0. 참고자료

1) 디자인 패턴

- 소프트웨어 디자인 패턴(Software Design Pattern)은 소프트웨어 공학에서 특정 문맥에서 공통

적으로 발생하는 문제에 대해 재사용 가능한 해결책

- 소스나 기계 코드로 바로 전환될 수 있는 완성된 디자인은 아니며, 다른 상황에 맞게 사용될 수

있는 문제들을 해결하는데 쓰이는 서술이나 템플릿

- 프로그래머가 어플리케이션이나 시스템을 디자인할 때 공통된 문서들을 해결하는데 쓰이는 형식

화된 가장 좋은 관행

2) 디자인 패턴의 종류

- 창조 패턴(Creational Patterns)

- 구조 패턴(Structural Patterns)

- 행위 패턴(Behavioral Patterns)

- 동시실행 패턴(Concurrency Patterns)

3) 디자인 원칙

- 애플리케이션에서 달라지는 부분을 찾아내고, 달라지지 않는 부분으로부터 분리시킨다

- 구현이 아닌 인터페이스에 맞춰서 프로그래밍한다

- 상속보다는 구성을 활용한다

4) 객체지향의 기초

- 추상화, 캡슐화, 다형성, 상속

- 훌륭한 객체지향 디자인 : 재사용성, 확장성, 관리의 용이성

5) 객체지향 원칙

- 바뀌는 부분을 캡슐화한다

- 상속보다는 구성을 활용한다

- 구현이 아닌 인터페이스에 맞춰서 프로그래밍 한다

- 서로 상호작용을 하는 객체 사이에서는 가능하면 느슨하게 결합하는 디자인을 사용해야 한다

- 클래스는 확장에 대해서는 열려 있지만 변경에 대해서는 닫혀 있어야 한다(OCP)

- 추상화된 것에 의존하라. 구상클래스에 의존하지 않도록 한다

- 친한 친구들하고만 이야기 한다

- 먼저 연락하지 마세요. 저희가 연락드리겠습니다

- 어떤 클래스가 바뀌게 되는 이유는 한가지 뿐이어야만 한다

1. 중재자 패턴(Mediator Pattern)

