

1. Iterator
2. Observer
3. Template
4. Strategy
5. Chain of responsibility
6. Decorator

**Soạn bài:**

Contents

[**Decorator Pattern** 2](#_Toc105138286)

[Chain of Responsibility Pattern 3](#_Toc105138287)

[Strategy Pattern 4](#_Toc105138288)

[Template Method/Pattern 5](#_Toc105138289)

[Observer Pattern 6](#_Toc105138290)

[Iterator Pattern 7](#_Toc105138291)

[Composite Pattern 8](#_Toc105138292)

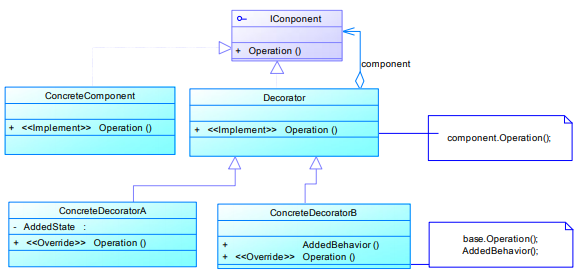
**Decorator Pattern**

Decorator Pattern được sử dụng để **thay đổi** một instance riêng lẻ của một class**, bằng cách tạo một class decorator bao bọc class gốc.**

Bằng cách này, việc thay đổi hoặc thêm chức năng của object decorator sẽ **không ảnh hưởng đến cấu trúc hoặc chức năng của object ban đầu**.

**Nó khác với kế thừa là nó được thực hiện trong thời gian** chạy và chỉ áp dụng cho một cá thể, trong khi kế thừa sẽ ảnh hưởng đến tất cả các trường hợp.

<https://freetuts.net/decorator-pattern-trong-java-2853.html>

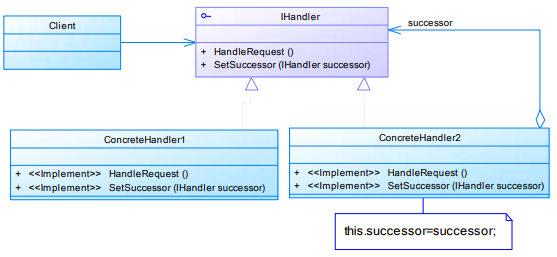


## Chain of Responsibility Pattern

Chain of Resppinsibility Pattern được sử dụng rất nhiều trong lập trình. Nó **thực hiện công việc định nghĩa** ra một chuỗi các object, chúng ta sẽ **gọi** **lần lượt từng object** để **xử lý yêu cầu**. Vì vậy, mỗi bộ xử lý trong chuỗi có **các logic xử lý riêng.**

Một điều quan trọng nữa là nó rất tiện dụng để tách sender ra khỏi receiver.

1. **Mục dích:** Tách rời người gởi và người thực hiện request.

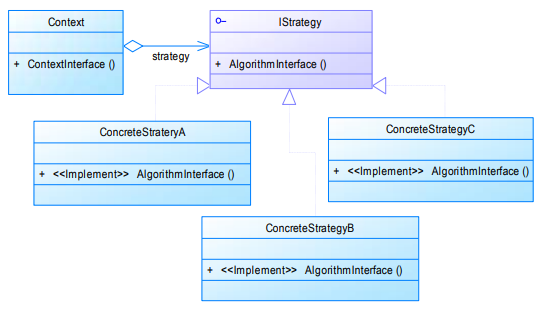


[**https://freetuts.net/chain-of-responsibility-pattern-trong-java-2879.html**](https://freetuts.net/chain-of-responsibility-pattern-trong-java-2879.html)

## Strategy Pattern

Strategy Pattern được sử dụng trong các tình huống các thuật toán hoặc hành vi của class là các **các hành vi động (có thể thay đổi).** Điều này có nghĩa là cả hành vi và **thuật toán đều có thể thay đổi trong thời gian chạy**, dựa vào đầu vào của client.

Tương tự như State Pattern, Strategy Pattern **sử dụng nhiều object xác định**, tùy thuộc vào các mục tiêu khác nhau.



