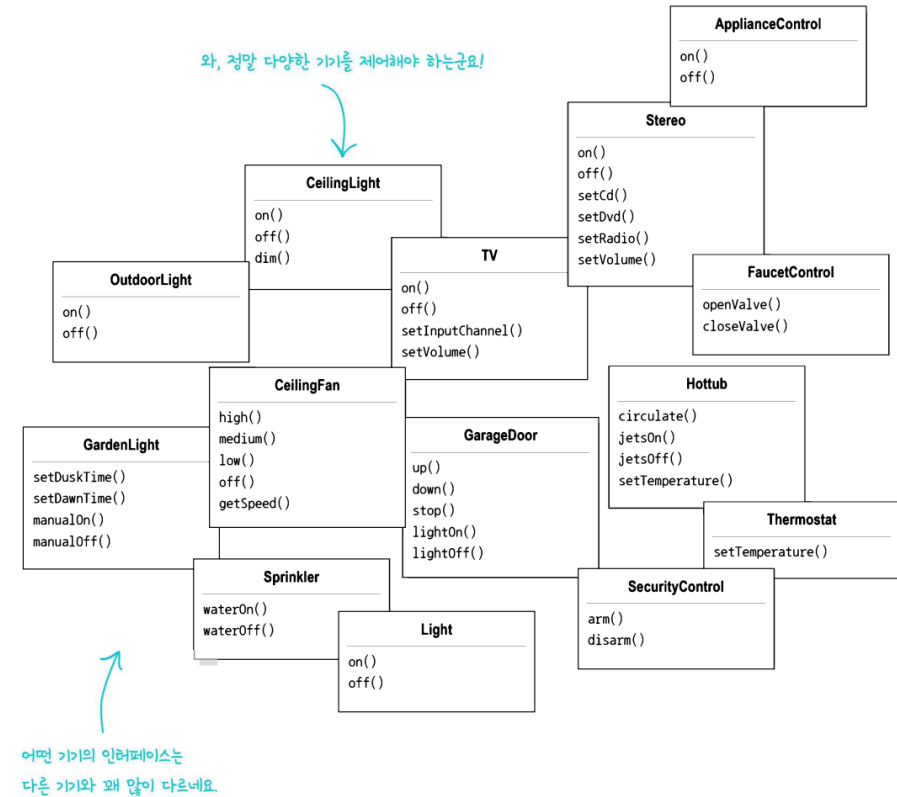
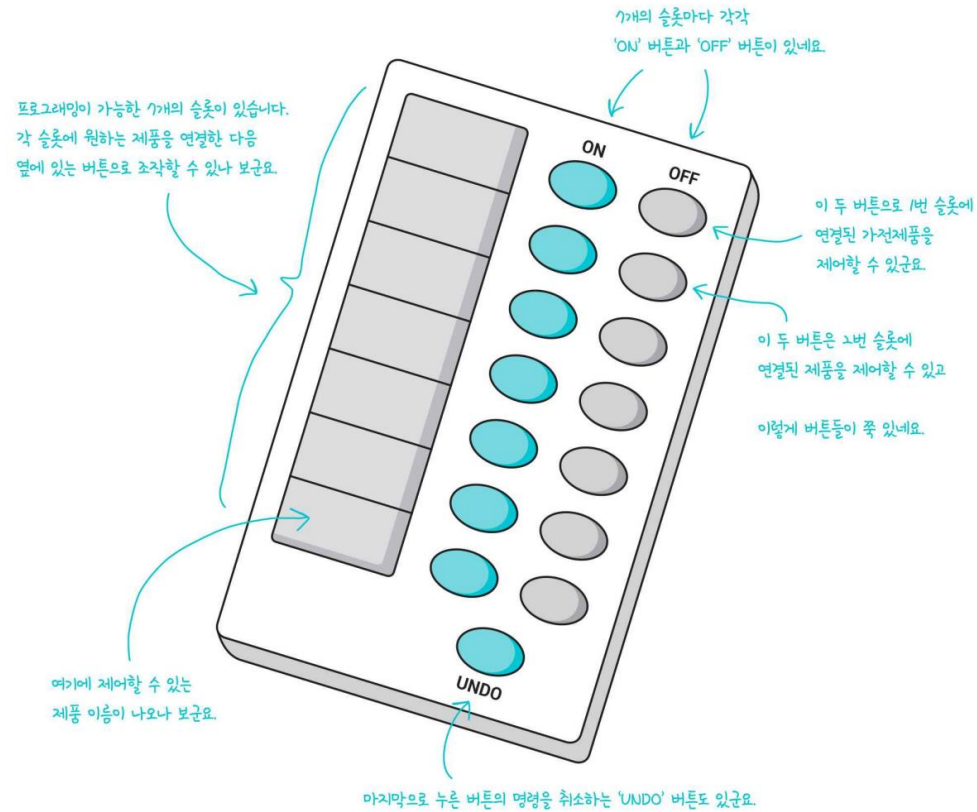




