Proxy Pattern

4조

유현수 윤지훈 이윤건 장원혁

GoF Design Pattern

객체 지향 프로그래밍 설계를 할 때

자주 발생하는 문제들을 피하기 위해 사용되는 표준 패턴.

GoF Design Pattern

생성(Creational) 패턴
객체 생성에 관련된 패턴객체의 생성과 조합을 캡슐화해 특정 객체가 생성되거나 변경되어도 프로그램 구조에 영향을 크게 받지 않도록 유연성을 제공한다.

구조(Structural) 패턴
클래스나 객체를 조합해 더 큰 구조를 만드는 패턴예를 들어 서로 다른 인터페이스를 지닌 2개의 객체를 묶어 단일 인터페이스를 제공하거나 객체들을 서로 묶어 새로운 기능을 제공하는 패턴이다.

행위(Behavioral)
객체나 클래스 사이의 알고리즘이나 책임 분배에 관련된 패턴한 객체가 혼자 수행할 수 없는 작업을 여러 개의 객체로 어떻게 분배하는지, 또 그렇게 하면서도 객체 사이의 결합도를 최소화하는 것에 중점을 둔다.

GoF Design Pattern

목적 / 범위	생성	구조	행위
클래스	Factory Method	Adapter(Class)	Interpreter Template Method
객체	Abstract Factory Builder Prototype Singleton	Adapter(Object) Bridge Composite Decorator Facade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

Proxy Pattern

Proxy ??

- 다른 무언가와 이어지는 인터페이스의 역할을 하는 클래스
- 어떤 객체에 대한 접근을 제어하기 위한 용도로 대리인이나 대변인에 해당하는 객체를 제공하는 패턴
- 실제 대상 객체가 메모리에 존재하지 않아도 기본적인 정보를 참조하거나 설정할 수 있고, 실제 객체의 기능이 반드시 필요한 시점까지 객체의 생성을 미룰 수 있음

Proxy Pattern

Proxy ??

대리, 대리인, 대용물

무엇을?

대신 처리하는 것!! / 분업해주는 것!!

