SWIFT
Template Method

Bill Kim(김정훈) | ibillkim@gmail.com

목차

Template Method

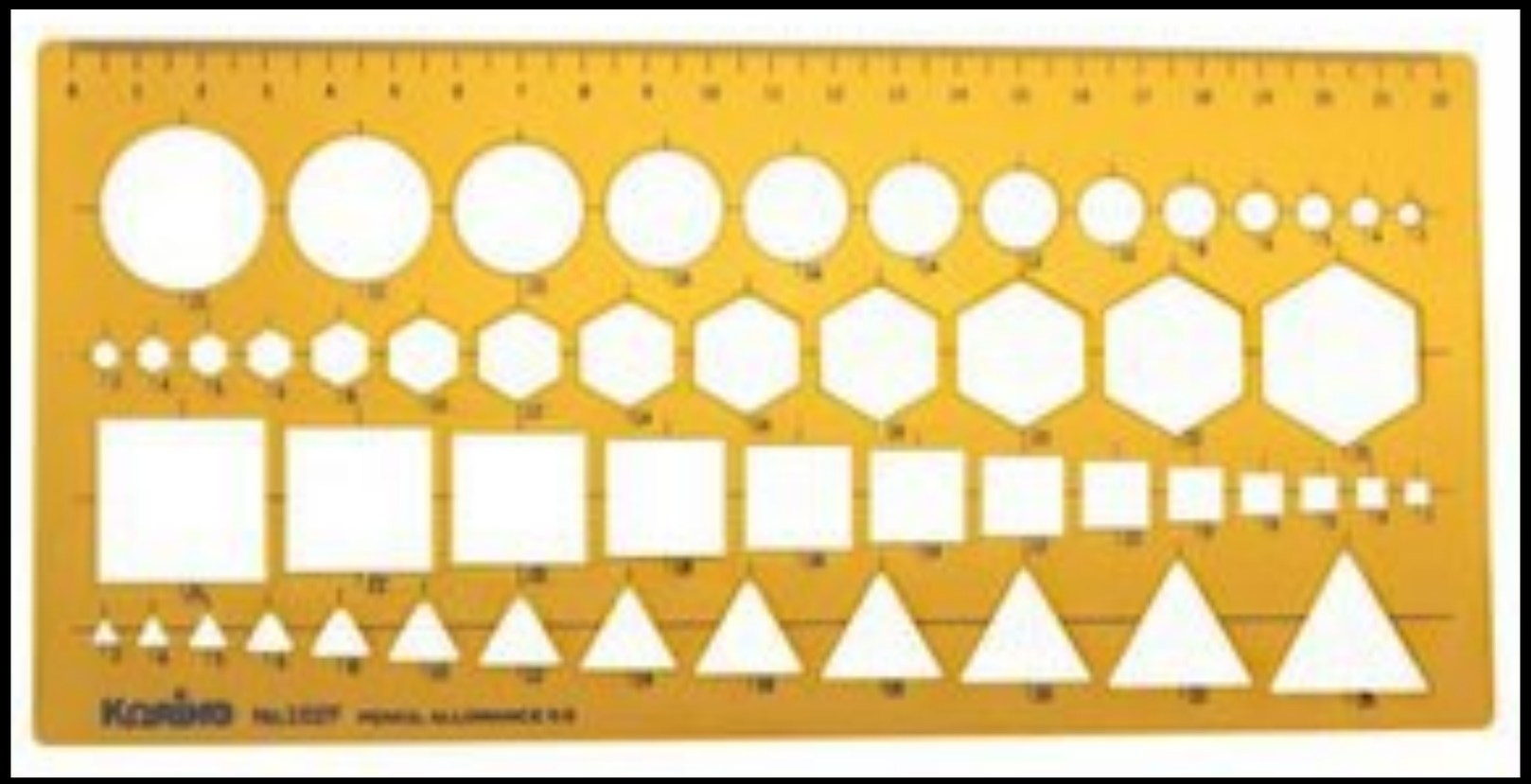
Template Method 패턴은 어떤 작업을 처리하는 일부분을 서브 클래스로 캡슐화해 전체 일을 수행하는 구조는 바꾸지 않으면서 특정 단계에서 수행하는 내역을 바꾸는 행위 관련 패턴입니다.