헤드퍼스트 디자인패턴

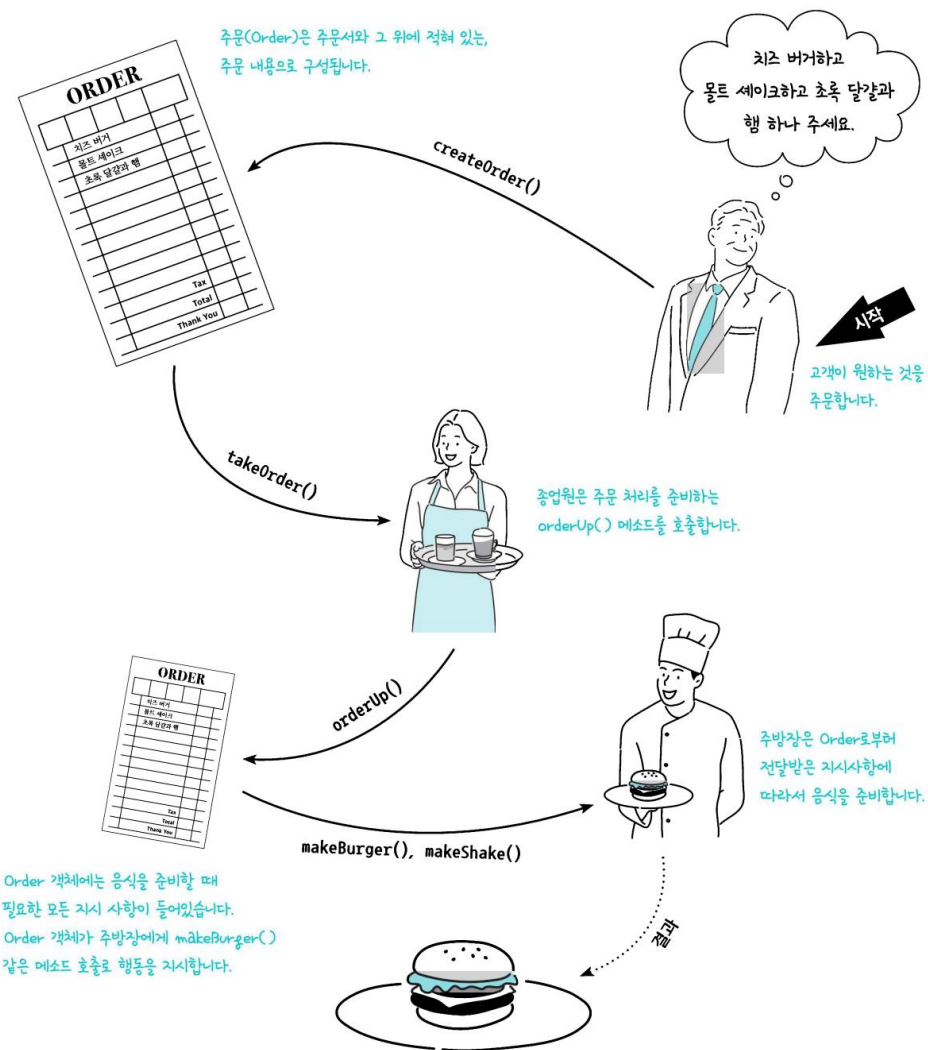