Proxy Pattern

Client

동영상 SNS 사이트

마우스 커서를 대면
미리보기 영상 실행

Developer

Proxy Pattern

Proxy Pattern

사이트 로딩 Error

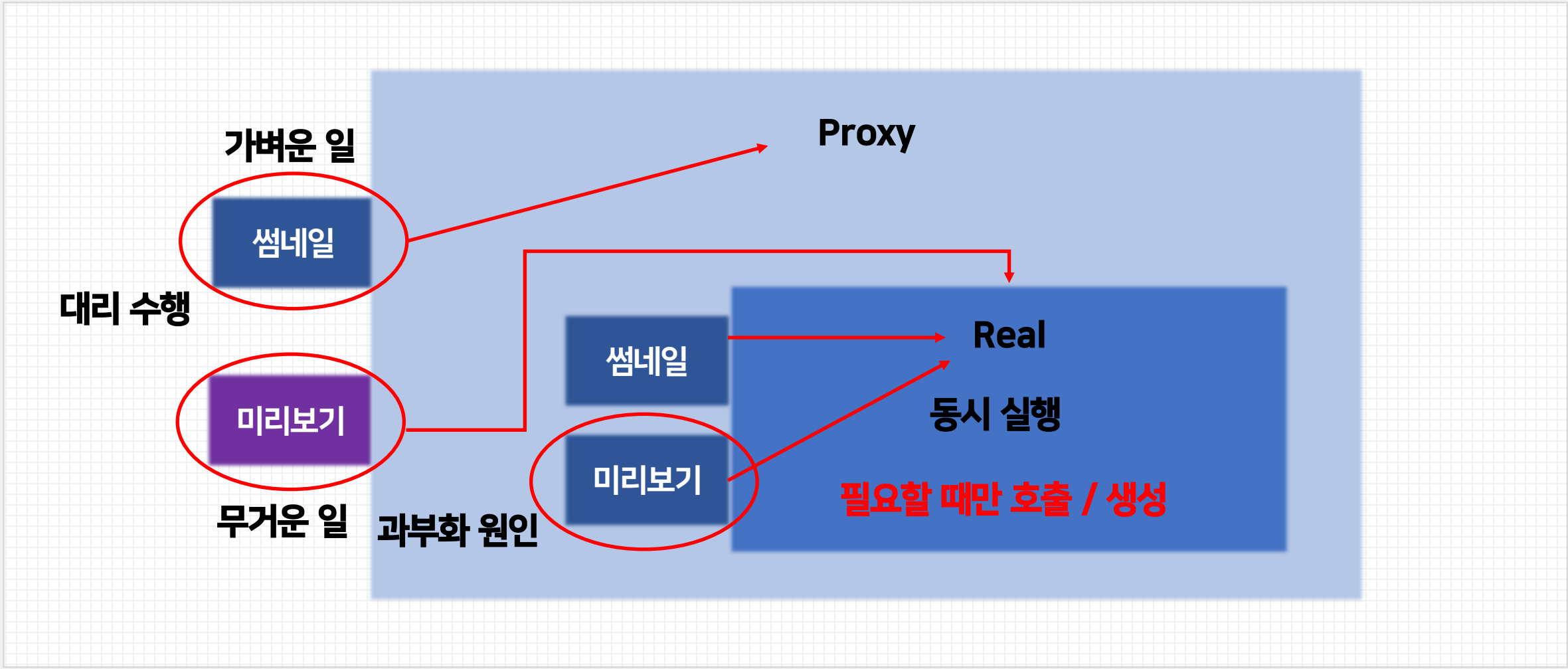
Proxy Pattern

사이트 로딩 Error

영상데이터 크기, 개수 ↑ == 로딩 속도 저하

사이트 과부화

Proxy Pattern



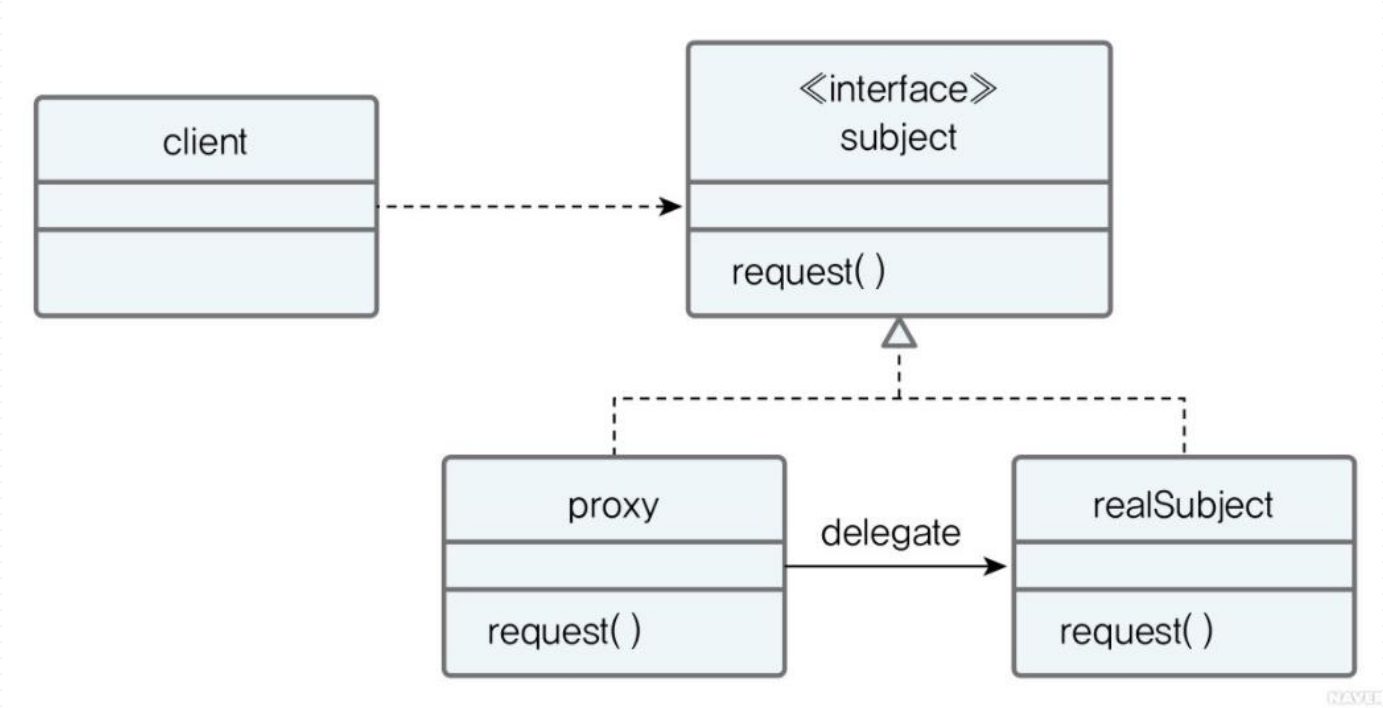
Proxy Pattern