좀더 쉽게 설명해보면 상위 클래스에서 템플릿과 같은 틀을 제공하고 하위 클래스에서 이 틀을 이용하여 작업을 수행하도록 합니다.

결국 상위 클래스에서는 뼈대를 구성하고 하위 클래스에서 구체적인 내용을 구현하도록 합니다.

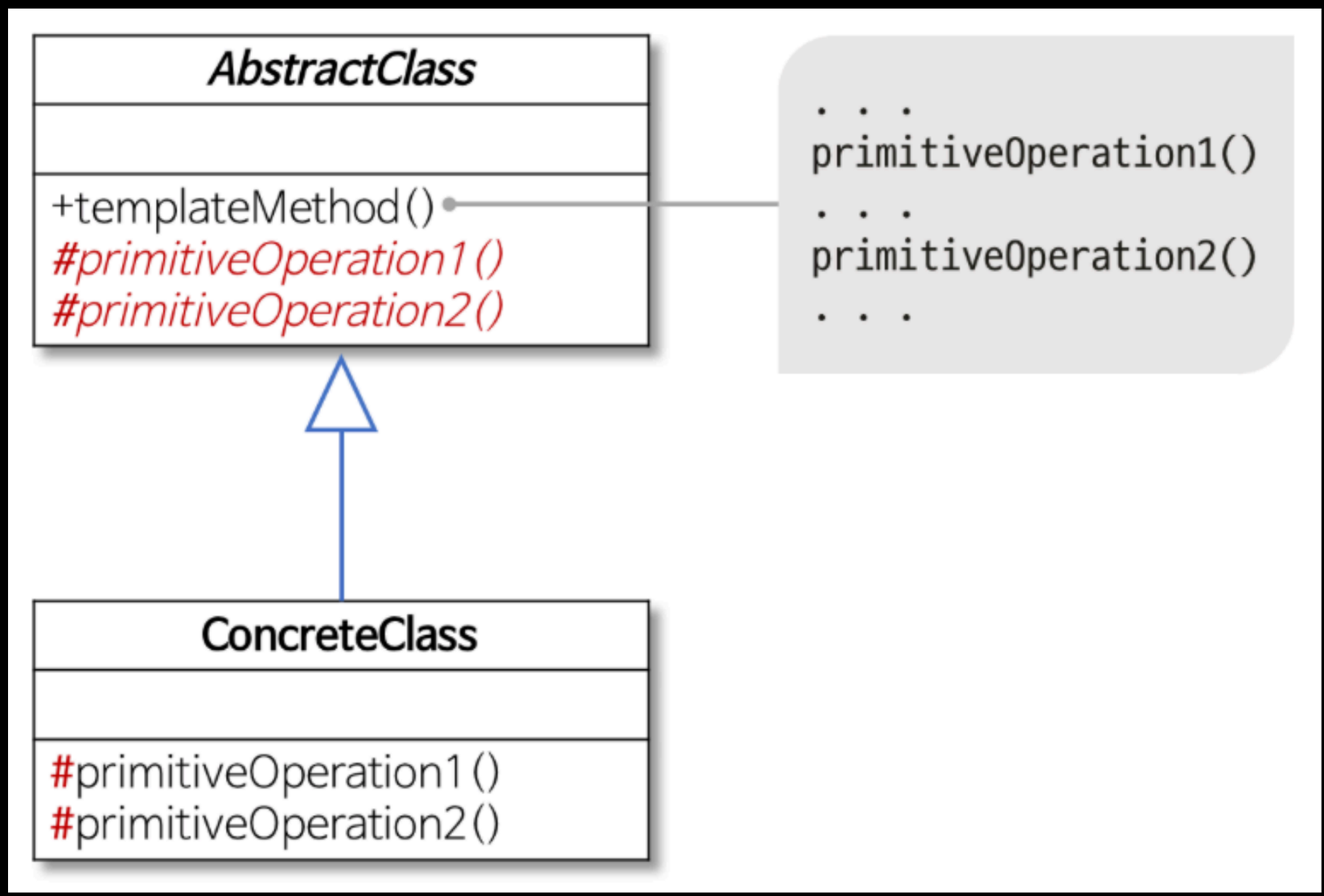
Template Method

Template Method 패턴을 특징을 한번에 보여주는 예시입니다.