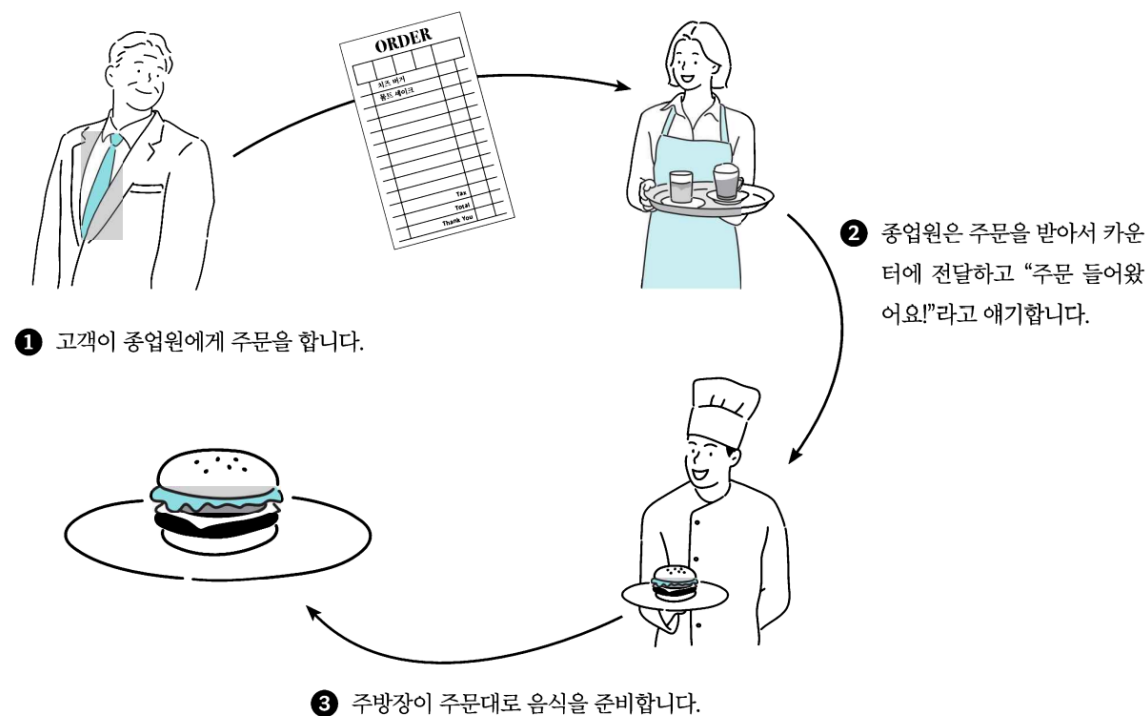
호출 캡슐화하기
커맨드패턴

