**소프트웨어 설계 패턴 (실습 5)**

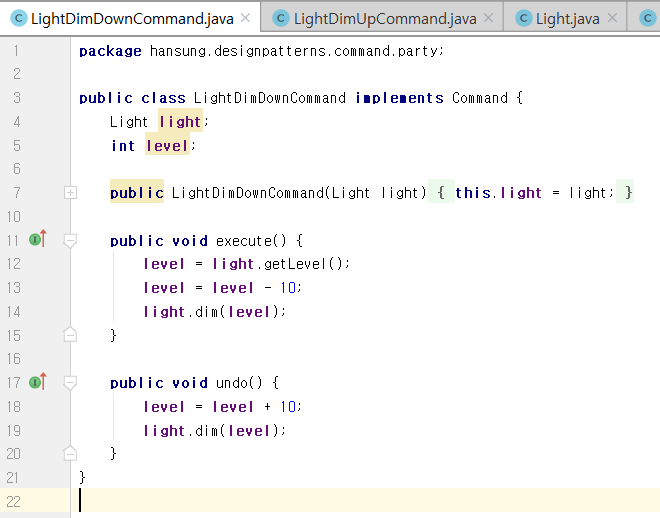
이번 실습에서는 리모컨에 파티 모드를 구현한 예제 프로젝트인

<https://github.com/kwanulee/PatternExample/tree/master/command/party> 를 바탕으로 진행합니다.

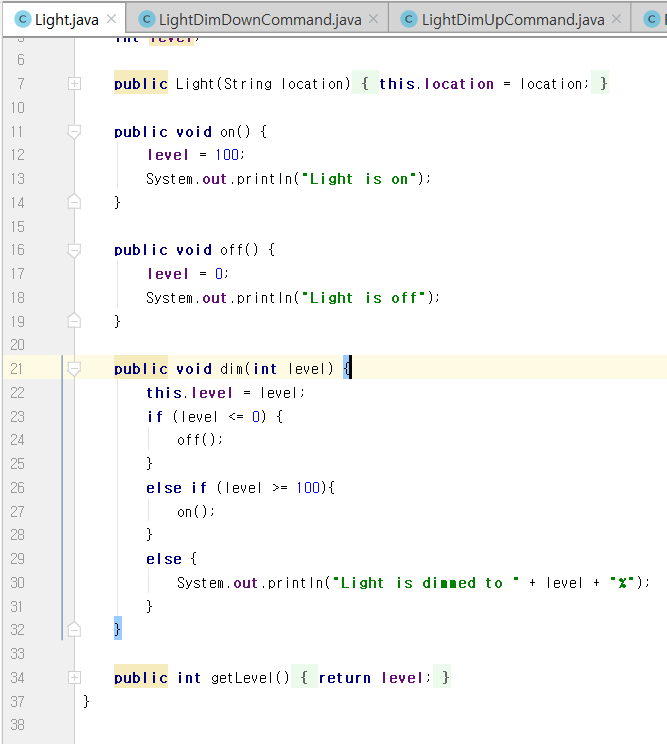
1. 이 예제에서는 Light 클래스의 on(), off() 메소드를 호출하는 두 개의 커맨드 클래스 (LightOnCommand, LightOffCommand)가 포함되어 있습니다. 하지만, Light 클래스의 dim() 메소드를 호출하는 커맨트 클래스는 없습니다. dim() 메소드는 조명의 밝기 수준을 설정할 수 있는 메소드인데, 이를 사용하여 조명의 밝기를 한단계(예, 10%) 낮출수 있는 LightDimDownCommand와 밝기를 한단계(예, 10%) 높일 수 있는LightDimUpCommand클래스를 정의해 보세요. 그리고, 아래와 같이 동작하는 지를 확인해 보세요



