SWIFT
Flyweight

Bill Kim(김정훈) | ibillkim@gmail.com

목차

Flyweight

Flyweight 패턴은 객체의 내부에서 참조하는 객체를 직접 만드는 것이 아니라, 없다면 만들고 만들어져 있다면 객체를 공유하여 전달해주는 구조 관련 패턴입니다.

이렇게 하기 위해 대부분 팩토리 메소드 패턴을 사용해 객체를 생성한다.

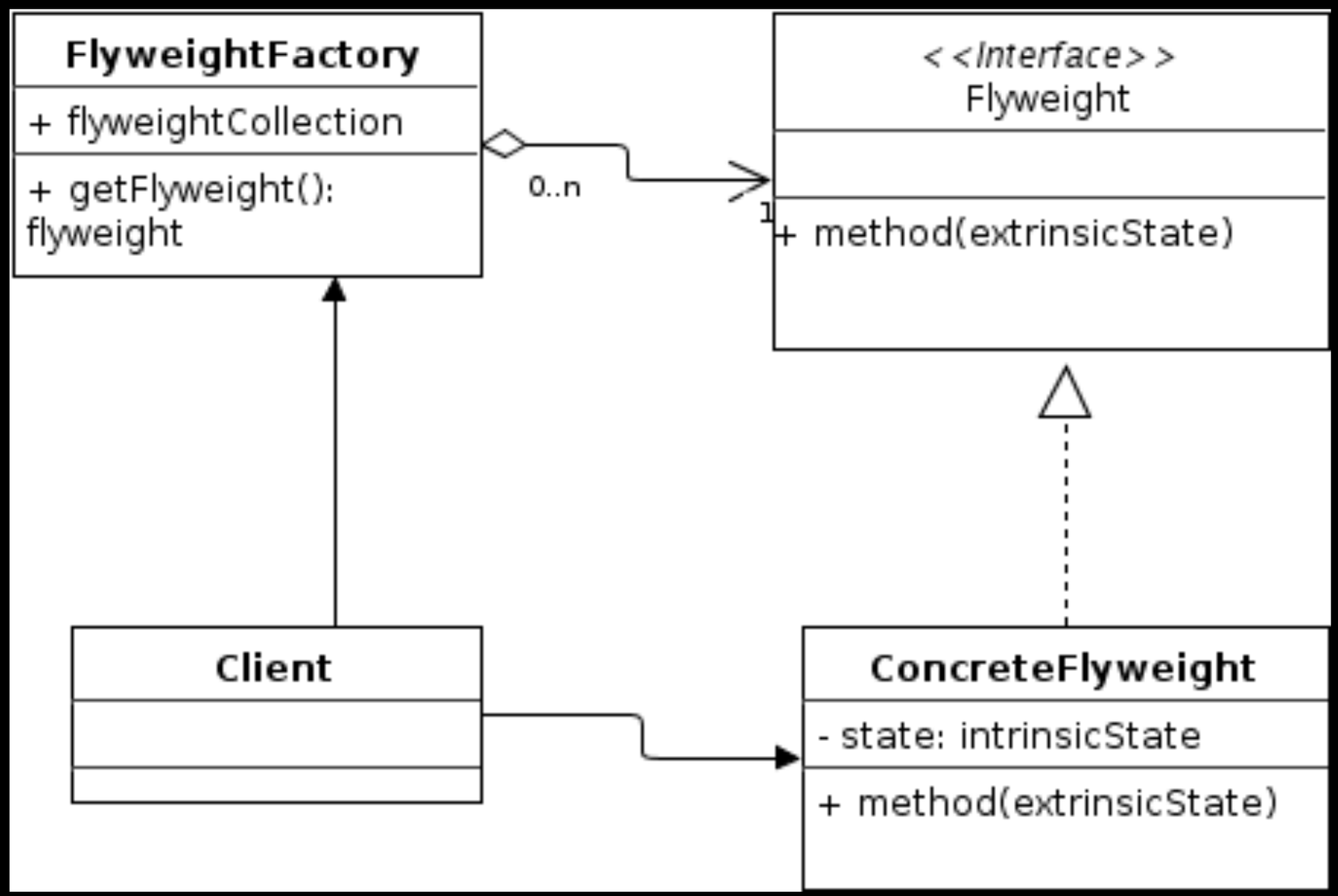
팩토리 메소드 안에서는 객체(Flyweight 객체)를 새로 생성합니다.

Flyweight 패턴을 사용하면 객체의 할당에 사용되는 메모리를 줄일 수 있을 뿐 아니라, 객체를 생성하는 시간도 들지 않게 도와줍니다.