하고 싶은 것은 많은데 공통 인터페이스가 없는 것 같아요

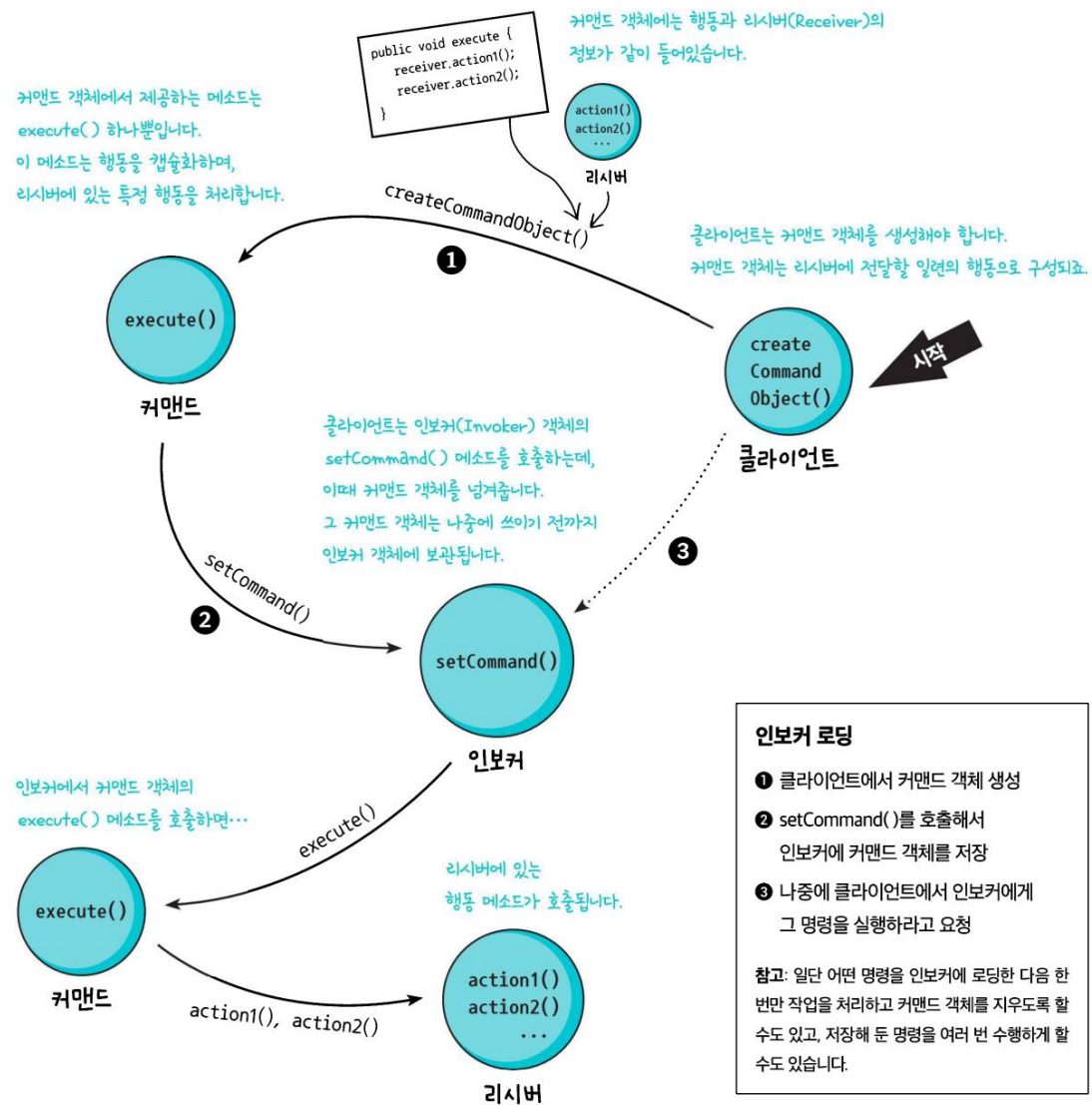


음식 주문 과정

음식 주문 과정



커맨드 패턴

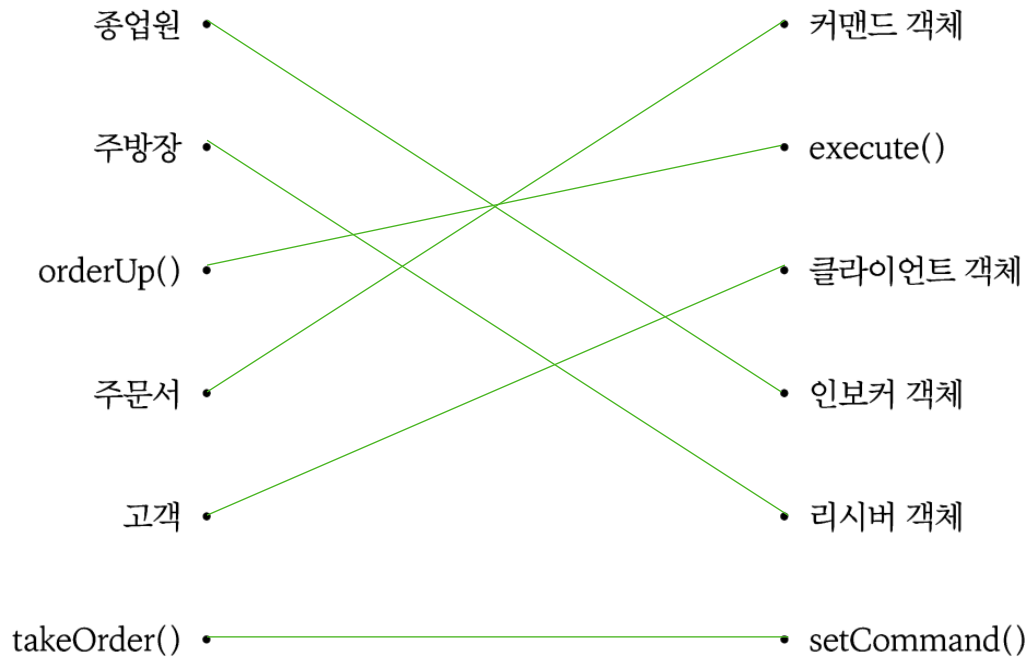


누가 무엇을 할까요?

객체마을 식당에 등장했던 요소와 그에 대응하는 커맨드 패턴의 요소를 연결해 보세요.

객체마을 식당

커맨드 패턴



커맨드 패턴 맛보기

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

Light

```
on()  
off()
```

```
public class LightOnCommand implements Command {  
    Light light;  
  
    public LightOnCommand(Light light) {  
        this.light = light;  
    }  
  
    public void execute() {  
        light.on();  
    }  
}
```

```
public class SimpleRemoteControl {  
    Command slot;  
    public SimpleRemoteControl() {}  
  
    public void setCommand(Command command) {  
        slot = command;  
    }  
    public void buttonWasPressed() {  
        slot.execute();  
    }  
}
```

```
public class RemoteControlTest {  
    public static void main(String[] args) {  
        SimpleRemoteControl remote = new SimpleRemoteControl();  
        Light light = new Light();  
        LightOnCommand lightOn = new LightOnCommand(light);  
  
        remote.setCommand(lightOn);  
        remote.buttonWasPressed();  
    }  
}
```

커맨드 객체를 인보커에게 전달해 줍니다.

정말 쉽지 않나요?

```
public class RemoteLoader {
```

```
    public static void main(String[] args) {
```

```
        RemoteControl remoteControl = new RemoteControl();
```

```
        Light livingRoomLight = new Light("Living Room");
        Light kitchenLight = new Light("Kitchen");
