# SWIFT
# Facade

Bill Kim(김정훈) | ibillkim@gmail.com

# 목차

# Facade

Facade(퍼사드) 디자인 패턴은 복잡한 클래스 시스템에 대해서 간단한 인터페이스를 제공해주는 구조 설계 관련 디자인 패턴입니다.

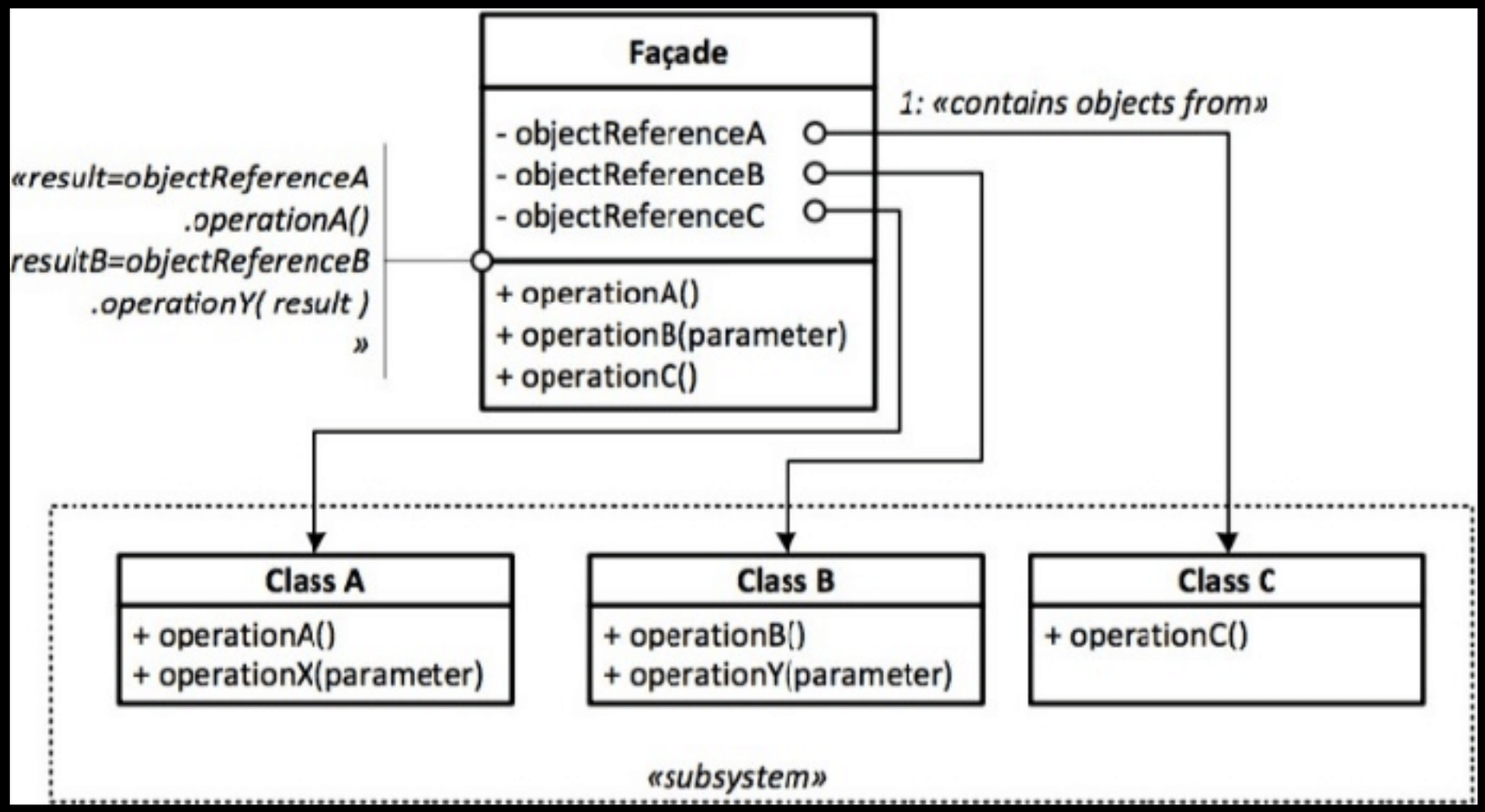
Facade 패턴은 많은 클래스를 포함하는 복잡한 서브 시스템을 사용하기 위한 간단한 인터페이스를 제공합니다.

