**Tìm hiểu về Bridge Pattern**

1. Bridge Pattern là gì

Bridge Pattern là một trong những Pattern thuộc nhóm Structural Pattern. Ý tưởng của nó là tách tính trừu tượng (abstraction) ra khỏi tính hiện thực (implementation) của nó. Từ đó có thể dễ dàng chỉnh sửa hoặc thay thế mà không làm ảnh hưởng đến những nơi có sử dụng lớp ban đầu.

Bridge Pattern khá giống với mẫu Adapter Pattern ở chỗ là sẽ nhờ vào một lớp khác để thực hiện một số xử lý nào đó. Tuy nhiên, ý nghĩa và mục đích sử dụng của hai mẫu thiết kế này hoàn toàn khác nhau:

* Adapter Pattern được dùng để biến đổi một class / interface sang một dạng khác có thể sử dụng được.
* Bridge Patter được sử dụng để tách thành phần trừu tượng (abstraction) và thành phần thực thi (implementation) riêng biệt.
* Adapter Pattern làm cho mọi thứ có thể hoạt động với nhau sau khi chúng đã được thiết kế. Bridge Pattern nên được thiết kế trước khi phát triển hệ thống để Abstraction và Implementation có thể thực hiện một cách độc lập.

2. Cài đặt Bridge Pattern

Một Bridge Pattern bao gồm những thành phần sau:

* Client: đại diện cho khách hàng sử dụng các chức năng thông qua Abstraction
* Abstraction: định ra một abstract interface quản lý việc tham chiếu đến đối tượng hiện thực cụ thể (Implementor)
* Refined Abstraction: hiện thực (implement) các phương thức đã được định ra trong Abstraction bằng cách sử dụng một tham chiếu đến đối tượng của Implementer
* Implementor: định ra các interface cho các lớp hiện thực. Thông thường nó là interface định ra các tác vụ nào đó của Abstraction
* ConcreteImplementor: Hiện thực Implementor interface

3. Lợi ích của Bridge Pattern

- Giảm sự phụ thuộc giữa abstraction và implementation (loose coupling): tính kế thừa trong OOP thường gắn chặt abstraction và implementation trong lúc build chương trình. Bridge Pattern có thể được dùng để cắt đứt sự phụ thuộc này và cho phép chúng ta chọn implementation phù hợp lúc runtime.

- Giảm số lượng những lớp con không cần thiết: một số trường hợp sử dụng tính inheritance sẽ tăng số lượng subclass rất nhiều. Trong khi áp dụng Bridge sẽ giảm số lượng của chúng xuống.

- Code sẽ gọn gàng hơn và kích thước ứng dụng sẽ nhỏ hơn: do giảm được số lượng subclass không cần thiết

- Dễ bảo trì hơn: các Abstraction và Implementation của nó sẽ dễ dàng thay đổi lúc runtime cũng như khi cần thay đổi thêm bớt trong tương lai

- Dễ dàng mở rộng về sau: thông thường các ứng dụng lớn thường yêu cầu chúng ta thêm module cho ứng dụng có sẵn nhưng không được sửa đổi framwork/ ứng dụng có sẵn vì các framework/ ứng dụng đó có thể được công ty nâng cấp lên version mới. Bridge Pattern sẽ giúp chúng ta trong trường hợp này.

- Cho phép ẩn các chi tiết implement từ client: do abstraction và implementation hoàn toàn độc lập nên chúng ta có thể thay đổi một thành phần mà không ảnh hưởng đến phía client.

4. Sử dụng Bridge Pattern khi nào?

- Khi bạn muốn tách ràng buộc giữa Abstraction và Implementation, để có thể dễ dàng mở rộng độc lập với nhau

- Cả Abstraction và Implementation của chúng nên được mở rộng bằng subclass

- Sử dụng ở những nơi mà những thay đổi được thực hiện trong implement không ảnh hưởng đến phía client