: Trong thực tế khi chúng ta code sẽ tạo ra những phụ thuộc, ràng buộc lẫn nhau (dependency) . DIP yêu cầu :

# Các module cấp cao không được phụ thuộc module cấp thấp mà phải thông qua abstraction.

# Các interface không phụ thuộc vào chi tiết mà ngược lại, được dùng để các class giao tiếp với nhau

# MINH HỌA

# 

# Giải thích: Bạn có thể dễ dàng hình dung qua ví dụ sau: Bạn thèm ăn gà giòn vui vẻ của KFC hoặc Jolibee, họ sẽ phải giao gà cho bạn. Lúc này có thể xem bạn là module cấp thấp và họ là module cấp cao. Tuy nhiên họ kinh doanh gà nên không thể lúc nào cũng giao gà cho bạn được, sẽ có đôi lúc họ phát triển thêm thị phần, hoặc phân khúc khác, họ sẽ ship gà không kịp. Do đó họ sẽ làm việc với một bộ phận shipper để giao gà cho bạn. Vấn đề được giải quyết, shipper ở đây bạn có thể hình dung là interface.

Ngược lại, vi phạm DIP, khi sự phụ thuộc tồn tại thì chúng ta muốn đổi sự phụ thuộc buộc chúng ta phải thay đổi thuộc tính, phương thức của nó, trong khi đó việc giao tiếp qua trừu tượng giúp giữa chúng có thay đổi sẽ không ảnh hưởng nhau.

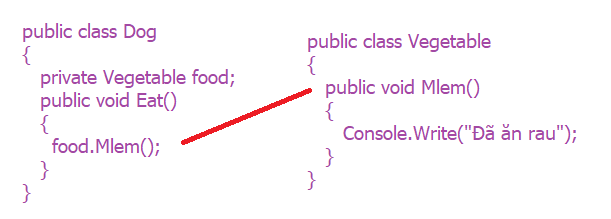
Quay lại về **Inversion of Control** , dịch nôm na là đảo ngược điều khiển (!!)   
 Tức là thay vì một lớp sẽ chịu sự phụ thuộc ràng buộc với những lớp khác (giống hình minh họa) thì với IoC nó sẽ tự do, không bị ràng buộc bởi các lớp khác, các lớp khác cũng không bị ràng buộc mà chúng liên kết nhau qua cách khác như khởi tạo, setter,interface (như ví dụ là shipper). Trong code thì khó áp dụng nhất đó là cách áp interface vào, tuy nhiên đó là cách dễ mở rộng, dễ phát triển sửa chữa nhất.

Ioc được chia làm nhiều cách hiện thực bao gồm: Service Locator, Event, Delegate,.. và Dependency Injection là một trong số cách hiện thực IoC.

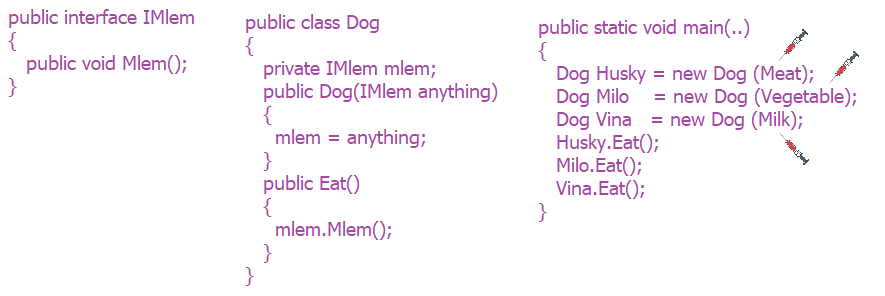
Dependency Injection được xem như là một trạm trung gian để kết nối các object với nhau và đồng thời tiêm những ràng buộc một cách thích hợp. Dễ dàng hình dung đó là shipper có thể ship bất cứ món ăn nào mà tiệm gà cung cấp cho bạn, khi có thay đổi món ăn hoặc khách hàng, họ chỉ cần tiêm ràng buộc thích hợp, tức là họ sẽ đổi địa chỉ giao hàng và món hàng thôi (^^)