[**https://freetuts.net/strategy-pattern-trong-java-2893.html**](https://freetuts.net/strategy-pattern-trong-java-2893.html)

## Template Method/Pattern

Template Method còn được gọi là Template Pattern **được sử dụng để xác định một class abstract (trừu tượng),** cung cấp các cách để chạy chạy phương thức của nó. Các class con **kế thừa các phương thức** này cũng phải tuân theo các định nghĩa bên trong nó.

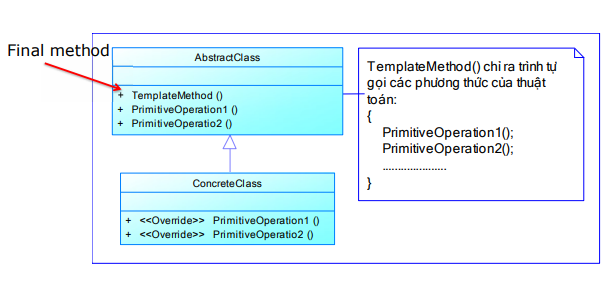
Trong một số trường hợp, class abstract có thể đã bao gồm một phương thức đã được triển khai trước đó. Và dĩ nhiên nó sẽ được chia sẽ đến tất cả các lớp con.

1. **Mục dích:** Định nghĩa khung sườn của một thuật toán bao gồm nhiều bước trong một phương thức.

+ Một số bước được khai báo abstract ở lớp cơ sở và ủy quyền cho lớp con cài đặt.

+ Việc cài đặt các bước ở lớp con không làm ảnh hưởng đến cấu trúc của thuật toán.

<https://freetuts.net/template-pattern-trong-java-2878.html>



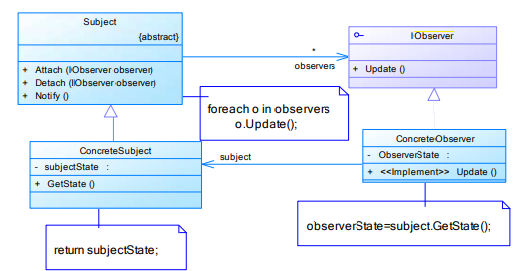
## Observer Pattern

Observer Pattern được **sử dụng để theo dõi trạng thái của một object nhất định**. Thường là trong một nhóm hoặc một mối quan hệ một - nhiều.

Trong những trường hợp như vậy, hầu hết thời gian và trạng thái thay đổi của một object sẽ ảnh hưởng đến trạng thái của phần còn lại. **Vì vậy, phải có một hệ thống ghi nhận lại những thay đổi, và thông báo cho các object khác.**

1. **Mục đích:** Định nghĩa một phụ thuộc **one-to-many** giữa các đối tượng sao cho khi một đối tượng thay đổi trạng thái, tất cả các đối tượng phụ thuộc nó **được thông báo và được cập nhật một cách tự động.**

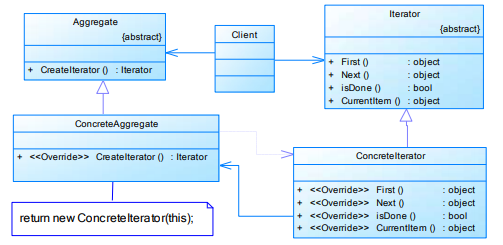
[**https://freetuts.net/observer-pattern-trong-java-2888.html**](https://freetuts.net/observer-pattern-trong-java-2888.html)



## Iterator Pattern

Iterator Pattern được sử dụng rất nhiều trong Java, và được xem như là mẫu cốt lõi của Java's Collection Framework.

**Pattern này có thể truy cập vào tất cả các phần tử của một collection object một cách tuần tự.**



**https://freetuts.net/iterator-pattern-trong-java-2883.html**

**MẪu phụ**

## Composite Pattern

Composite Pattern được sử dụng khi chúng ta cần xử lý một nhóm các object tương tự như cách xử lý một object.

Điều này thường được thực hiện bới class “owns” của object và cung cấp một tập hợp các phương thức để xử lý chúng như thể xử lý một object.

