자바 디자인패턴 보고서

20626 조성운

목차

1. 디자인 패턴이란?

디자인 패턴이란 건축학, 컴퓨터 프로그래밍에서 사용되는 용어로, 프로그램 설계에대해 문제점을 찾고 그 문제점에 대한 해답을 문서로 정리 하기 위해 고안된 방식이다.. 건축가 크리스토퍼 알렉산더가 건축학에서 만들어 사용한것을 시작으로, 이후 여러 분야에서 도입해 사용하게 되었다.

자바로 구현할땐 주로 인터페이스를 이용해 메소드들을 만들어두고 클래스를 이용해 오버라이딩 하는 방식으로 사용된다.

2. 디자인 패턴의 종류

디자인 패턴의 종류로는 현재 여러가지가 존재하는데,

어댑터패턴, 템플릿메소드패턴, 팩토리메소드패턴, 싱글톤패턴, 프로토타입패턴, 빌더패턴, 추상팩토리패턴, 브릿지패턴, 컴포짓패턴, 데코레이터패턴, 방문자패턴, 책임사슬패턴, 퍼사드패턴, 옵저버패턴, 중재자패턴, 상태패턴, 메멘토패턴, 플라이웨이트패턴, 프록시패턴, 스트래티지패턴 등 총 21 개의 패턴이 존재하고 있으며,

이번 보고서에서는 커맨드 패턴에 대해 알아보려 합니다.

3.1 커맨드 패턴이란?

여러 다른 패턴들도 존재하지만, 커맨드 패턴은 이름에 걸맞게 명령을 객체화 시켜, 저장하는 형태이다. 이러하여 객체를 캡슐화시키면 프로그램 각 부분들을 결정화 시키기 때문에 어떠한 코드를 호출한 객체에서는 어떤식으로 프로세스를 진행해야하는 지에 대해서는 알필요 가 없다. 호출당한 객체가 알아서 해줄것이기 때문이다.

커맨드 패턴은 클라이언트, 인보커, 리시버, 커맨드로 구성되어있는데,

이를 설명해 보자면,
클라이언트 : 커맨드 객체를 생성하고 인보커를 통해 리시버에 전달 하는 역할을 수행한다.
인보커 : 클라이언트의 커맨드를 리시버에 전달해주는 역할을 수행한다.
리시버 : 인보커가 전달해준 요청에따라 정해진 프로세스를 진행한다.
커맨드 : 어떠한 정해진 프로세스를 가지고 있는 객체이다.

이정도로 정리할수 있겠다.
이후에 좀더 쉽게 풀이를 해보자면,
(손님 = 클라이언트, 주문서 = 커맨드 객체, 홀서빙 = 인보커, 주방장 = 리시버)

1. 손님이 홀서빙에게 주문을 한다.
2. 홀 서빙이 주문서에 받아 적는다.
3. 홀 서빙은 주문서를 주방에 전달한다.
4. 주방장이 주문서에 따라 요리를 한다.

이러한 프로세스를 가지고 있다.
여기서 홀서빙이 바로 주방에 주문을 전달하는것이 아닌, 주문을 주문서에 적어서 전달하는데, 이를 캡슐화 하는것과 같이 생각하면 될것같다.

3.2 커맨드 패턴 구현

이제 커맨드 패턴을 구현해 보도록 하자.

간단하게 TV 를 켜는 리모콘의 작동 방식을 구현해보려 한다.

```
public interface Command {
    public void execute();
}
```

이렇게 실행하다의 뜻을 가진 execute 메소드를 Command 인터페이스 안에 선언해 두었다.

이제 TV 를 켜는 OnTV 객체를 만들어 보도록 하자.

```
public class OnTv implements Command {
    private Television televisionn;

    OnTv(Television television){
```

```
        televisionn = television;
    }
    public void execute(){
        televisionn.on();
    }
}
```

이렇게 Command 인터페이스를 상속받아 OnTv 라는 생성자를 작성하고 그 안에는 TV 의 요청을 받아 인스턴스로 저장한다. Execute 메소드는 오버라이드 하여 TV 를 켜는 코드를 작성해 두었다.

이제 TV 를 켤 리모콘 객체를 만들어 보도록 하자.

```
public class Remote {
    private Command button;

    public void setCommand(Command command) {
        button = command;
    }

    public void onClick() {
        button.execute();
    }
}
```

이런 형태로 작성했고 setCommand 는 요청을 입력받아 인스턴스로 저장하는 코드이며, onClick 은 리모콘이 버튼을 눌렀을때를 표현한 코드로, execute 메소드를 실행한다.

이제 리모콘을 작동시켜보자.

```
. public class Main {
    public static void main(String[] args) {
        Remote remote = new Remote();
        Television television = new Television();
        OnTv TvOn = new OnTv(television);
        remote.setCommand(TvOn);
        remote.onClick();
    }
}
```

리모트 객체를 생성하고, Television 객체를 생성한후 OnTv 객체에 tetevlsion 을 파라메터로 생성자를 실행시킨다. 그후 setCommand 메소드에 TvOn 객체를 전달하여 Tv 를 켤 준비를 한다. 리모콘을 드는것과 같다. 이후 onClick 메소드를 실행해 Tv 를 켜게 된다.

4. 마치며

이번 보고서를 작성하면서 느낀점은 내가 객체지향 언어인 JAVA 를 배우면서 프로그램을 짤때는 항상 최대한 객체지향성을 잘 이용한다고 생각했지만, 이번 디자인 패턴을 보고 직접 구현해 보면서, 내가 프로그램을 짜던 방식은 결코 객체지향성를 추구하지 않는 다는것을 여실히 느꼈고, 간단하게 짤수 있는 프로그램이지만, 객체지향성을 제대로 이용하려면, 어렵게 바뀌는 구나. 를 느끼게 되었다.

5. 참고문헌

https://www.youtube.com/watch?v=Y5vh6fsW7Q&list=PLsoscMhnRc7pPsRHmgN4M8tqUdWZzkpxY&index=27

https://ko.wikipedia.org/wiki/커맨드_패턴

http://javacan.tistory.com/entry/6

http://huiyu.tistory.com/entry/7-Command-Pattern 커맨드-패턴