        CeilingFan ceilingFan = new CeilingFan("Living Room");
        GarageDoor garageDoor = new GarageDoor("Garage");
        Stereo stereo = new Stereo("Living Room");
```

장치를 각자의 위치에 맞게 생성합니다.

```
        LightOnCommand livingRoomLightOn =
            new LightOnCommand(livingRoomLight);
        LightOffCommand livingRoomLightOff =
            new LightOffCommand(livingRoomLight);
        LightOnCommand kitchenLightOn =
            new LightOnCommand(kitchenLight);
        LightOffCommand kitchenLightOff =
            new LightOffCommand(kitchenLight);
```

조명용 커맨드 객체

```
        CeilingFanOnCommand ceilingFanOn =
            new CeilingFanOnCommand(ceilingFan);
        CeilingFanOffCommand ceilingFanOff =
            new CeilingFanOffCommand(ceilingFan);
```

선풍기를 켜고 끄는
커맨드 객체

```
        GarageDoorUpCommand garageDoorUp =
            new GarageDoorUpCommand(garageDoor);
        GarageDoorDownCommand garageDoorDown =
            new GarageDoorDownCommand(garageDoor);
```

차고 문을 열고 닫는
커맨드 객체

```
        StereoOnWithCDCommand stereoOnWithCD =
            new StereoOnWithCDCommand(stereo);
        StereoOffCommand stereoOff =
            new StereoOffCommand(stereo);
```

오디오를 켜고 끄는
커맨드 객체

```
        remoteControl.setCommand(0, livingRoomLightOn, livingRoomLightOff);
        remoteControl.setCommand(1, kitchenLightOn, kitchenLightOff);
        remoteControl.setCommand(2, ceilingFanOn, ceilingFanOff);
        remoteControl.setCommand(3, stereoOnWithCD, stereoOff);
```

