VERISOFT

SDET Course

Design Patterns - Visitor

3 Types of Design Patterns

- Creational
  - Singleton
  - Builder
  - Prototype
  - Factory Method
  - Abstract Factory
- Structural
  - Adapter
  - Composite
  - Proxy
  - Flyweight
  - Bridge
  - Facade
  - Decorator
- Behavioral
  - Strategy
  - Observer
  - Command
  - Memento
  - State
  - Template Method
  - Mediator
  - Chain of Responsibility
  - Interpreter
  - **Visitor**
  - Iterator

Agenda

- Description
- Diagram
- Code sample (Java)
- Use cases

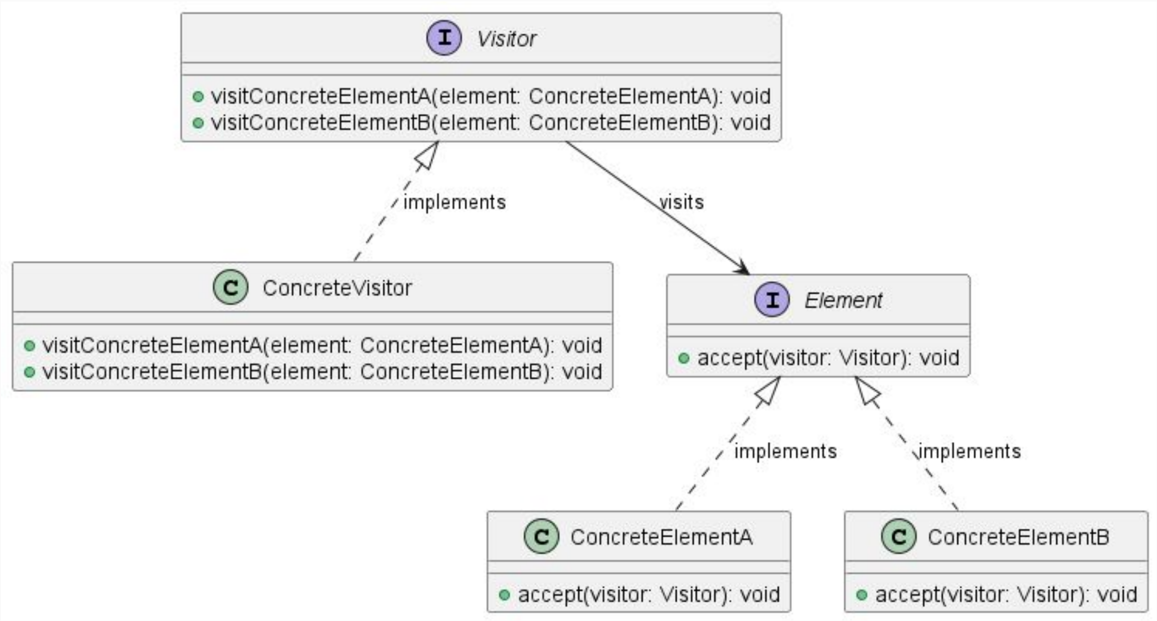
Description

Introduction - The Visitor

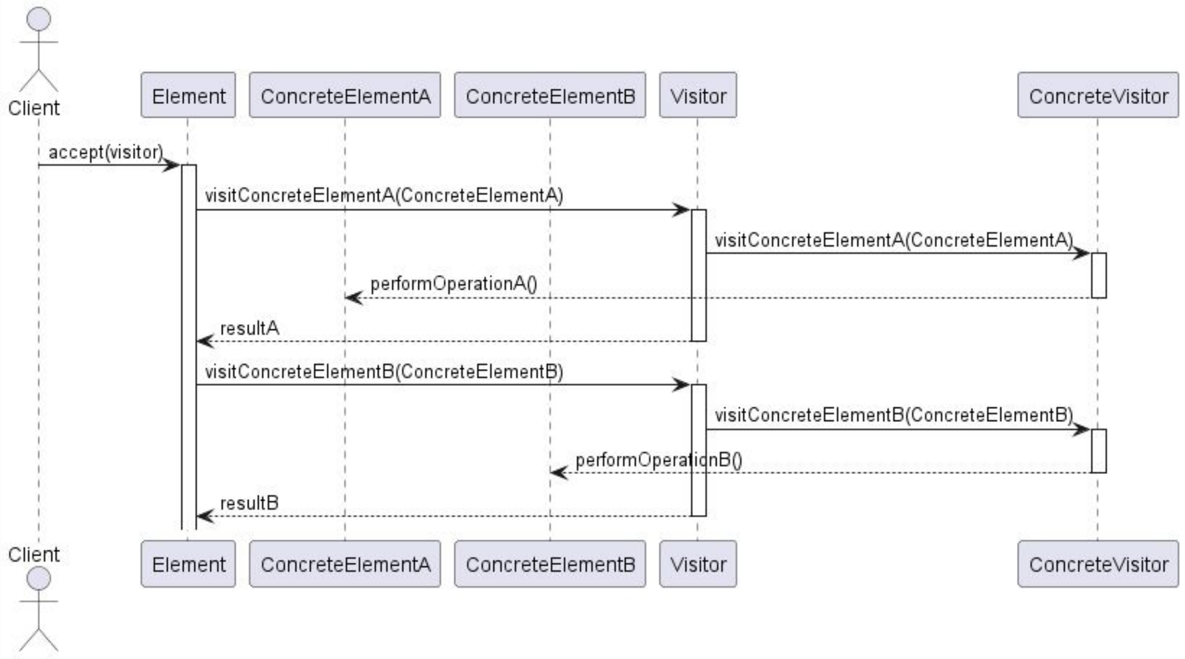
Description

The Visitor design pattern is a behavioral pattern in software engineering that facilitates adding new operations or functionalities to a collection of objects without altering their structure. It separates the algorithm from the object structure upon which it operates. The pattern involves defining a visitor interface with methods corresponding to each class in the object structure. Each class then accepts visitors, allowing them to perform operations specific to their type. This pattern is particularly useful when the object structure is complex and changes infrequently, but new operations need to be added frequently. It promotes extensibility and maintainability by encapsulating operations in separate visitor classes, enhancing the flexibility of the system.

Class Diagram


I Visitor
visitConcreteElementA(element: ConcreteElementA): void
visitConcreteElementB(element: ConcreteElementB): void
implements
visits
C ConcreteVisitor
visitConcreteElementA(element: ConcreteElementA): void
visitConcreteElementB(element: ConcreteElementB): void
I Element
accept(visitor: Visitor): void
implements
implements
C ConcreteElementA
accept(visitor: Visitor): void
C ConcreteElementB
accept(visitor: Visitor): void

Sequence Diagram



Code Sample

Use cases

- General
  - Document Processing (XML, JSON): Validation, Transformation, Extraction
  - Data Structure Operations: Trees / Graphs data design, perform search, manipulation etc.
  - Database Query Processing: Optimization / Execution strategy

- In Test Automation
  - Optimize SQL
  - Build Smart Page Objects

Happy Coding