SWIFT
Decorator

Bill Kim(김정훈) | ibillkim@gmail.com

목차

Decorator

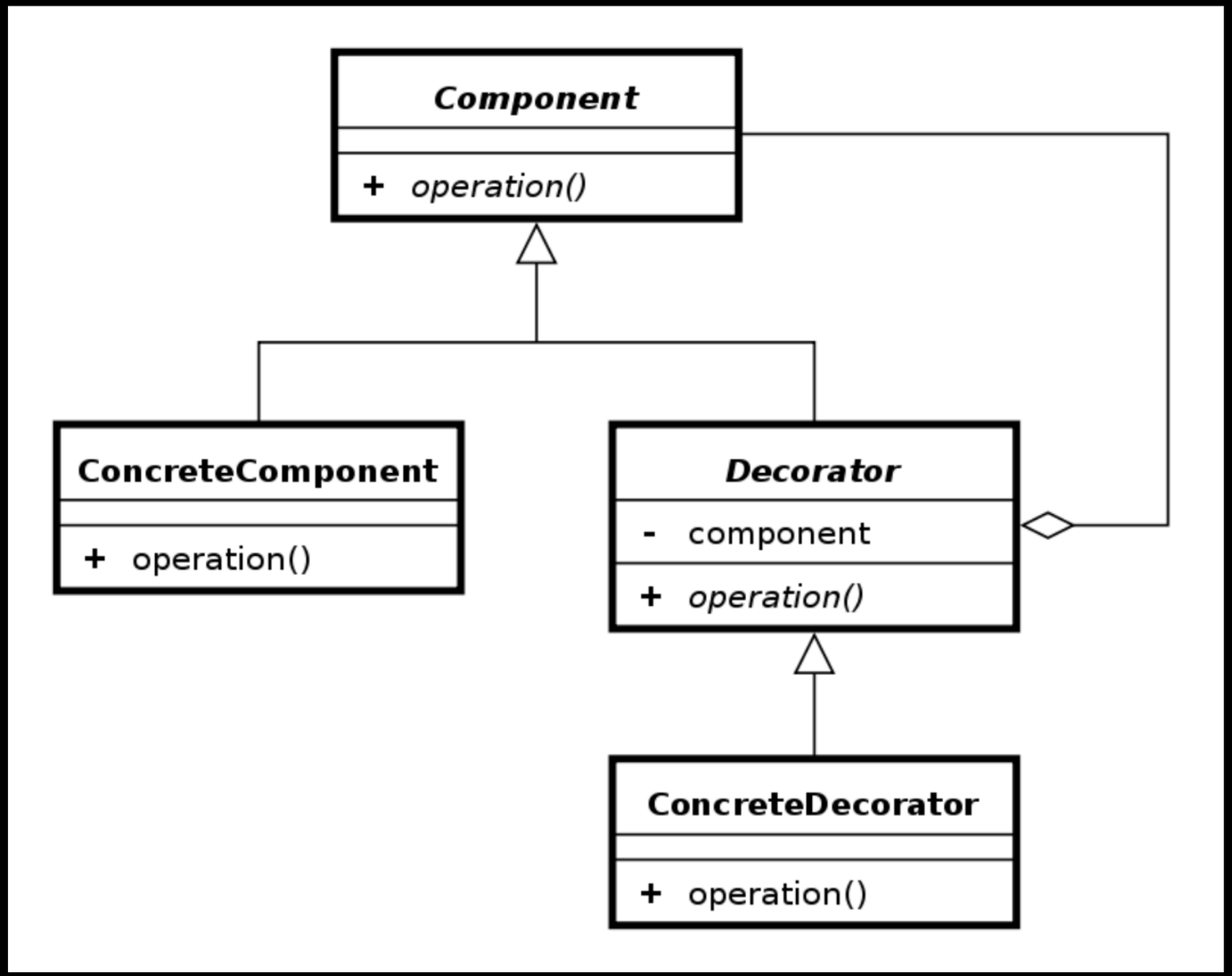
Decorator(데코레이터) 디자인 패턴은 특정 객체에 대해서 새로운 기능을 동적(Run-time)으로 추가하기 위한 구조 설계 패턴입니다.

객체에 동적으로 기능을 추가하고 코드의 추가를 숨기고 싶은 경우 사용하면 좋습니다.

또한 상속을 통해서 객체를 확장할 수 없는 경우에서도 사용할 수 있습니다.

Decorator

Decorator 패턴을 UML로 도식화하면 아래와 같습니다.



Structure

Component : 기능을 동적으로 확장하기위해 기본이 되는 추상 클래스 객체

ConcreteComponent : 컴포넌트(Component) 객체의 인터페이스를 구체화시키는 객체

Decorator : Component 객체를 소유하며 Component의 인터페이스를 실행하는 객체

ConcreteDecorator : Decorator 가 가지고 있는 Component 객체에 대한 인터페이스의 구체화를 실현하는 객체

Implementation

구체적인 구현에 대해서 소스 코드를 통하여 살펴봅니다.

```
protocol Component {
    func operation() -> String
}

class ConcreteComponent: Component {
    func operation() -> String {
        return "ConcreteComponent operation"
    }
}

class Decorator : Component {
    private var component: Component

    init(_ component: Component) {
        self.component = component
    }

    func operation() -> String {
        return component.operation()
    }
}
```

Implementation

```
class ConcreteDecoratorA : Decorator {
    override func operation() -> String {
        return "ConcreteDecoratorA(" + super.operation() + ")"
    }
}

class ConcreteDecoratorB : Decorator {
    override func operation() -> String {
        return "ConcreteDecoratorB(" + super.operation() + ")"
    }
}

let component = ConcreteComponent()
print("Result: " + component.operation())
// Result: ConcreteComponent operation

let decorator1 = ConcreteDecoratorA(component)
print("Result: " + decorator1.operation())
// Result: ConcreteDecoratorA(ConcreteComponent operation)

let decorator2 = ConcreteDecoratorB(decorator1)
print("Result: " + decorator2.operation())
// Result: ConcreteDecoratorB(ConcreteDecoratorA(ConcreteComponent operation))
```

References

[1] [Swift-Design Pattern] 데코레이터 패턴(Decorator pattern) : http://throughkim.kr/2019/09/09/swift-decorator/

[2] Decorator in Swift : https://refactoring.guru/design-patterns/decorator/swift/example#lang-features

[3] [DesignPattern]데코레이터 패턴(Decorator Pattern) : http://minsone.github.io/programming/designpattern-decorator

[4] Top 5 스위프트 디자인 패턴 (번역) : https://leejigun.github.io/Top_5_Design_Patterns

[5] Design Patterns in Swift: Decorator Pattern : https://medium.com/design-patterns-in-swift/design-patterns-in-swift-decorator-pattern-2026e7112869

References

[6] Decorator pattern in Swift : https://medium.com/jeremy-codes/decorator-pattern-in-swift-e5fa11ea3c3f

[7] Swift Solutions: Decorator Pattern : https://swiftcraft.io/decorator-pattern-swift/

[8] [Design Pattern] 데코레이터(Decorator) 패턴 - 디자인 패턴 : https://palpit.tistory.com/193

[9] 데코레이터 패턴(Decorator Pattern) : https://unabated.tistory.com/entry/데코레이터-패턴Decorator-Pattern

[10] Design Patterns on iOS using Swift - Part 1/2 : https://www.raywenderlich.com/477-design-patterns-on-ios-using-swift-part-1-2

Thank you!