LightDimUpCommand

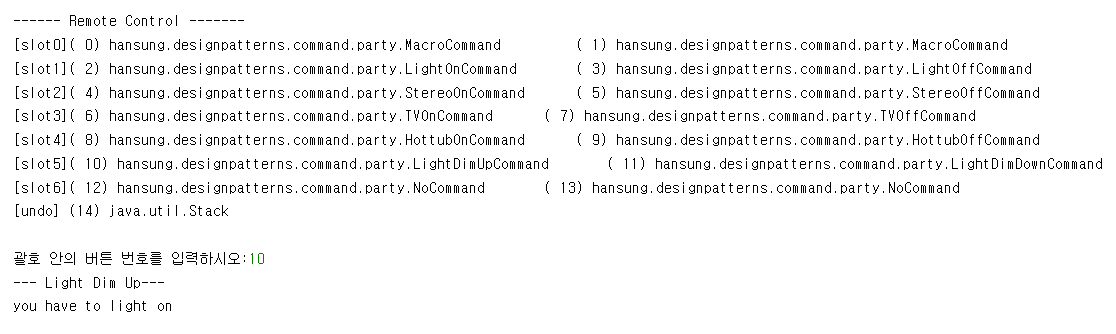


LightDimDownCommand

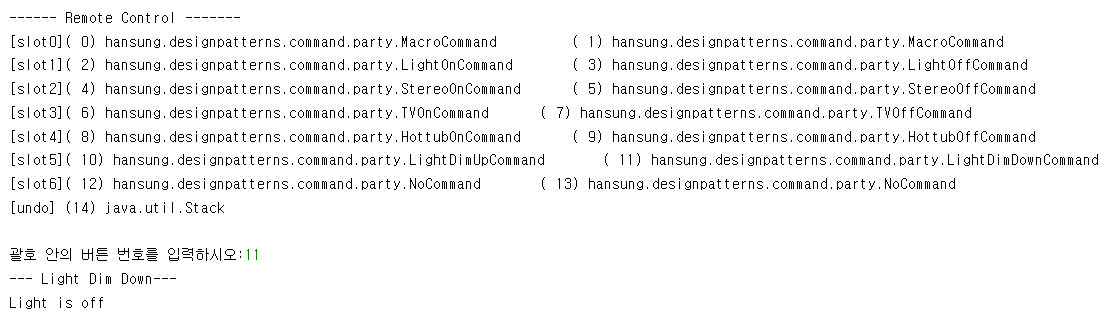


Light

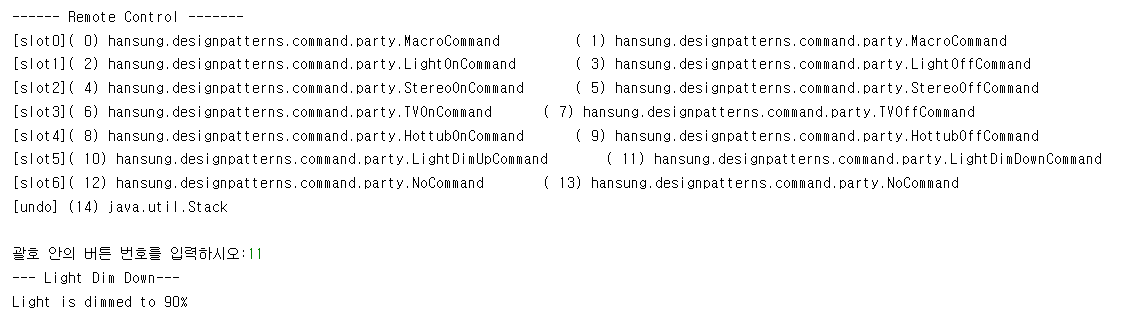
* + 두 커맨드는 Light가 켜진 상태에서만 동작해야 합니다. 만약 Light이 켜지지 않은 상태라면, 에러 메시지를 출력해야 합니다.



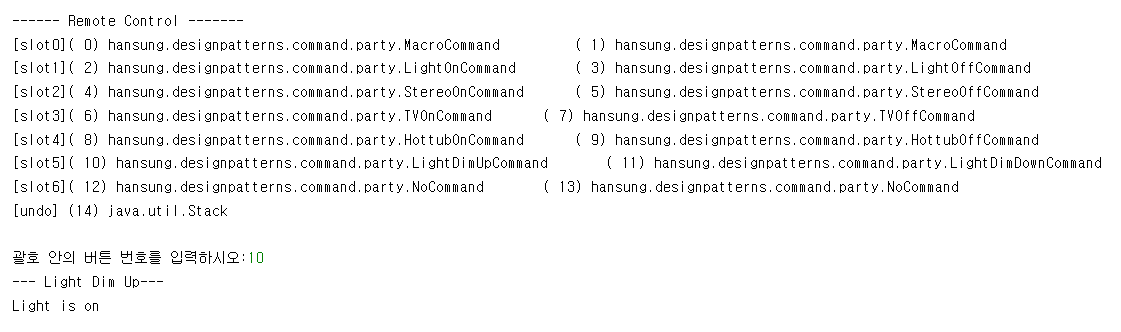
꺼진 상태에서 up



꺼진 상태에서 down



켜진 상태에서 down



켜진 상태에서 up

* + LightDimDownCommand의 수행에 의해서 밝기를 한단계 낮추었을 때, 밝기수준이 0가 된다면, 조명이 꺼진것입니다.



밝기가 10%에서 0%로 떨어지자 꺼짐으로 인식하는 모습

* + LightDimUpCommand의 수행에 의해서 높일 수 있는 밝기 수준은 최대 100%를 초과할 수 없습니다.



100%인 상태에서는 밝기가 더 올라가지 않는다

* + 두 커맨드 클래스의 undo메소드를 구현하여 이전 밝기 상태로 되돌아 갈 수 있어야 합니다.



80%로 떨구고 Undo 결과 다시 90으로 올라가는 모습

1. 1번의 실습에서 추가된 커맨드 클래스로 인해 기존 코드의 어떤 부분을 수정하였는지를 분석해 보고, 커맨드 패턴의 장점 및 단점을 논의해 보세요.

Light 와 관련된 Light클래스와 LightDimDown/UpCommand만 수정해 주면 실행이 가능한 실습이었다. 이번 실습을 통해 알게 된 점은 작업을 수행하는 객체와 명령하는 객체들의 결합이 서로 느슨하게 되어있어 코드 수정에 있어 두 객체가 독립적임이 장점으로 다가왔다. 그리고 기능을 추가함에 있어 기존 클래스와 무관하게 확장이 가능하다는 점이 장점이었다.

단점은 Undo와 같은 새로운 기능을 추가해 주려면 기존 인터페이스부터 수정을 거친 후 구상을 해 주어야 한다는 단점이 있다.

1. 현재 RemoteControl 클래스는 한번의 작업 취소(Undo)만 가능하다. 이를 수정하여 여러번의 작업 취소가 가능하도록 RemoteControl 클래스를 수정하고 이 기능을 테스트한 결과를 제출하세요.

[힌트] 실행한 커맨드를 스택에 집어넣고, 사용자가 UNDO 버튼을 누를때마다 스택 맨 위에 있는 항목을 꺼내서 undo() 메소드를 호출하면 됩니다.



RemoteControl클래스를 수정해 주었다.

Stack을 활용하여 작업들을 저장한 후에 undo작업과 off작업시에 pop시켜 줄 수 있도록 코딩했다.

실행결과

6번으로 TV를 켜고 8번으로 Hottub을 작동시켜보았다.



두 번의 Undo명령을 통해 Hottub과 TV를 차례로 Undo시켜 보았다.