Proxy ??

기능의 무게가 큰 일을 수행해야할 때,
많은 기능들을 한번에 수행해야할 때,

=> **'과부화'를 막기위해 고안된 '분업 / 대리인 패턴'**

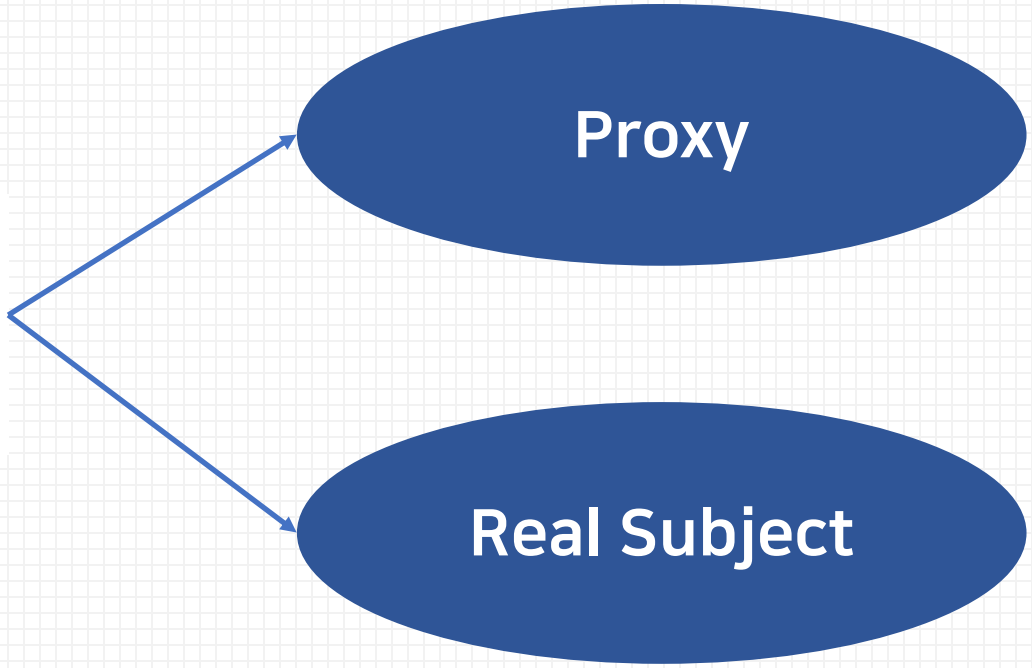
Proxy Pattern



Proxy Pattern

interface

```
public interface Thumbnail {
    public void showThumbnail();
    public void ShowPreview();
}
```



