

Visitor Design Pattern

Mục lục

CHỦ ĐỀ THẢO LUẬN

- Visitor Design Pattern là gì?
- Thành phần của pattern
- Ứng dụng của pattern
- Ví dụ

Visitor Design Pattern là gì?

- Design pattern thuộc nhóm **behavioral**
- Cho phép định nghĩa các thao tác mới trên một class mà **không** làm thay đổi cấu trúc của đối tượng đó.
- Tách biệt thuật toán (hành vi) ra khỏi đối tượng (dữ liệu).
- Dựa trên kĩ thuật **Double Dispatch**. Thay vì đối tượng tự thực hiện hành vi, nó accept một visitor để visitor đó thực hiện hành vi trên dữ liệu của mình.

Thành phần của Visitor Design Pattern

► Gồm 5 thành phần chính:

- ◆ **Visitor (Interface)**: Khai báo visit cho từng loại phần tử cụ thể.
- ◆ **ConcreteVisitor (Class)**: Cài đặt chi tiết hành vi. Mỗi ConcreteVisitor đại diện cho một thao tác hoàn chỉnh.
- ◆ **Element (Interface)**: Công r vào cho Visitor.
- ◆ **ConcreteElement (Class)**: Các đối tượng thực tế (dữ liệu). Chúng cài đặt phương thức accept và gọi ngược lại phương thức visit tương ứng của Visitor.
- ◆ **Client**: Nơi chứa cấu trúc dữ liệu và thực hiện duyệt qua các phần tử để gọi accept.

Ứng dụng

► Nên dùng khi:

- ◆ Khi bạn có một cấu trúc đối tượng tương đối ổn định (ít khi thêm class mới), nhưng thường xuyên phải thêm các hành vi/ thuật toán mới xử lý chúng.
- ◆ Khi các hành vi mới không liên quan đến chức năng chính của đối tượng.

► Ưu điểm:

- ◆ Tuân thủ Open/Closed Principle: Dễ dàng thêm hành vi mới (tạo Visitor mới) mà không sửa code cũ.
- ◆ Gom nhóm logic: Các hành vi liên quan được gom vào một nhóm thay vì rải rác khắp nơi.

► Nhược điểm:

- ◆ Khó thêm Element mới: Nếu bạn thêm một loại đối tượng mới, bạn phải sửa tất cả các interface Visitor và các class ConcreteVisitor hiện có để thêm hàm visitTriangle.

Ví dụ