2.**Các giải pháp/ hiện thực Inversion of Control DP**

Đầu tiên chúng ta cùng đi qua thử đoạn code sau:



Như trên chúng ta thấy giữa lớp Dog và lớp Vegetable có sự phụ nhau, lúc này nếu chúng ta có loài chó mới muốn ăn thịt hoặc uống sữa thì sẽ rất vướng víu , khó khăn để sửa đổi code do Dog đã phụ thuộc vào Vegetable dẫn đến toàn bộ loài chó trong thế giới của chúng ta chỉ được ăn rau. Sau đây là một cách mà Dependency Injection hoạt động dùng Interface để tách sự phụ thuộc ra và tiêm vào trong lúc khởi tạo:



Từ đó ta thấy loài chó có thể thoải mái ăn mọi thứ trên đời !!

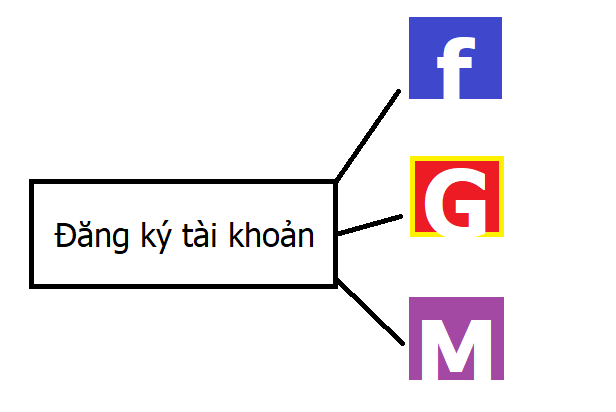
Tuy nhiên chó là động vật ăn tạp, hôm nay có thể nó ăn bánh nhưng hôm sau nó sẽ uống sữa, ăn cà lem, vân vân… Cũng vì vậy chúng ta cũng phải tự tay cho nó ăn, tự tay code những dòng inject vào ( thông qua setter chẳng hạn ) rất là tốt sức. Do đó sẽ có DIContainer giữ trách nhiệm lưu trữ và inject (giống như một kho) các component dependency trong vòng đời của nó.

Có rất nhiều thư viện DIContainer như: Windsor, Unity, Ninject,.. trong đó DependencyInjection là một thư viện mặc định của ASP.NET

3.**Những trường hợp cần sử dụng Inversion of Control DP**

Chúng ta nên sử dụng IoC DP khi giữa các dependency trong hệ thống có khả năng lớn phát sinh nhiều thay đổi phức tạp. Ví dụ trang phục của nhân vật trong game vì mỗi khi update sẽ có skin mới chẳng hạn. Cũng vậy chúng ta nên tránh IoC DP khi giữa các dependency đó ít có khả năng phát sinh thay đổi nhất, hoặc hệ thống chỉ nhỏ để thực hiện đủ chức năng, không cần phát sinh thêm phức tạp rườm rà. Ví dụ điểm của học sinh cấp 3, thông thường các môn sẽ có các thang điểm như 0-10 hoặc 0-100, ít phát sinh loại điểm kì dị gì khác nên không cần phải áp dụng ở đây. Giữ gìn sự gọn gàng, dễ đọc của code.

Chúng ta sử dụng IoC DP sau khi đã chắc chắn sau khi sử dụng thì hệ thống vẫn bảo mật, tốt. Ngược lại, áp dụng IoC DP một cách vô lý (vì thực chất DIContainer phát sinh biến toàn cục) sẽ dẫn đến không bảo mật.



Lấy ví dụ hình bên,

Tuy rằng việc đăng ký tài khoản có nhiều cách, nhưng ta không nên áp dụng IoC DP vào !!!!