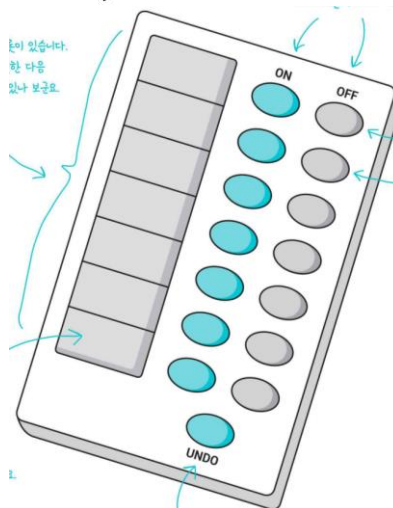
커맨드가 준비되었으니
리모컨 슬롯에 커맨드를 로드합니다.

```
        System.out.println(remoteControl);
```

toString() 메소드로
리모컨 슬롯의 정보를 출력합니다.

```
        remoteControl.onButtonWasPushed(0);
        remoteControl.offButtonWasPushed(0);
        remoteControl.onButtonWasPushed(1);
        remoteControl.offButtonWasPushed(1);
        remoteControl.onButtonWasPushed(2);
        remoteControl.offButtonWasPushed(2);
        remoteControl.onButtonWasPushed(3);
        remoteControl.offButtonWasPushed(3);
```

슬롯을 켜다가 꺼 보시다.



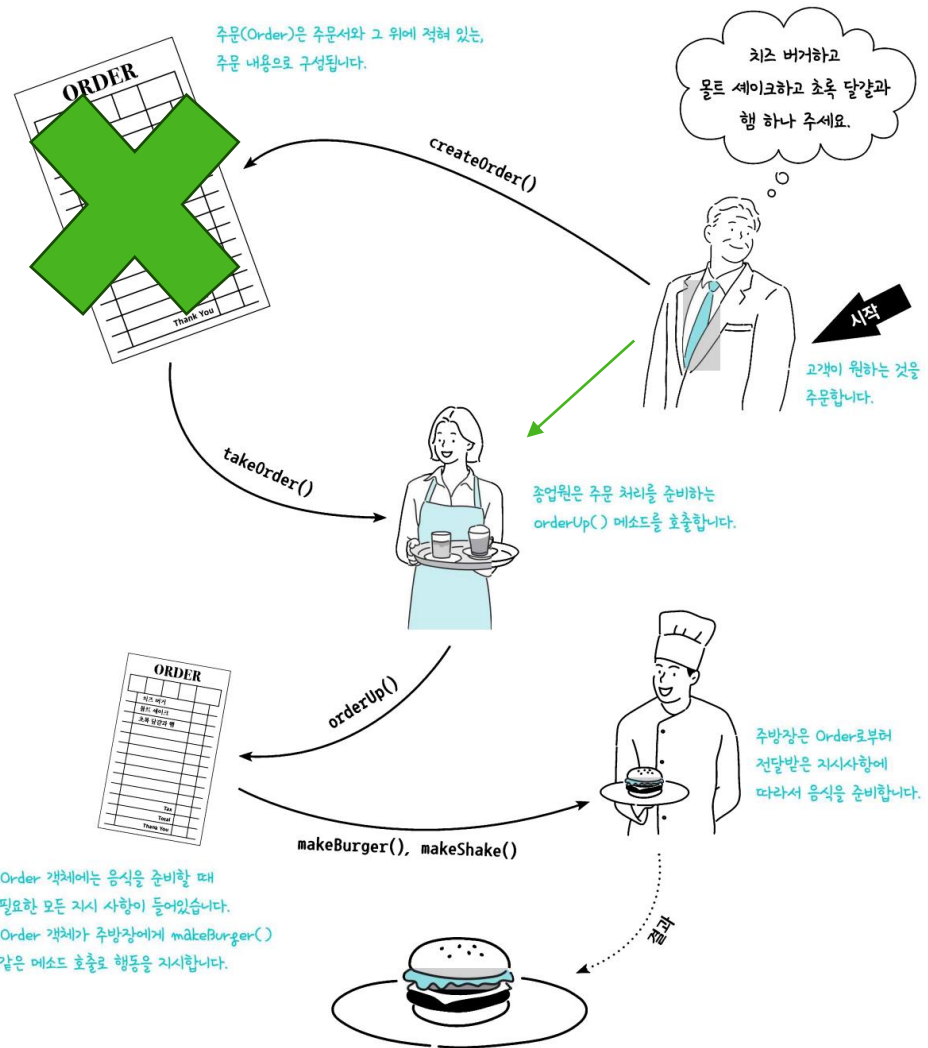
원이 있습니다.
원 다음
앞나 보군요.

```
File Edit Window Help CommandsGetThingsDone
% java RemoteLoader
----- 리모컨 -----
[slot 0] LightOnCommand      LightOffCommand
[slot 1] LightOnCommand      LightOffCommand
[slot 2] CeilingFanOnCommand CeilingFanOffCommand
[slot 3] StereoOnWithCDCommand StereoOffCommand
[slot 4] NoCommand           NoCommand
[slot 5] NoCommand           NoCommand
[slot 6] NoCommand           NoCommand

거실 조명이 켜졌습니다
거실 조명이 꺼졌습니다
주방 조명이 켜졌습니다
주방 조명이 꺼졌습니다
거실 선풍기 속도가 HIGH로 설정되었습니다
거실 선풍기가 꺼졌습니다
거실 오디오가 켜졌습니다
거실 오디오에서 CD가 재생됩니다
거실 오디오의 볼륨이 11로 설정되었습니다.
거실 오디오가 꺼졌습니다
%
```