Proxy Pattern

Real Subject

```
public class RealThumbnail implements Thumbnail {

    private String thumbnail;
    private String movieUrl;

    public RealThumbnail (String thumbnail, String movieUrl) {
        this.thumbnail = thumbnail;
        this.movieUrl = movieUrl;

        //URL로부터 영상 다운받는 작업 -
        System.out.println(movieUrl + "로부터 " + thumbnail + "영상 다운로드");
    }
    // 제목 텍스트 출력 - 저용량 작업, proxy에서 진행
    @Override
    public void showThumbnail() {
        System.out.println("썸네일 : " + thumbnail);
    }
    // 프리뷰 영상 재상 - 고용량 작업, 실제 클래스에서만 진행
    @Override
    public void ShowPreview() {
        System.out.println(thumbnail + "의 프리뷰 영상 재생");
    }

}
```

영상 미리보기
(오래 걸리는 작업)

영상 데이터 다운로드

실제 영상이 있기 때문에
썸네일 출력, 미리보기 실행
둘 다 가능

Proxy Pattern

Proxy

```
public class ProxyThumbnail implements Thumbnail {
    private String thumbnail;
    private String movieUrl;

    //ShowPreview 기능을 위해 RealThumbnail 필드변수 생성
    //초기값 : null === showTitle 기능에는 쓰이지 않기 때문 / 필요할때만 값 지정
    private RealThumbnail realThumbnail;

    public ProxyThumbnail(String thumbnail, String movieUrl) {
        this.thumbnail = thumbnail;
        this.movieUrl = movieUrl;
    }

    @Override
    public void showThumbnail() {
        System.out.println("제목 : " + thumbnail);
    }

    //Proxy에서 기능 구현을 직접 하지 않음
    @Override
    public void ShowPreview() {
        if(realThumbnail == null) {
            realThumbnail = new RealThumbnail(thumbnail, movieUrl);
        }
        realThumbnail.ShowPreview();
    }

}
```

기능 실행
(썸네일 출력)

영상데이터 다운 X
썸네일 출력 담당 O

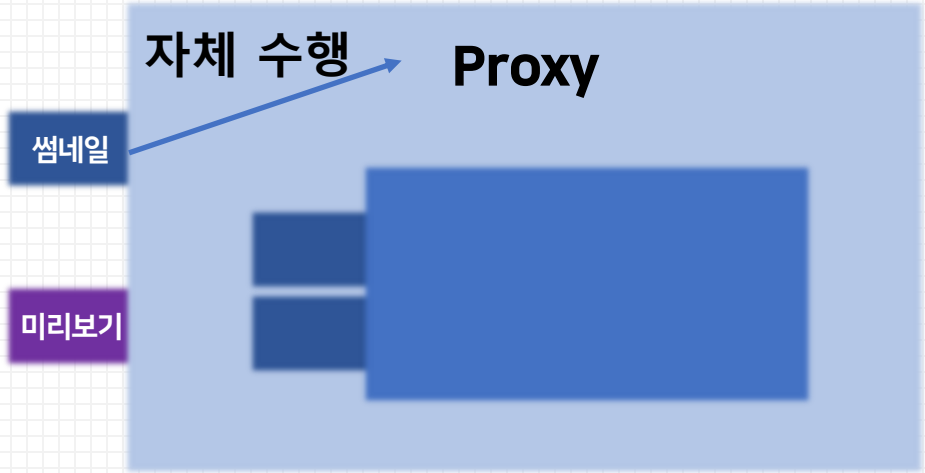
Proxy Pattern

Proxy

```
public class ProxyThumbnail implements Thumbnail {
    private String thumbnail;
    private String movieUrl;

    //ShowPreview 기능을 위해 RealThumbnail 필드변수 생성
    //초기값 : null === showTitle 기능에는 쓰이지 않기 때문 / 필요할때만 값 지정
    private RealThumbnail realThumbnail;

    public ProxyThumbnail(String thumbnail, String movieUrl) {
        this.thumbnail = thumbnail;
        this.movieUrl = movieUrl;
    }

    @Override
    public void showThumbnail() {
        System.out.println("제목 : " + thumbnail);
    }

    //Proxy에서 기능 구현을 직접 하지 않음
    @Override
    public void ShowPreview() {
        if(realThumbnail == null) {
            realThumbnail = new RealThumbnail(thumbnail, movieUrl);
        }
        realThumbnail.ShowPreview();
    }

}
```

