Tìm hiểu về Chain of Reponsibility

1. Chain of Responsibility Pattern là gì

Chain of Responsibility (COR) là một trong những Pattern thuộc nhóm hành vi (Behavior Pattern).

Toàn bộ các mẫu hành vi (Behavior Pattern) xoay quanh nguyên tắc “thành phần đối tượng (composition) hơn là thừa kế (inheritance)”. Nguyên tắc này nói rằng, thay vì mở rộng từ một lớp hiện có, thiết kế lớp của bạn để tham chiếu đến một lớp hiện có mà bạn muốn sử dụng. Trong Java, nó được thực hiện bằng cách khai báo một biến tham chiếu đối tượng của lớp hiện có. Sau đó, bằng cách khởi tạo đối tượng thông qua hàm tạo (constructor) hoặc phương thức (setter), và cuối cùng là tham chiếu

Chain of Responsibility cho phép một đối tượng gửi một yêu cầu nhưng không biết đối tượng nào sẽ nhận và xử lý nó. Điều này được thực hiện bằng cách kết nối các đối tượng nhận yêu cầu thành một chuỗi (chain) và gửi yêu cầu theo chuỗi đó đến khi có một đối tượng xử lý nó.

Chain of Responsibility hoạt động như một danh sách liên kết (Linked list) với việc đệ quy duyệt qua các phần tử (recursive traversal).

2. Cài đặt Chain of Responsibility như thế nào

Các thành phần tham gia mẫu Chain of Responsibility:

* Handler: định nghĩa 1 interface để xử lý các yêu cầu. Gán giá trị cho đối tượng successor (không bắt buộc).
* ConcreteHandler: xử lý yêu cầu. Có thể truy cập đối tượng successor (thuộc class Handler). Nếu đối tượng ConcreteHandler không thể xử lý yêu cầu, nó sẽ gửi yêu cầu cho successor của nó.
* Client: tạo ra các yêu cầu và yêu cầu đó sẽ được gửi đến các đối tượng tiếp nhận.

Client gửi một yêu cầu để xử lý gửi nó đến chuỗi (chain) các trình xử lý (handers), đó là các lớp mở rộng Handler. Mỗi Handler trong chuỗi lần lượt cố gắng xử lý yêu cầu nhận được từ Client. Nếu trình xử lý đầu tiên (ConcreteHandler) có thể xử lý nó, thì yêu cầu sẽ được xử lý. Nếu không được xử lý thì sẽ gửi đến trình xử lý tiếp theo trong chuỗi.

3. Lợi ích của Chain of Responsibility

* Giảm kết nối (loose coupling): Thay vì một đối tượng có khả năng xử lý yêu cầu chứa tham chiếu đến tất cả các đối tượng khác, nó chỉ cần một tham chiếu đến đối tượng tiếp theo. Tránh sự liên kết trực tiếp giữa đối tượng gửi yêu cầu (sender) và các đối tượng nhận yêu cầu (receivers).
* Tăng tính linh hoạt: đảm bảo Open/Closed Principle.
* Phân chia trách nhiệm cho các đối tượng: đảm bảo Single Responsibility Principle.
* Có khả năng thay đổi dây chuyền (chain) trong thời gian chạy
* Không đảm bảo có đối tượng xử lý yêu cầu.

4. Sử dụng Chain of Responsibility khi nào

* Có nhiều hơn một đối tượng có khả năng thực xử lý một yêu cầu trong khi đối tượng yêu cầu xử lý lại phụ thuộc vào ngữ cảnh sử dụng.
* Muốn gửi yêu cầu đến một trong số vài đối tượng nhưng không xác định đối tượng cụ thể nào sẽ xử lý yêu cầu đó
* Khi cần phải thực thi các trình xử lý theo một thứ tự nhất định
* Khi một tập hợp các đối tượng xử lý có thể thay đổi động: tập hợp các đối tượng có khả năng xử lý yêu cầu có thể không biết trướ, có thể thêm bớt hay thay đổi thứ tự sau này.