원래 리모컨은 리모컨이
각 장치에 연결된 채널을
일부 장치에서 다중 채널로
동작할 수 있습니다. 예를 들어,
각각의 채널은 리모컨이
리모컨의 채널 번호를 입력하여
원하는 채널로 튜닝할 수 있습니다.

간단하게 사용하기



람다 표현식을 써서 고친 코드

```
public class RemoteLoader {  
    public static void main(String[] args) {  
        RemoteControl remoteControl = new RemoteControl();  
        Light livingRoomLight = new Light("Living Room");  
        ...  
        LightOnCommand livingRoomLightOn =  
        new LightOnCommand(livingRoomLight);  
        LightOffCommand livingRoomLightOff =  
        new LightOffCommand(livingRoomLight);  
        ...  
        remoteControl.setCommand(0, () -> livingRoomLight.on(),  
                                () -> livingRoomLight.off());  
        ...  
    }  
}
```

Light 객체는 이전 코드와 똑같이 만듭니다.


LightOnCommand와
LightOffCommand 구상 객체를
생성하는 부분은 자웁니다.

나중에 어떤 리모컨 버튼을 누르면
리모컨은 버튼 슬롯에 할당되어 있는
커맨드 객체의 execute() 메소드를

대신 원래 구상 커맨드 객체의 execute() 메소드에서
하던 일(거실 불을 켜거나 끄는 일)을 람다 표현식으로
적어 줍니다.


작업 취소 기능 추가하기

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




새로 추가된 undo() 메소드

```
public class LightOnCommand implements Command {  
    Light light;  
  
    public LightOnCommand(Light light) {  
        this.light = light;  
    }  
  
    public void execute() {  
        light.on();  
    }  
  
    public void undo() {  
        light.off();  
    }  
}
```



execute()는 불을 켜니까
undo()는 그냥 불을 끄기만 하면 되겠군요.

```
public class LightOffCommand implements Command {  
    Light light;  
  
    public LightOffCommand(Light light) {  
        this.light = light;  
    }  
  
    public void execute() {  
        light.off();  
    }  
  
    public void undo() {  
        light.on();  
    }  
}
```



undo() 메소드는
불을 다시 켜면 되겠네요.

```
public class RemoteControlWithUndo {
```

```
    Command[] onCommands;  
    Command[] offCommands;  
    Command undoCommand;
```

