# 싱글톤 패턴

Tags 디자인 패턴

💡 디자인 패턴이란 프로그램을 설계할 때 발생했던 문제점들을 객체 간의 상호 관계 등을 이용하여 해결할 수 있도록 하나의 '규약' 형태로 만들어 놓은 것을 의미한다.

## 싱글톤 패턴(singleton pattern)

싱글톤 패턴은 하나의 클래스에 오직 하나의 인스턴스만 가지는 패턴이다. 하나의 클래스를 기반으로 여러 개의 개별적인 인스턴스를 만들 수 있지만, 그렇게 하지 않고 하나의 클래스를 기반으로 단 하나의 인스턴스를 만들어 이를 기반으로 로직을 만드는 데 쓰인다.

하나의 인스턴스를 만들어 놓고 해당 인스턴스를 다른 모듈들이 공유하며 사용하기 때문에 인스턴스를 생성할 때 드는 비용이 줄어드는 장점이 있지만, 의존성이 높아진다는 단점이 있다.

생성자가 여러번 호출되도, 실제로 생성되는 객체는 하나이며 최초로 생성된 이후에 호출된 생성자는 이미 생성한 객체를 반환시키도록 만든다.

## 사용 이유

가장 먼저 떠올릴 수 있는 이점은 아무래도 **메모리 측면**일 것이다. 최초 한번의 new 연산자를 통해서 고정된 메모리 영역을 사용하기 때문에 추후 해당 객체에 접근할 때 메모리 낭비를 방지할 수 있다. 뿐만 아니라 이미 생성된 인스턴스를 활용하니 속도 측면에서도 이점이 있다고 볼 수 있다.

또다른 이점은 다른 클래스 간에 **데이터 공유가 쉽다**는 것이다. 싱글톤 인스턴스가 전역으로 사용되는 인스턴스이기 때문에 다른 클래스의 인스턴스들이 접근하여 사용할 수 있다. 하지만 여러 클래스의 인스턴스에서 싱글톤 인스턴스의 데이터에 동시에 접근하게 되면 동시성 문제가 발생할 수 있으니 이점을 유의해서 설계하는 것이 좋다.

이 외에도 도메인 관점에서 인스턴스가 한 개만 존재하는 것을 보증하고 싶은 경우 싱글톤 패턴을 사용하기도 한다.

## 싱글톤 패턴의 문제점

싱글톤 패턴을 적용하면 위와 같은 효율에서의 이점을 얻을 수 있지만, 다음과 같은 문제점들을 수반하기 때문에 trade-off를 잘 고려해야 한다.

먼저 싱글톤 패턴을 구현하는 코드 자체가 많이 필요하다. 앞서 소개한 구현 방법 외에도 정적 팩토리 메서드에서 객체 생성을 확인하고 생성자를 호출하는 경우에 멀티스레딩 환경에서 발생할 수 있는 동시성 문제 해결을 위해 syncronized 키워드를 사용해야 한다.

두 번째는 테스트하기 어렵다는 것이다. 싱글톤 인스턴스는 자원을 공유하고 있기 때문에 테스트가 결정적으로 격리된 환경에서 수행되려면 매번 인스턴스의 상태를 초기화시켜주어야 한다. 그렇지 않으면 어플리케이션 전역에서 상태를 공유하기 때문에 테스트가 온전하게 수행되지 못한다.

세 번째로는 객체 지향 설계 원칙 중에 개방-폐쇄 원칙이 위배될 수 있다. 만약 싱글톤 인스턴스가 혼자 너무 많은 일을 하거나, 많은 데이터를 공유시키면 다른 클래스들 간의 결합도가 높아지게 되는데, 이때 개방-폐쇄 원칙이 위배된다. 결합도가 높아지게 되면, 유지보수가 힘들고 테스트도 원활하게 진행할 수 없다.

이외에도 자식클래스를 만들수 없다는 점과, 내부 상태를 변경하기 어렵다는 점 등 여러가지 문제들이 존재한다. 결과적으로 이러한 문제들을 안고있는 싱글톤 패턴은 유연성이 많이 떨어지는 패턴이라고 할 수 있다.

# Reference


https://tecoble.techcourse.co.kr/post/2020-11-07-singleton/

https://gyoogle.dev/blog/design-pattern/Singleton Pattern.html