SDET Course

Design Patterns - Decorator

3 Types of Design Patterns

VERISOFT

- 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

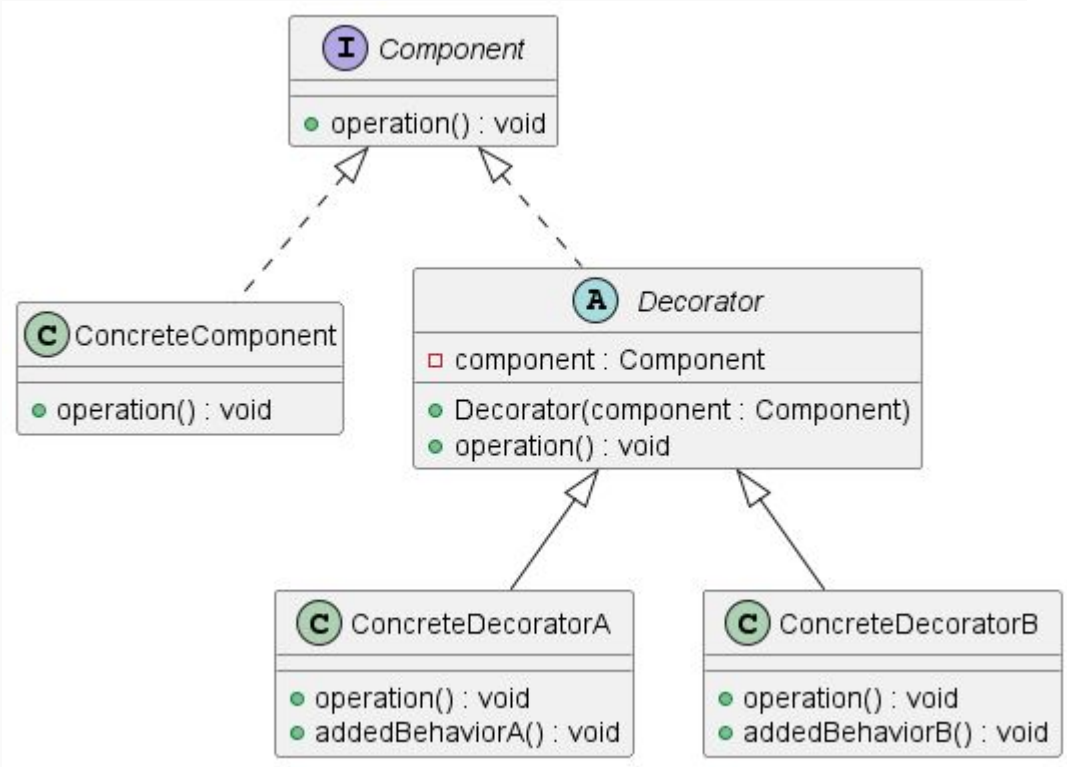
VERISOFT

Descri
on

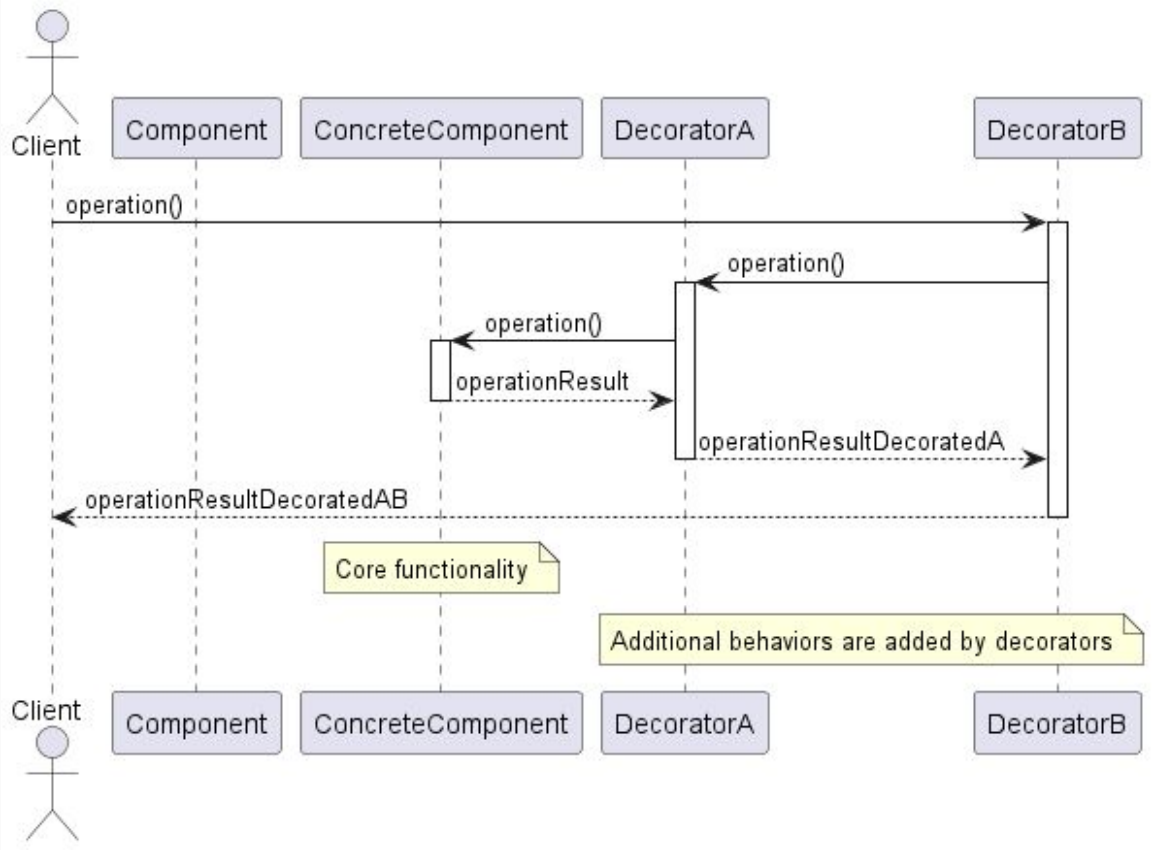
Description

The Decorator design pattern is a structural pattern used to extend the functionality of objects in a flexible and dynamic manner without modifying their structure. It allows for the addition of responsibilities to objects on the fly, by wrapping them with new decorator classes that contain the additional functionality. This pattern is particularly useful for adhering to the Open/Closed Principle, one of the SOLID principles of object-oriented design, which states that software entities (classes, modules, functions, etc.) should be open for extension but closed for modification. Decorators provide a flexible alternative to subclassing for extending functionality, by composing objects with specific behaviors or functionalities at runtime, rather than inheriting them statically at compile time. This results in a more modular, scalable, and maintainable codebase, where changes can be made without affecting the existing code.

Class Diagram



Sequence Diagram



Code Sample

Use cases

- General
  - Permission and Role-based Access Control
  - Logging and Monitoring
  - Dynamic Feature Toggle

- In Test Automation
  - Dynamic Test Configuration
  - Selective Test Retry Logic
  - Conditional Test Execution

Happy Coding