1) 정의

- 객체들의 집합이 상호작용하는지를 함축해 놓은 객체를 정의한 디자인 패턴. 프로그램의 실행 행위를 변경할 수 있기 때문에 행위 패턴으로 간주됨

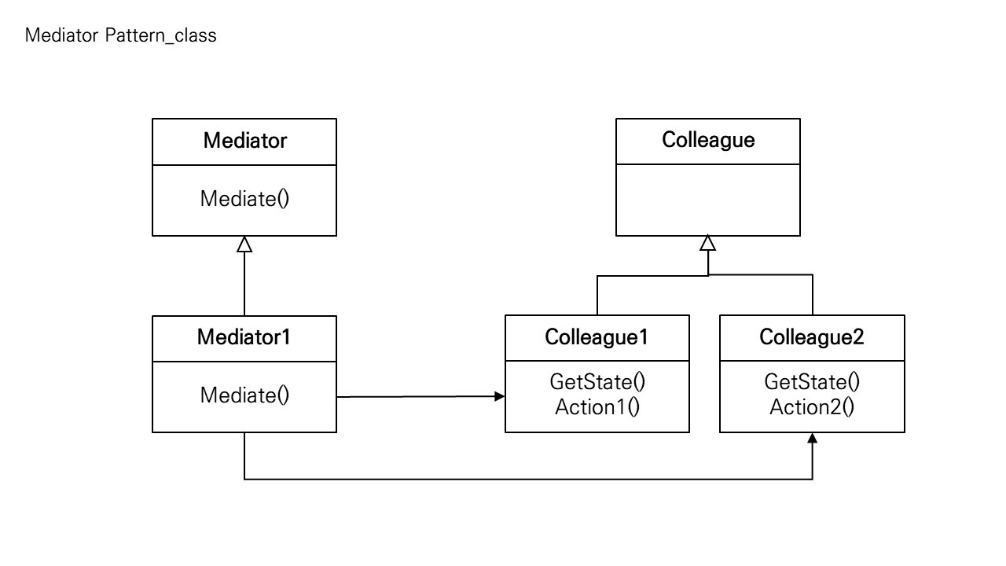
2) 목적

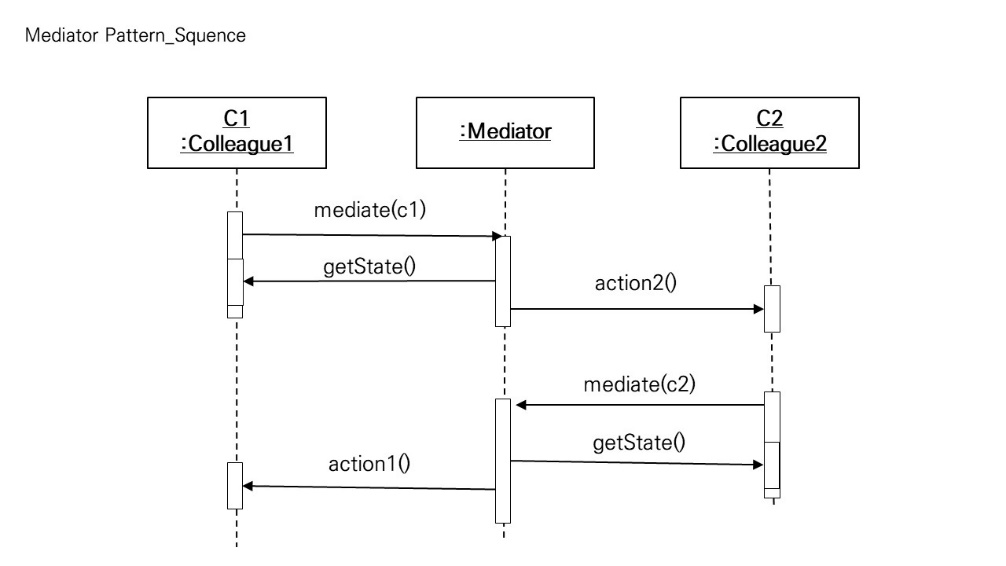
- 중재자 패턴 사용시 객체 간 통신은 중재자 객체 안에 함축됨. 객체들은 더 이상 다른 객체와