UNDO 버튼을 눌렀을 때를 대비해서 마지막으로
사용한 커맨드 객체를 넣는 변수입니다.

```
    public RemoteControlWithUndo() {  
        onCommands = new Command[7];  
        offCommands = new Command[7];
```

```
        Command noCommand = new NoCommand();  
        for(int i=0;i<7;i++) {
```

```
            onCommands[i] = noCommand;  
            offCommands[i] = noCommand;
```

```
        }
```

```
        undoCommand = noCommand;
```

```
    }
```

다른 슬롯과 마찬가지로 작업 취소 기능을
만들 때도 NoCommand를 사용합니다.
사용자가 다른 버튼을 한 번도 누르지 않은 상태에서
UNDO 버튼을 누르더라도 별문제가 없도록 말이죠.

```
    public void setCommand(int slot, Command onCommand, Command offCommand)
```

```
    {
```

```
        onCommands[slot] = onCommand;  
        offCommands[slot] = offCommand;
```

```
    }
```

```
    public void onButtonWasPushed(int slot) {  
        onCommands[slot].execute();  
        undoCommand = onCommands[slot];  
    }
```

사용자가 버튼을 누르면 우선 해당 커맨드 객체의
execute() 메소드를 호출한 다음 그 객체의 레퍼런스를
undoCommand 인스턴스 변수에 저장합니다.
ON과 OFF 버튼을 처리할 때도 같은 방법을 씁니다.

```
    public void offButtonWasPushed(int slot) {  
        offCommands[slot].execute();  
        undoCommand = offCommands[slot];  
    }
```

```
    public void undoButtonWasPushed() {  
        undoCommand.undo();  
    }
```

사용자가 UNDO 버튼을 누르면 undoCommand에
저장된 커맨드 객체의 undo() 메소드를 호출합니다.
그러면 마지막으로 했던 작업이 취소됩니다.

```
    public String toString() {
```

```
        // toString 코드
```

```
    }
```

```
}
```

undoCommand를 추가한 내용으로 갱신해 줍니다.

```
public class RemoteLoader {
```

```
    public static void main(String[] args) {
```

```
        RemoteControlWithUndo remoteControl = new RemoteControlWithUndo();
```

```
        Light livingRoomLight = new Light("Living Room");
```

```
        LightOnCommand livingRoomLightOn =  
            new LightOnCommand(livingRoomLight);
```

```
        LightOffCommand livingRoomLightOff =  
            new LightOffCommand(livingRoomLight);
```

```
        remoteControl.setCommand(0, livingRoomLightOn, livingRoomLightOff);
```

```
        remoteControl.onButtonWasPushed(0);  
        remoteControl.offButtonWasPushed(0);  
        System.out.println(remoteControl);  
        remoteControl.undoButtonWasPushed();  
        remoteControl.offButtonWasPushed(0);  
        remoteControl.onButtonWasPushed(0);  
        System.out.println(remoteControl);  
        remoteControl.undoButtonWasPushed();
```

← Light 객체와 undo() 기능이 추가된 조명을
켜고 끄는 커맨드 객체를 만듭니다.

← 조명 커맨드 객체를 리모컨 0번 슬롯에 추가합니다

← 불을 켰다가 끈 다음 작업을 취소합니다.

← 불을 끄고 켰다가 다시 작업을 취소합니다.

```
File Edit Window Help UndoCommandsDefyEntropy
% java RemoteLoader
조명이 켜졌습니다
조명이 꺼졌습니다

----- 리모컨 -----
[slot 0] LightOnCommand    LightOffCommand
[slot 1] NoCommand        NoCommand
[slot 2] NoCommand        NoCommand
[slot 3] NoCommand        NoCommand
[slot 4] NoCommand        NoCommand
[slot 5] NoCommand        NoCommand
[slot 6] NoCommand        NoCommand
[undo] LightOffCommand

조명이 켜졌습니다
조명이 꺼졌습니다
조명이 켜졌습니다

----- 리모컨 -----
[slot 0] LightOnCommand    LightOffCommand
[slot 1] NoCommand        NoCommand
[slot 2] NoCommand        NoCommand
[slot 3] NoCommand        NoCommand
[slot 4] NoCommand        NoCommand
[slot 5] NoCommand        NoCommand
[slot 6] NoCommand        NoCommand
[undo] LightOnCommand

조명이 꺼졌습니다
```