파사드 패턴

퍼사드란?

파사드(프랑스어: Façade)는 건물의 출입구로 이용되는 정면 외벽 부분을 가리키는 말이다.
건축물의 정면 외관, 건축물의 얼굴 등으로 이해하면 된다.

〉〉 햄버거를 주문하기 위해서 맥도날드 운영 프로세스를 꼭 알아야만 할까?

퍼사드 패턴이란?

어떤 소프트웨어의 복잡하고 많은 코드에 대하여 간략화된 인터페이스를 제공해주는 디자인 패턴을 의미합니다.

퍼사드 객체는 복잡한 소프트웨어 바깥쪽의 코드가 라이브러리의 안쪽 코드에 의존하는 일을 감소시켜 주고, **복잡한** 소프트웨어를 사용 할 수 있게 **간단한 인터페이스**를 제공해 줍니다.

퍼사드 패턴의 필요성

어떤 사람이 영화를 보고자 합니다. **영화를 보기** 위해서는 다음과 같은 과정을 거치게 됩니다.
음료를 준비한다 -> TV를 켠다 -> 영화를 검색한다 -> 영화를 결제한다 -> 영화를 재생한다.

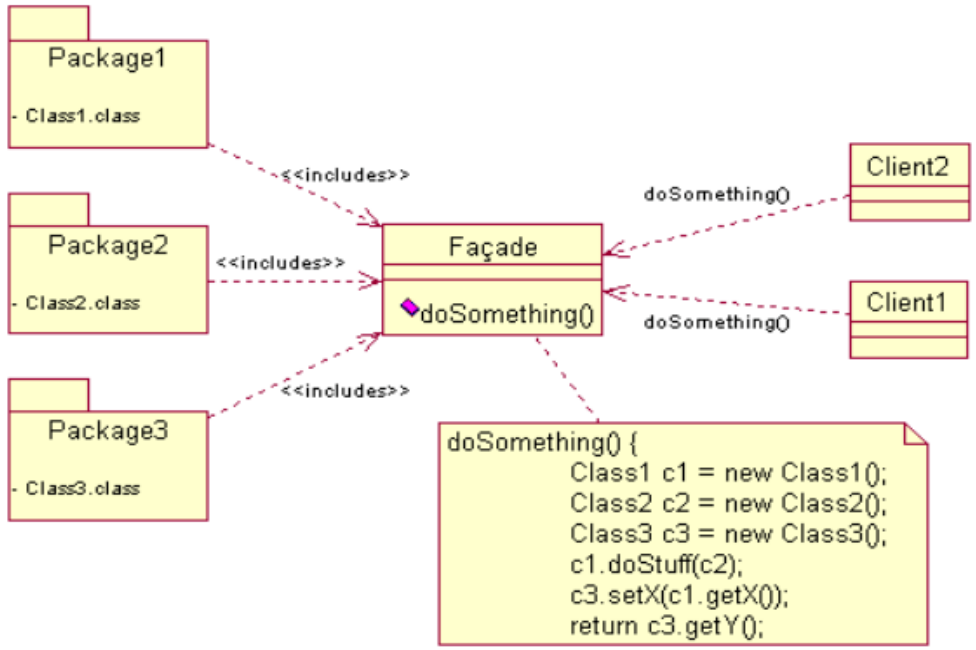
```
public void view()
{
      Beverage beverage = new Beverage("콜라");
      Remote_Control remote= new Remote_Control();
      Movie movie = new Movie("어벤져스");

      beverage.Prepare();  //음료 준비
      remote.Turn_On();    //tv를 켜다
      movie.Search_Movie();  //영화를 찾다
      movie.Charge_Movie();  // 영화를 결제하다
      movie.play_Movie();    //영화를 재생하다
}
```

Colored by Color Scripter

>> 사용자 입장에서는 영화를 보기위해서는 이런 복잡한 코드를 사용하여 영화를 봐야만 합니다.

퍼사드 패턴의 필요성

여기서 퍼사드 객체가 등장하게 되는데 퍼사드는 이런 사용자와 영화를 보기위해 사용하는 서브 클래스들 사이의 간단한 통합 인터페이스를 제공해주는 역할을 하게 됩니다.



〉〉 퍼사드 객체의 doSomething() 메서드를 호출함으로써 복잡한 서브 클래스의 사용을 도와준다.

퍼사드 패턴 예제

영화를 좀 더 쉽게 보려면 어떻게 해야 할까?

Remote_Control.java

```
1
2  public class Remote_Control {
3
4      public void Turn_On()
5      {
6          System.out.println("TV를 켜다");
7      }
8      public void Turn_Off()
9      {
10         System.out.println("TV를 끄다");
11     }
12 }
```

Colored by Color Scripter

Movie.java

```
1  public class Movie {
2
3      private String name="";
4
5      public Movie(String name)
6      {
7          this.name = name;
8      }
9
10     public void Search_Movie()
11     {
12         System.out.println(name+" 영화를 찾다");
13     }
14
15     public void Charge_Movie()
16     {
17         System.out.println("영화를 결제하다");
18     }
19     public void play_Movie()
20     {
21         System.out.println("영화 재생");
22     }
23 }
```

Colored by Color Scripter

Beveragejava

```
1  public class Beverage {
2
3      private String name="";
4
5      public Beverage(String name)
6      {
7          this.name = name;
8      }
9
10     public void Prepare()
11     {
12         System.out.println(name+" 음료 준비 완료 ");
13     }
14
15 }
```

Colored by Color Scripter

퍼사드 패턴 예제

Facade 클래스

복잡한 서브 클래스들에 대한 인스턴스를 가지며

복잡한 호출 과정을 통합하여

View_Movie() 메소드 내에 모두 구현했습니다.

Facade.java

```java
public class Facade {

    private String beverage_Name ="";
    private String Movie_Name="";

    public Facade(String beverage,String Movie_Name)
    {
        this.beverage_Name=beverage_Name;
        this.Movie_Name=Movie_Name;
    }

    public void view_Movie()
    {
        Beverage beverage = new Beverage(beverage_Name);
        Remote_Control remote= new Remote_Control();
        Movie movie = new Movie(Movie_Name);

        beverage.Prepare();
        remote.Turn_On();
        movie.Search_Movie();
        movie.Charge_Movie();
        movie.play_Movie();
    }
}
```

퍼사드 패턴 예제

사용자 입장에서는 이제 서브 클래스에 대해 알 필요가 없습니다.
단지 Facde 객체의 view_Movie() 메서드를 호출하면 서브 클래스들의 복잡한 기능을 모두 수행할 수 있습니다.

Viewer.java

```
1
2  public class Facade {
3
4      public void view()
5      {
6          Facade facade = new Facade("콜라","어벤져스");
7          facade.view_Movie();
8      }
```

활용 방안

복잡한 처리 절차를 가지는 로직이 있다면,
퍼사드 패턴을 이용해 간단하게 활용할 수 있도록 구축해봅시다.