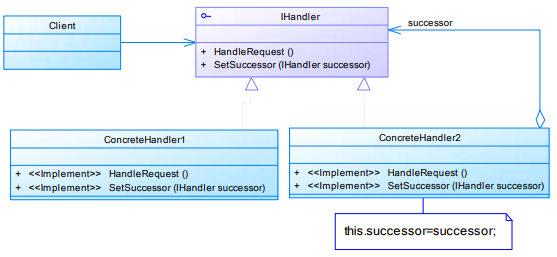
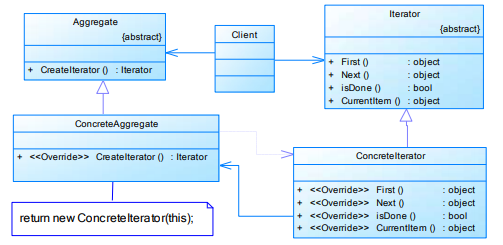
**BEHAVIORAL PATTERNS**

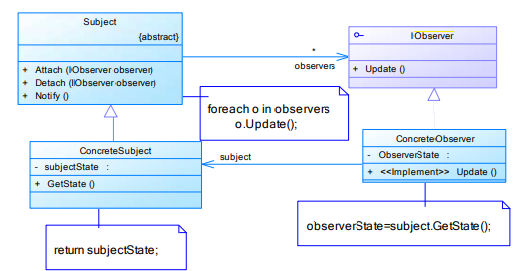
1. **Chain of reposibility**
2. **Mục dích:** Tách rời người gởi và người thực hiện request.
3. **Cấu trúc:**



1. **Iterator**
2. **Mục dích:** cung cấp một cách truy cập các phần tử của một tập hợp một cách tuần tự mà không cần biết cấu trúc của tập hợp đó.
3. **Cấu trúc:**



1. **Observer**
2. **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.
3. **Cấu trúc:**



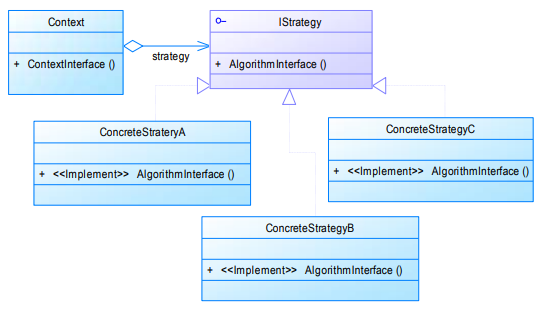
1. **Strategy**

**Ví dụ: có rất nhiều cách đi đến sân bay( máy bay, xe máy…)**

1. **Mục dích:**

Định nghĩa tập hợp các thuật toán, đóng gói từng thuật toán lại, và dễ dàng thay đổi linh hoạt các thuật toán bên trong object. Strategy cho phép thuật toán biến đổi độc lập khi người dùng sử dụng chúng.

1. **Cấu trúc:**



**Context là cái ngữ cảnh, cái phương pháp khi họ cần gì họ sẽ gọi**

**Iterface chứa các Phương thức**

**ConcreteStatery implement (trong này thì định nghĩa các phương thức như thế nào)**

**Main khi chạy thì chọn cách tình rồi đến phương thức và truyền vào**

1. **Template**

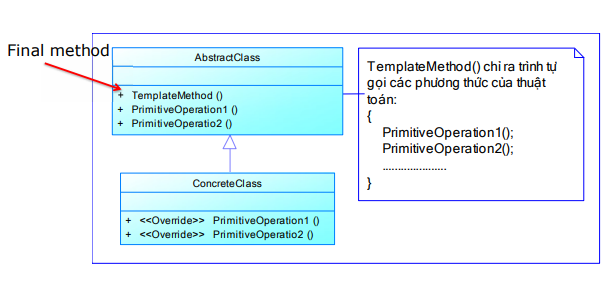
**Ví dụ**

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.

1. **Cấu trúc:**



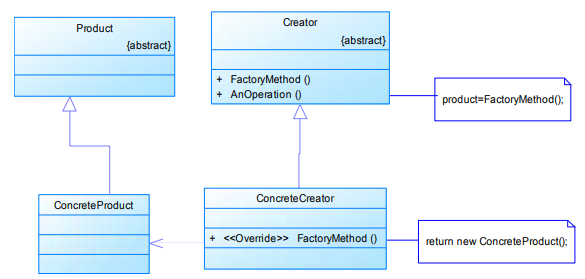
**So sánh strategy với template**

**St: nhiều cách giải quyết**

**T: 1 cách giải quyết**

**CREATIONAL PATTERNS**

1. **Factory Method**
2. **Mục đích:** định nghĩa một giao diện để tạo đối tượng, nhưng để cho các lớp con quyết định lớp nào sẽ được khởi tạo.
3. **Cấu trúc:**



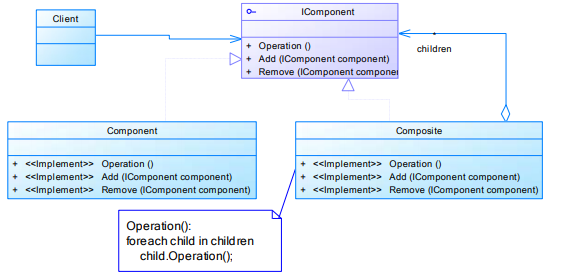
**STRUCTURAL PATTERNS**

1. **Composite**
2. **Mục dích:**

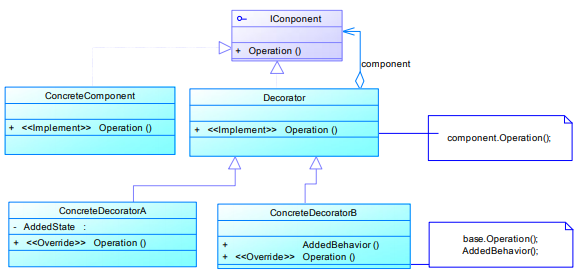
+ Sắp xếp các đối tượng vào các cấu trúc cây để biểu diễn các phân cấp part-whole giữa các đối tượng.

+ Đối xử với các đối tượng riêng lẻ và nhóm các đối tượng theo các cách giống nhau.

1. **Cấu trúc:**



1. **Decorator**
2. **Mục đích:** Cho phép thêm mới các trạng thái và hành vi vào một đối tượng lúc run-time bằng cách dùng kỹ thuật subclassing để mở rộng các chức năng của lớp.
3. **Cấu trúc:**



Theo mục đích :  
♣ Creational : Áp dụng trong quá trình khởi tạo, độc lập việc khai  
báo, khởi tạo một đối tượng mới.  
♣ Structural : Áp dụng trong việc tạo cấu trúc giữa các lớp hay đối  
tượng, mối liên quan giữa chúng.  
♣ Behavioral : Áp dụng trong việc độc lập các hành vi đối tượng,  
các thuật toán thực hiện hành vi đó.  
ϖ Theo phạm vi áp dụng :  
♣ Cho phạm vi đối tượng (objects).  
♣ Cho phạm vi lớp đối tượng (classes)