SWIFT
Strategy

Bill Kim(김정훈) | ibillkim@gmail.com

목차

Strategy

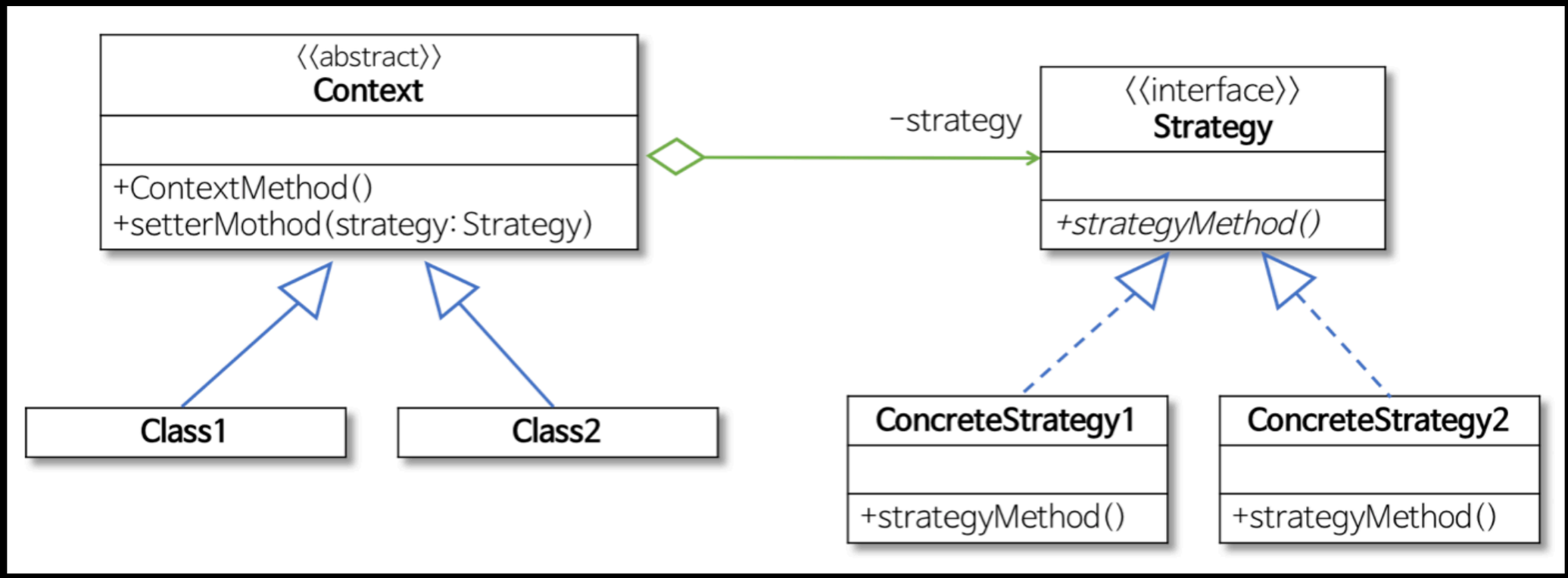
Strategy 패턴은 클래스의 행위를 캡슐화 하여 동적으로 행위를 자유롭게 바꿀 수 있도록 돕는 패턴이다.

Strategy 패턴은 특정한 계열의 알고리즘(행위) 및 동작을 정의하고 해당 알고리즘을 캡슐화하며 이 알고리즘들을 해당 계열 안에서 상호 교체가 가능하게 만들어 줍니다.

즉 Strategy 패턴은 유연하고 재사용 가능한 객체 지향 소프트웨어를 설계하기 위해 반복되는 디자인 문제를 해결하는 방법으로서 객체는 구현, 변경, 테스트, 재사용이 쉬워야 한다는 것을 잘 반영한 행위(Behavioral) 관련 디자인 패턴 중의 하나입니다.

Structure

Strategy 패턴을 UML로 도식화하면 아래와 같습니다.



Structure

Context : Strategy 객체를 소유하고 동적으로 행위를 실행하는 객체

Strategy : ConcreteStrategy 객체의 부모 객체로서 동적으로 지정할 행위에 대한 알고리즘 인터페이스를 정의하는 추상 객체

ConcreteStrategy : 주요 알고리즘 인터페이스 함수에 대한 구현을 담당하는 객체

Implementation

구체적인 구현에 대해서 소스 코드를 통하여 살펴봅니다.

```
class Context {
    private var strategy: Strategy

    init(strategy: Strategy) {
        self.strategy = strategy
    }

    func update(strategy: Strategy) {
        self.strategy = strategy
    }

    func doSomeBusinessLogic() {
        let result = strategy.doAlgorithm(["a", "b", "c", "d", "e"])
        print(result.joined(separator: ","))
    }
}
```

Implementation

```
protocol Strategy {
    func doAlgorithm<T: Comparable>(_ data: [T]) -> [T]
}

class ConcreteStrategyA : Strategy {
    func doAlgorithm<T: Comparable>(_ data: [T]) -> [T] {
        return data.sorted()
    }
}
```

```
class ConcreteStrategyB : Strategy {
    func doAlgorithm<T: Comparable>(_ data: [T]) -> [T] {
        return data.sorted(by: >)
    }
}
```

Implementation

```
let context = Context(strategy: ConcreteStrategyA())
print("Client: Strategy is set to normal sorting.")
context.doSomeBusinessLogic() // a,b,c,d,e

print("\nClient: Strategy is set to reverse sorting.")
context.update(strategy: ConcreteStrategyB())
context.doSomeBusinessLogic() // e,d,c,b,a
```

References

[1] [Swift-Design Pattern] 스트래티지 패턴 (Strategy pattern) : http://throughkim.kr/2019/09/04/swift-strategy/

[2] Strategy pattern in Swift : https://medium.com/flawless-app-stories/strategy-pattern-in-swift-1462dbddd9fe

[3] Strategy Pattern (with iOS, Swift) : https://rhammer.tistory.com/347

[4] Design Pattern - Strategy : https://ehdrjsdlzzzz.github.io/2019/01/18/Design-Pattern-Strategy/

[5] Strategy in Swift : https://refactoring.guru/design-patterns/strategy/swift/example

References

[6] Design Patterns with Swift: Quick look at a Strategy Pattern : https://brightinventions.pl/blog/quick-look-on-a-strategy-pattern-using-swift/

[7] Design Pattern In Use for iOS Developers: Strategy Pattern : https://viblo.asia/p/design-pattern-in-use-for-ios-developers-strategy-pattern-eW65GbWjlDO

[8] The Strategy Pattern in iOS Apps : https://blog.usejournal.com/the-strategy-pattern-in-ios-apps-346abc9e86a6

[9] The Strategy Pattern : http://www.thomashanning.com/the-strategy-pattern/

[10] 전략 패턴 : https://ko.wikipedia.org/wiki/전략_패턴

Thank you!