Structure

Template Method 패턴을 UML로 도식화하면 아래와 같습니다.



Structure

AbstractClass : 여러 기본 뼈대가 되는 함수들을 정의하고 공통의 템플릿 메소드(함수)를 정의하는 클래스 객체

ConcreteClass : AbstractClass 객체를 상속받아 기본 뼈대 함수에 대한 실제적인 구현을 하며 템플랫 메소드를 호출하는 객체

Implementation

구체적인 구현에 대해서 소스 코드를 통하여 살펴봅니다.

```
protocol AbstractClass {
    func templateMethod() // 기본 템플릿 메소드

    // 기본 골격이 되는 메소드들

    func baseOperation1()
    func baseOperation2()
    func baseOperation3()

    func requiredOperations1()
    func requiredOperation2()

    func hook1()
    func hook2()
}
```

Implementation

```
extension AbstractClass {
    func templateMethod() {
        baseOperation1()
        requiredOperations1()
        baseOperation2()
        hook1()
        requiredOperation2()
        baseOperation3()
        hook2()
    }

    func baseOperation1() {
        print("baseOperation1")
    }

    func baseOperation2() {
        print("baseOperation2")
    }

    func baseOperation3() {
        print("baseOperation3")
    }

    func hook1() {}
    func hook2() {}
}
```

Implementation

```
class ConcreteClass1: AbstractClass {
    func requiredOperations1() {
        print("ConcreteClass1 requiredOperations1")
    }

    func requiredOperation2() {
        print("ConcreteClass1 requiredOperation2")
    }

    func hook2() {
        print("ConcreteClass1 hook2")
    }
}

class ConcreteClass2: AbstractClass {
    func requiredOperations1() {
        print("ConcreteClass2 requiredOperations1")
    }

    func requiredOperation2() {
        print("ConcreteClass2 requiredOperation2")
    }

    func hook1() {
        print("ConcreteClass2 hook1")
    }
}
```

Implementation

```
let template1 = ConcreteClass1()
template1.templateMethod()

// baseOperation1
// ConcreteClass1 requiredOperations1
// baseOperation2
// ConcreteClass1 requiredOperation2
// baseOperation3
// ConcreteClass1 hook2
```

```
let template2 = ConcreteClass2()
template2.templateMethod()

// baseOperation1
// ConcreteClass2 requiredOperations1
// baseOperation2
// ConcreteClass2 hook1
// ConcreteClass2 requiredOperation2
// baseOperation3
```

References

[1] 템플릿 메소드 패턴(Template Method Pattern) : https://jdm.kr/blog/116

[2] Template Method in Swift : https://refactoring.guru/design-patterns/template-method/swift/example

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

[4] 템플릿 메서드 패턴이란 : https://gmlwjd9405.github.io/2018/07/13/template-method-pattern.html

[5] Template Method in Swift : https://www.sm-cloud.com/template-method/

References

[6] #4 Template Method 패턴[Java 디자인 패턴] : https://puzzleleaf.tistory.com/114

[7] Design Patterns in Swift: Part II— Behavioral Design Pattern : https://medium.com/@lubabahasnain93/design-patterns-in-swift-part-ii-behavioral-design-pattern-5e4e5f13ccf7

[8] Template Method Pattern (템플릿 메소드 패턴, hook 메소드, 예제) : https://sjh836.tistory.com/140

[9] Template Design Pattern in Swift : https://agilesolutionspace.blogspot.com/2017/04/template-design-pattern-in-swift.html

[10] Design patterns in Swift Ch4: Iterator & Template Method : https://www.slideshare.net/ChihyangLi/design-patterns-in-swift-ch4-iterator-template-method

Thank you!