서로 직접 통신하지 않으며 중재자를 통해 통신하게 됨. 따라서 객체 간 의존성을 줄일 수 있으

므로 객체 간 결합도를 감소시킴

3) 다이어그램





4) 사용 방법

- 상태가 바뀔 때마다 중재자에게 알려주고, 중재자에서 보낸 요청을 수행함

5) 활용 예시

- 서로 연관된 GUI 구성요소들을 관리하기 위한 용도

6) 장점 & 단점

- 장점

시스템하고 각 객체를 분리시킴으로써 재사용성을 획기적으로 향상시킬 수 있음

제어 로직을 한 군데 모아 놨기 때문에 관리하기가 수월함

시스템에 들어있는 객체 사이에서 오가는 메시지의 종류를 확 줄이고 단순화시킬 수 있음

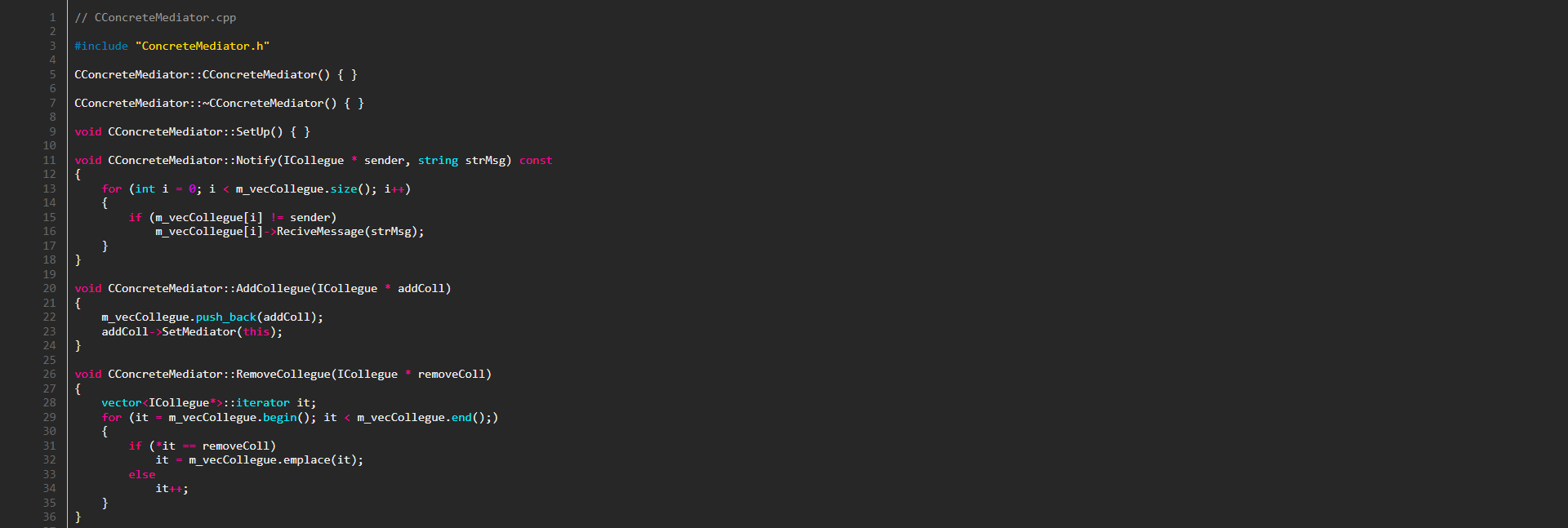
- 단점

디자인을 잘 하지 못하면 중재자 객체 자체가 너무 복잡해질 수 있음

7) 소스코드











2. 싱글턴 패턴(Singlton Pattern)

1) 정의

- 생성자가 여러 차례 호출되더라도 실제로 생성되는 객체는 하나이고 최초 생성 이후에 호출된

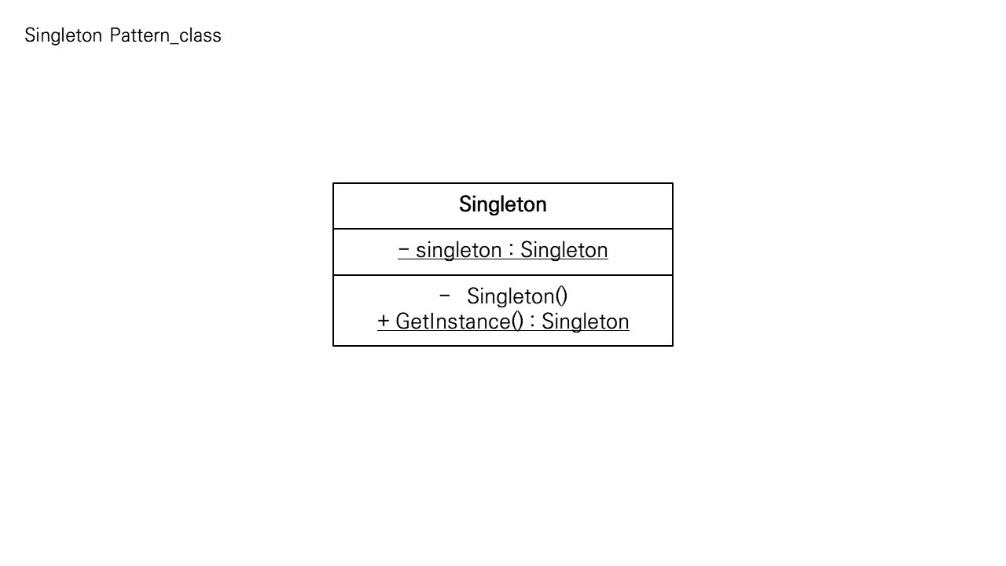
생성자는 최초의 생성자가 생성한 객체를 리턴하는 디자인 유형

2) 목적

- 해당 클래스의 인스턴스가 하나만 만들어지고, 어디서든지 그 인스턴스에 접근할 수 있도록 전역

접근 제공

3) 다이어그램



4) 사용 방법

- 싱글턴이 적용된 클래스를 생성한 후, 필요한 곳에서 인스턴스를 받아와서 사용

5) 활용 예시

- 상태 객체

- 공통된 객체를 여러 개 생성해서 사용하는 DBCP(Database Connection Pool)와 같은 상황

에서 많이 사용

6) 장점 & 단점

- 장점

어디에서든 해당 인스턴스에 쉽게 접근 할 수 있음

- 단점

클래스의 단독 인스턴스가 실제로 필요하지 않은 상황에서 불필요한 제한을 도입

응용 프로그램에 전역 상태 도입

다중 스레드를 사용하는 경우 속도와 자원 문제를 파악한 후 적절하게 구현해야 함

클래스 로더가 여러 개 있을 경우, 싱글턴이 제대로 작동하지 않고 여러 개의 인스턴스가 생길 수 있음

7) 전역변수보다 선호되는 경우

- 불필요한 변수로 전역 네임 스페이스(또는 중첩 네임 스페이스가 있는 언어, 포함 네임 스페이스)

를 오염시키지 않음

- 지연 할당 및 초기화를 허용하는 반면 많은 언어의 전역변수는 항상 리소스를 소비함

8) 소스코드