Facade 패턴은 복잡한 하위 시스템을 직접 사용하여 확장할 수 있는 기능이 제한된 단순화된 인터페이스를 제공합니다.

이 단순화된 인터페이스는 클라이언트가 필요로 하는 기능만 제공하면서 다른 모든 기능은 숨깁니다.

# Facade

Facade(퍼사드) 패턴을 UML로 도식화하면 아래와 같습니다.



# Structure

Subsystem : Facade 객체가 사용하게될 서브 시스템 객체이다.

Facade : Subsystem들을 소유하며 관리 및 사용하는 객체, 클라이언트에게 통합된 외부 인터페이스 함수를 제공한다.

Client : Facade 객체를 통해서 여러 복잡한 서브 시스템의 다양한 인터페이스를 간단한 인터페이스로 사용하는 객체이다.

# Implementation

구체적인 구현에 대해서 소스 코드를 통하여 살펴봅니다.

```
class Subsystem1 {
    func operation1() -> String {
        return "Sybsystem1: Ready!\n"
    }

    func operationN() -> String {
        return "Sybsystem1: Go!\n"
    }
}

class Subsystem2 {
    func operation1() -> String {
        return "Sybsystem2: Get ready!\n"
    }

    func operationZ() -> String {
        return "Sybsystem2: Fire!\n"
    }
}
```

# Implementation

```
class Facade {
    private var subsystem1: Subsystem1
    private var subsystem2: Subsystem2

    init(subsystem1: Subsystem1 = Subsystem1(),
         subsystem2: Subsystem2 = Subsystem2()) {
        self.subsystem1 = subsystem1
        self.subsystem2 = subsystem2
    }

    func operation() -> String {

        var result = "Facade initializes subsystems:\n"
        result += " " + subsystem1.operation1()
        result += " " + subsystem2.operation1()
        result += "\n" + "Facade orders subsystems to perform the action:\n"
        result += " " + subsystem1.operationN()
        result += " " + subsystem2.operationZ()

        return result
    }
}
```

# Implementation

```
let facade = Facade(subsystem1: Subsystem1(), subsystem2: Subsystem2())
print(facade.operation())

// Facade initializes subsystems:
// Sybsystem1: Ready!
// Sybsystem2: Get ready!

// Facade orders subsystems to perform the action:
// Sybsystem1: Go!
// Sybsystem2: Fire!
```

# References


[1] [Swift] 퍼사드(Facade) 패턴 : https://m.blog.naver.com/PostView.nhn?blogId=horajjan&logNo=220804536746&proxyReferer=https:%2F%2Fwww.google.com%2F

[2] Facade in Swift : https://refactoring.guru/design-patterns/facade/swift/example

[3] Swift facade design pattern : https://theswiftdev.com/swift-facade-design-pattern/

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

[5] Swift World: Design Patterns — Facade : https://medium.com/swiftworld/swift-world-design-patterns-facade-579ef4b3319f

# References


[6] Implement the Facade Design Pattern in Swift 5 : https://medium.com/better-programming/implement-the-facade-design-pattern-in-swift-dcc4325754ff

[7] Design Patterns in Swift #3: Facade and Adapter : https://www.appcoda.com/design-pattern-structural/

[8] [디자인 패턴] 07. 어댑터 패턴과 퍼사드 패턴 (Adapter Pattern and Facade Pattern) : https://itchipmunk.tistory.com/357

[9] The Facade Design Pattern in Swift with Code Examples : https://www.iosapptemplates.com/blog/ios-development/facade-design-patterns-swift

[10] [Design Pattern] 파써드(Facade) 패턴 : https://palpit.tistory.com/192

Thank you!