공유된 객체 및 자원의 경우 해당 공유 자원을 사용하는 다른 곳에 영향을 줄 수 있으므로 그와 관련하여 주의하여 사용하여야 합니다.

Flyweight

Flyweight 패턴을 UML로 도식화하면 아래와 같습니다.



Structure

Flyweight : 공유할 자원 및 인터페이스를 가지고 있는 추상 부모 객체

ConcreteFlyweight : Flyweight를 상속받아 해당 인터페이스 함수를 구체화하는 클래스 객체

FlyweightFactory : Flyweight 객체 리스트를 관리하며 Flyweight가 가진 자원 및 인터페이스를 수행하는 클래스 객체

Implementation

구체적인 구현에 대해서 소스 코드를 통하여 살펴봅니다.

```
class Flyweight {
    var sharedState: [String]
    init(sharedState: [String]) { self.sharedState = sharedState }
    func operation(uniqueState: [String]) { }
}
```

```
class ConcreteFlyweight : Flyweight {
    override func operation(uniqueState: [String]) {
        print("Flyweight: Displaying shared (\(sharedState)) and unique
(\(uniqueState) state.\n")
    }
}

extension Array where Element == String {
    /// Returns a Flyweight's string hash for a given state.
    var key: String {
        return self.joined(separator: " ")
    }
}
```

Implementation

```
class FlyweightFactory {
    private var flyweights: [String: Flyweight]

    init(states: [[String]]) {
        var flyweights = [String: Flyweight]() // Dictionary

        for state in states {
            flyweights[state.key] = Flyweight(sharedState: state)
        }

        self.flyweights = flyweights
    }

    func getFlyweight(for state: [String]) -> Flyweight {
        let key = state.key

        guard let foundFlyweight = flyweights[key] else {
            print("FlyweightFactory: Can't find a \(key), creating new one.\n")
            let flyweight = Flyweight(sharedState: state)
            flyweights.updateValue(flyweight, forKey: key)
            return flyweight
        }

        print("FlyweightFactory: Reusing existing \(key).\n")
        return foundFlyweight
    }

    func printFlyweights() {
        print("FlyweightFactory: I have \(flyweights.count) flyweights:\n")

        for item in flyweights {
            print(item.key)
        }
    }
}
```

Implementation

```
func addCarToPoliceDatabase(_ factory: FlyweightFactory, _ plates: String, _ owner:
String, _ car: [String]) {
    print("Client: Adding a car to database.\n")

    let flyweight = factory.getFlyweight(for: car)

    flyweight.operation(uniqueState: [plates, owner])
}

let factory = FlyweightFactory(states:
[
    ["Chevrolet", "Camaro2018", "pink"],
    ["Mercedes Benz", "C300", "black"],
    ["Mercedes Benz", "C500", "red"],
    ["BMW", "M5", "red"],
    ["BMW", "X6", "white"]
])

factory.printFlyweights()
// FlyweightFactory: I have 5 flyweights:

// Mercedes Benz C300 black
// BMW M5 red
// BMW X6 white
// Chevrolet Camaro2018 pink
// Mercedes Benz C500 red
```

Implementation

```
addCarToPoliceDatabase(factory,
                       "CL234IR",
                       "James Doe",
                       ["BMW", "M5", "red"])

addCarToPoliceDatabase(factory,
                       "CL234IR",
                       "James Doe",
                       [“BMW", "X1", "red"])

factory.printFlyweights()

// BMW X6 white
// BMW X1 red
// Mercedes Benz C300 black
// Mercedes Benz C500 red
// Chevrolet Camaro2018 pink
// BMW M5 red
```

References

[1] Flyweight in Swift : https://refactoring.guru/design-patterns/flyweight/swift/example#lang-features

[2] Swift Solutions: Flyweight Pattern : https://medium.com/@emanharout/swift-solutions-flyweight-pattern-7b9dcae59f35

[3] A Design Pattern Story in Swift - Chapter 19: Flyweight : http://audreyli.me/2015/07/17/a-design-pattern-story-in-swift-chapter-19-flyweight/

[4] [Design Pattern] 플라이웨이트(Flyweight) 패턴 - 디자인 패턴 : https://palpit.tistory.com/198

[5] 플라이웨이트 패턴 : https://ko.wikipedia.org/wiki/플라이웨이트_패턴

References

[6] Flyweight Design Pattern In Swift : http://coursegalaxy.com/designpatterns/flyweight/flyweight_swift1.htm

[7] Flyweight 패턴 : https://effectiveprogramming.tistory.com/entry/Flyweight-패턴

[8] Flyweight 디자인 패턴 : http://astrod.github.io/design_pattern/2017/05/07/Flywegiht-디자인-패턴/

Thank you!