Visitor

David Suárez, Ignacio Tirado

1. Presentación del Visitor

El patrón Visitor proporciona una forma de agregar operaciones a objetos sin modificar las clases en las que se definen estos objetos. Esto se logra mediante la definición de una interfaz de Visitor que aclara los métodos de visita para cada clase de objeto posible. Las implementaciones de la interfaz de Visitor pueden realizar diferentes operaciones en dichos objetos.

2. **¿Qué es el Patrón Visitor?**

- Es un patrón de diseño que nos ayuda a separar operaciones de las clases de objetos.
- Nos permite agregar nuevas funcionalidades a un grupo de clases **sin tener que modificar esas clases**.

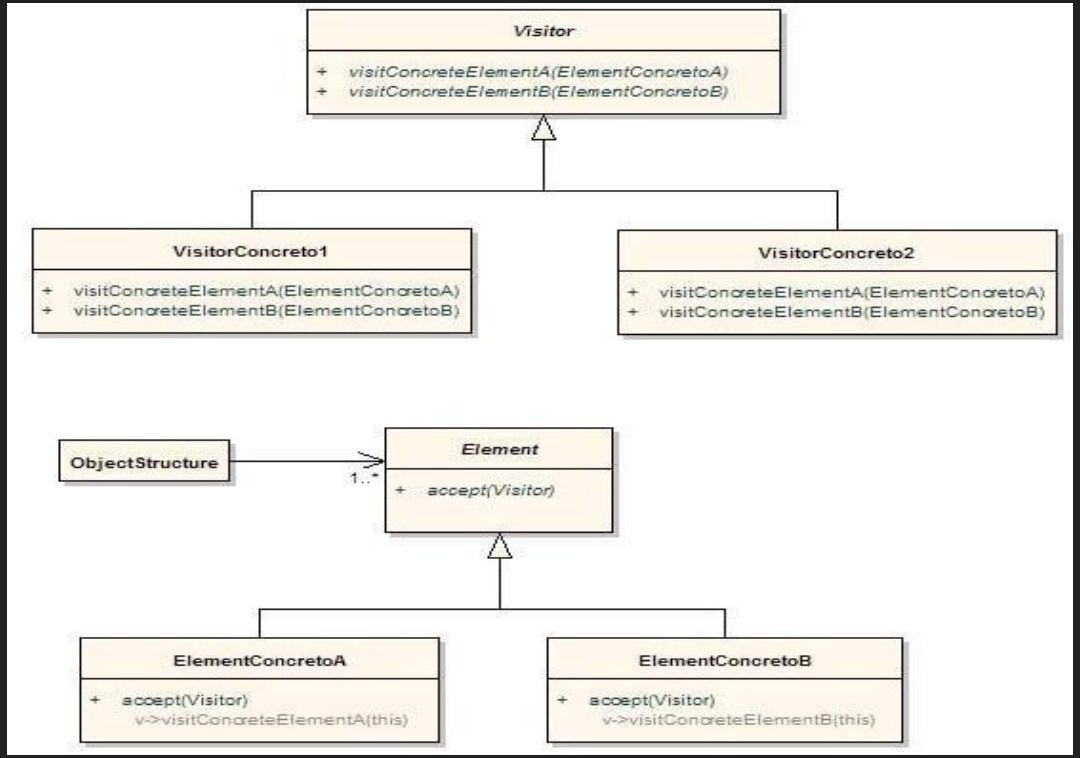
3. **¿Cuándo lo usamos?**

- Cuando tenemos una estructura de objetos con diferentes tipos (por ejemplo, libros, revistas, periódicos) y queremos aplicar varias operaciones a esos objetos.
- Si necesitamos que esas operaciones puedan crecer o cambiar sin afectar las clases originales.

4. ¿Cómo funciona?

1. **Visitante**: Creamos una clase `Visitor` que define las operaciones a realizar sobre cada tipo de objeto.
2. **Objetos**: Cada clase de objeto (como `Libro`, `Revista`) permite que el visitante “entre” y realice la operación necesaria.
3. **Ventaja**: Si necesitamos una nueva operación, simplemente la agregamos en el `Visitor`, sin cambiar las clases `Libro` o `Revista`.

UML genérico



4.1) Ejemplo

Imaginemos que tenemos figuras geométricas como círculos, cuadrados y triángulos. En lugar de poner todos los métodos para calcular áreas y dibujarlas en cada figura, usamos el patrón Visitor.

Creamos visitantes que realizan estas tareas: un visitante para calcular el área de cada figura y otro para imprimir cómo se dibujan. Esto mantiene el código de las figuras limpio y permite agregar nuevas operaciones fácilmente en el futuro, sin modificar las clases de las figuras. Así, el sistema se mantiene organizado y flexible.

UML del código de figuras

5. Beneficios del Patrón Visitor

- **Flexibilidad**: Agrega nuevas operaciones sin tocar la estructura de los objetos.
- **Organización**: Las clases originales no se llenan de funciones que no son esenciales para su propósito.
- **Reutilización**: El `Visitor` puede aplicarse a distintos objetos en diferentes contextos.

6) Conclusión

El Patrón Visitor es ideal cuando necesitamos realizar múltiples operaciones sobre objetos de distintas clases, pero queremos mantener cada clase limpia y enfocada en su propósito principal. Una gran solución para un código más modular y fácil de mantener.