기능 실행
(썸네일 출력)

Proxy 생성자 실행/썸네일 출력 시
Real Subject = null

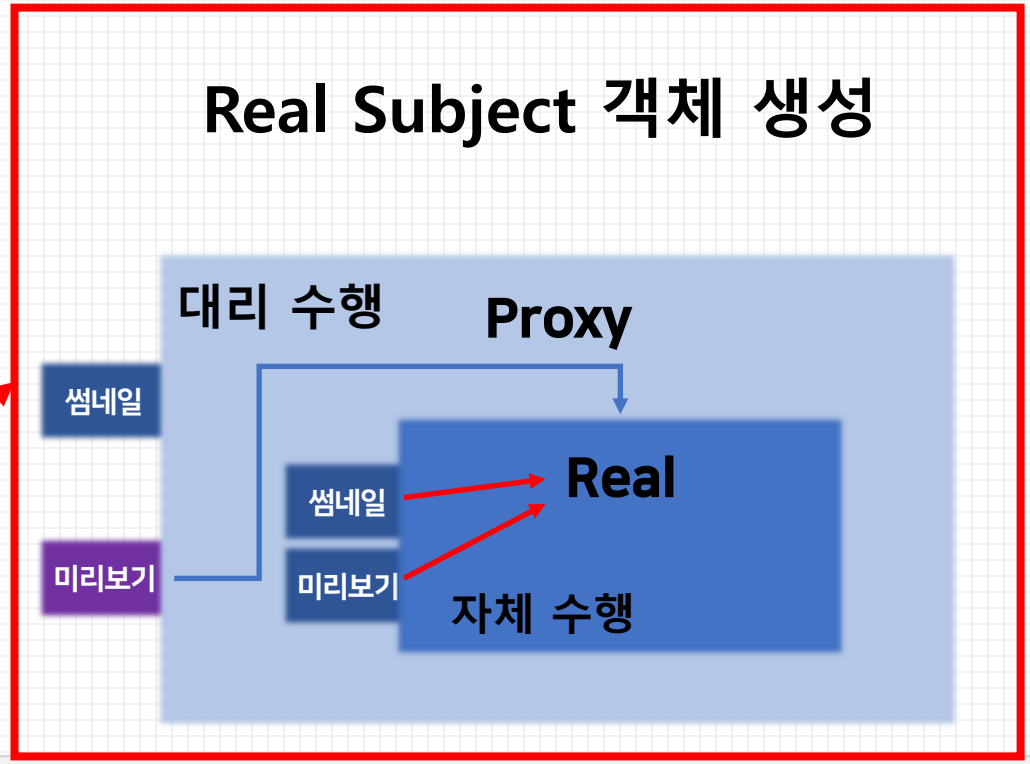


Proxy Pattern

Proxy

```
public class ProxyThumbnail implements Thumbnail {
    private String thumbnail;
    private String movieUrl;

    //ShowPreview 기능을 위해 RealThumbnail 필드변수 생성
    //초기값 : null === showTitle 기능에는 쓰이지 않기 때문 / 필요할때만 값 지정
    private RealThumbnail realThumbnail;

    public ProxyThumbnail(String thumbnail, String movieUrl) {
        this.thumbnail = thumbnail;
        this.movieUrl = movieUrl;
    }

    @Override
    public void showThumbnail() {
        System.out.println("제목 : " + thumbnail);
    }

    //Proxy에서 기능 구현을 직접 하지 않음
    @Override
    public void ShowPreview() {
        if(realThumbnail == null) {
            realThumbnail = new RealThumbnail(thumbnail, movieUrl);
        }
        realThumbnail.ShowPreview();
    }
}
```

기능 실행
(미리보기 영상 실행)



