**Tìm hiểu Decorator Pattern**

1. Decoration Pattern là gì

Decoration Pattern là một trong những Pattern thuộc nhóm cấu trúc (Structural Pattern). Nó cho phép người dùng thêm chức năng mới vào đối tượng hiện tại mà không muốn ảnh hưởng đến các đối tượng khác. Kiểu thiết kế này có cấu trúc hoạt động như một lớp bao bọc (wrap) cho lớp hiện có. Mỗi khi cần thêm tính năng mới, đối tượng hiện có được wrap trong một đối tượng mới.

2. Cài đặt Decorator Pattern

Decorator Pattern hoạt động dựa trên một đối tượng đặc biệt, được gọi là decorator (hay wrapper). Nó có cùng một interface như một đối tượng mà nó cần bao bọc (wrap), vì vậy phía client sẽ không nhận thấy khi bạn đưa cho nó một wrapper thay vì đối tượng gốc.

Tất cả wrapper có một trường để lưu trữ một giá trị của một đối tượng gốc. Hầu hết các wrapper khởi tạo trường đó với một đối tượng được truyền vào constructor của chúng.

Vậy làm thế nào để có thể thay đổi hành vi của đối tượng? Như đã đề cập, wrapper có cùng interface với các đối tượng đích. Khi bạn gọi một phương thức decorator, nó thực hiện cùng một phương thức trong một đối tượng được wrap và sau đó thêm một cái gì đó (tính năng mới) vào kết quả, công việc này tùy thuộc vào logic nghiệp vụ.

Các thành phần trong mẫu thiết kế Decorator:

* Component: là một interface quy định các method chung cần phải có cho tất cả các thành phần tham gia vào mẫu này.
* ConcreteComponent: là lớp hiện thực (implements) các phương thức của Component.
* Decorator: là một abstract class dùng để duy trì một tham chiếu của đối tượng Component và đồng thười cài đặt các phương thức của Component interface.
* ConcreteDecorator: là lớp hiện thực (implements) các phương thức của Decorator, nó cài đặt thêm các tính năng mới cho Component,
* Client: đối tượng sử dụng Component

3. Lợi ích của Decorator Pattern

- Tăng cường khả năng mở rộng của đối tượng, bởi bì những thay đổi được thực hiện bằng cách implement trên các lớp mới.

- Client sẽ không nhận thấy sự khác biệt khi bạn đưa cho nó một wrapper thay vì đối tượng gốc.

- Một đối tượng có thể được bao bọc bởi nhiều wrapper cùng một lúc

- Cho phép thêm hoặc xóa tính năng của một đối tượng lúc thực thi (runtime)

4. Sử dụng Decorator Pattern khi nào

- Khi muốn thêm tính năng mới cho các đối tượng mà không ảnh hưởng tới đối tượng này.

- Khi không thể mở rộng một đối tượng bằng cách kế thừa (inheritance). Chẳng hạn, một class sử dụng từ khóa final, muốn mở rộng class này chỉ còn cách sử dụng decorator.

- Trong một số trường hợp mà việc sử dụng kế thừa sẽ mất nhiều công sức trong việc viết code.

5. So sánh Decorator và Adapter

Giống nhau:

* Cả hai đều là structural pattern
* Cả hai đều sử dụng cách composition để cài đặt

Khác nhau:

* Decorator Pattern cho phép thêm một tính năng mới vào một object nhưng không được phép sử dụng kế thừa. Nó cho phép thay đổi lúc thực thi (runtime). Adapter được sử dụng khi bạn có một interface, và bạn muốn ánh xạ interface đó đến một đối tượng khác có vai trò chức năng tương tự.
* Decorator có xu hướng hoạt động trên một đối tượng. Adapter có xu hướng hoạt động trên nhiều đối tượng (có thể wrap nhiều interface).