중재자 패턴

개요

중재자 패턴은 객체들의 집합이 어떻게 상호작용하는지를 캡슐화 한 오브젝트이다.
이 패턴은 프로그램의 실행 결과를 바꿀 수 있다는 이유로 행동 패턴으로 분류된다.

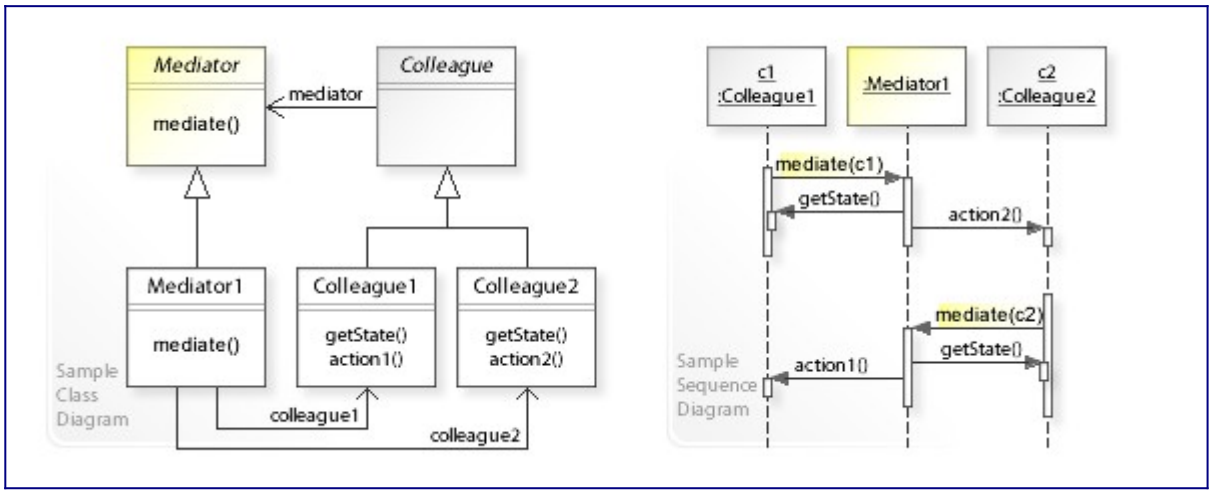
중재자 패턴을 통해 객체 간 커뮤니케이션은 중재 오브젝트로 캡슐화 되어
더이상 객체끼리 직접적으로 커뮤니케이션 되지 않고, 중재자를 통해 이루어진다.
이는 서로 커뮤니케이션 하는 객체간 의존성과 커플링을 줄여준다.

중재자 패턴이 해결할 수 있는 문제

서로 커뮤니케이션하는 객체들 간의 과도한 커플링
독자적으로 객체들간의 상호작용을 바꿔야 하는 경우

중재자 패턴이 제시하는 해결책

객체들간의 상호작용을 캡슐화한 중재자를 정의한다. (왼쪽 사진)
객체들의 상호작용을 객체간 직접적으로 연결하지 않고, 중재자에게 델리게이트 한다. (오른쪽 사진)



정의

중재자 패턴의 본질은 **객체들 간의 상호작용을 정의한 객체를 정의하는 것**이다.
이는 객체들이 명시적으로 다른 객체를 참조해야 하지 않도록 하여 느슨한 커플링을 만들어 내고,
객체들의 다양한 상호작용을 독립적으로 이루어지게 해준다.

클라이언트 클래스들은 중재자를 통해 다른 클라이언트에게 메세지를 보내고,
다른 클래스의 메세지를 중재자의 이벤트를 통해 받을 수 있다.

옵저버 패턴과의 차이

옵저버 패턴은 는 구독자(클라이언트)가 발행된 이벤트를 받기만 하지만,
중재자 패턴은 중재자를 통해 서로 통신한다는 것에서 차이가 존재한다.

예시 코드

```
interface IComponent
{
    void SetState(object state);
}

class Component1 : IComponent
{
    internal void SetState(object state)
    {
        throw new NotImplementedException();
    }
}

class Component2 : IComponent
{
    internal void SetState(object state)
    {
        throw new NotImplementedException();
    }
}

// Mediates the common tasks
class Mediator
{
    internal IComponent Component1 { get; set; }
    internal IComponent Component2 { get; set; }

    internal void ChangeState(object state)
    {
        this.Component1.SetState(state);
        this.Component2.SetState(state);
    }
}
```

도움을 받은 사이트

https://en.wikipedia.org/wiki/Mediator_pattern
https://www.crocus.co.kr/1542 옵저버와 중재자의 차이 부분 참고