Proxy Pattern

Client

```
public class Client {
    public static void main(String[] args) {
        ArrayList<Thumbnail> video = new ArrayList<Thumbnail>();

        video.add(new ProxyThumbnail("java 1주차 강의", "java_1st week.mp4"));
        video.add(new ProxyThumbnail("java 2주차 강의", "java_2st week.mp4"));
        video.add(new ProxyThumbnail("java 3주차 강의", "java_3st week.mp4"));
        video.add(new ProxyThumbnail("java 4주차 강의", "java_4st week.mp4"));

        // 제목 텍스트 출력 == Proxy에서 처리
        System.out.println("****** 썸네일 출력 (Proxy) ******");
        for(Thumbnail thumbnail : video ) {
            thumbnail.showThumbnail();
        }
        System.out.println();

        // 영상 프리뷰 출력 == Real Subject에서 처리
        System.out.println("****** 미리보기 영상 실행 (Real Subject) ******");
        for(int i = 0; i < video.size(); i++) {
            // Proxy가 Real Subject 객체 생성 후 ShowPreview 실행
            video.get(i).ShowPreview();
        }

        // 두 번째 출력부터는, url 다운로드 X
        System.out.println();
        for(int i = 0; i < video.size(); i++) {
            video.get(i).ShowPreview();
        }
```

결과

```
****** 썸네일 출력 (Proxy) ******
제목 : java 1주차 강의
제목 : java 2주차 강의
제목 : java 3주차 강의
제목 : java 4주차 강의

****** 미리보기 영상 실행 (Real Subject) ******
java_1st week.mp4로부터 java 1주차 강의영상 다운로드
java 1주차 강의의 프리뷰 영상 재생
java_2st week.mp4로부터 java 2주차 강의영상 다운로드
java 2주차 강의의 프리뷰 영상 재생
java_3st week.mp4로부터 java 3주차 강의영상 다운로드
java 3주차 강의의 프리뷰 영상 재생
java_4st week.mp4로부터 java 4주차 강의영상 다운로드
java 4주차 강의의 프리뷰 영상 재생

java 1주차 강의의 프리뷰 영상 재생
java 2주차 강의의 프리뷰 영상 재생
java 3주차 강의의 프리뷰 영상 재생
java 4주차 강의의 프리뷰 영상 재생
```

Proxy Pattern

proxy pattern 장점

1. 사이즈가 큰 객체가(ex : 영상) 로딩되기 전에도 프록시를 통해 참조를 할 수 있다.
2. 실제 객체의 public, protected 메소드들을 숨기고 인터페이스를 통해 노출시킬 수 있다.
3. 로컬에 있지 않고 떨어져 있는 객체를 사용할 수 있다.
4. 원래 객체의 접근에 대해서 사전처리를 할 수 있다.

proxy pattern 단점

1. 객체를 생성할때 한단계를 거치게 되므로, 빈번한 객체 생성이 필요한 경우 성능이 저하될 수 있다.
2. 프록시 내부에서 객체 생성을 위해 스레드가 생성, 동기화가 구현되야 하는 경우 성능이 저하될 수 있다.
3. 로직이 난해해져 가독성이 떨어질 수 있다.

Summary

Design Pattern

자주 발생하는 문제들을 피하기 위해 사용되는 표준 패턴.

Proxy Pattern

프로그램 성능 개선을 위해 고안된 Design Pattern의 한 종류
(‘대리’, '분업‘, ‘과부화 방지')

Reference

https://namu.wiki/w/%EB%94%94%EC%9E%90%EC%9D%B8%20%ED%8C%A8%ED%84%B4
https://blog.naver.com/eh034/222351020566
https://yupdown.tistory.com/3
https://4z7l.github.io/2020/12/25/design_pattern_GoF.html
https://jdm.kr/blog/235
https://blog.naver.com/2feelus/220655183083
https://blog.naver.com/dktmrorl/222090507671
https://www.youtube.com/watch?v=lJES5TQTTWE
https://www.youtube.com/watch?v=Nc-3QUJicvo