



브릿지 패턴

(Bridge Pattern)

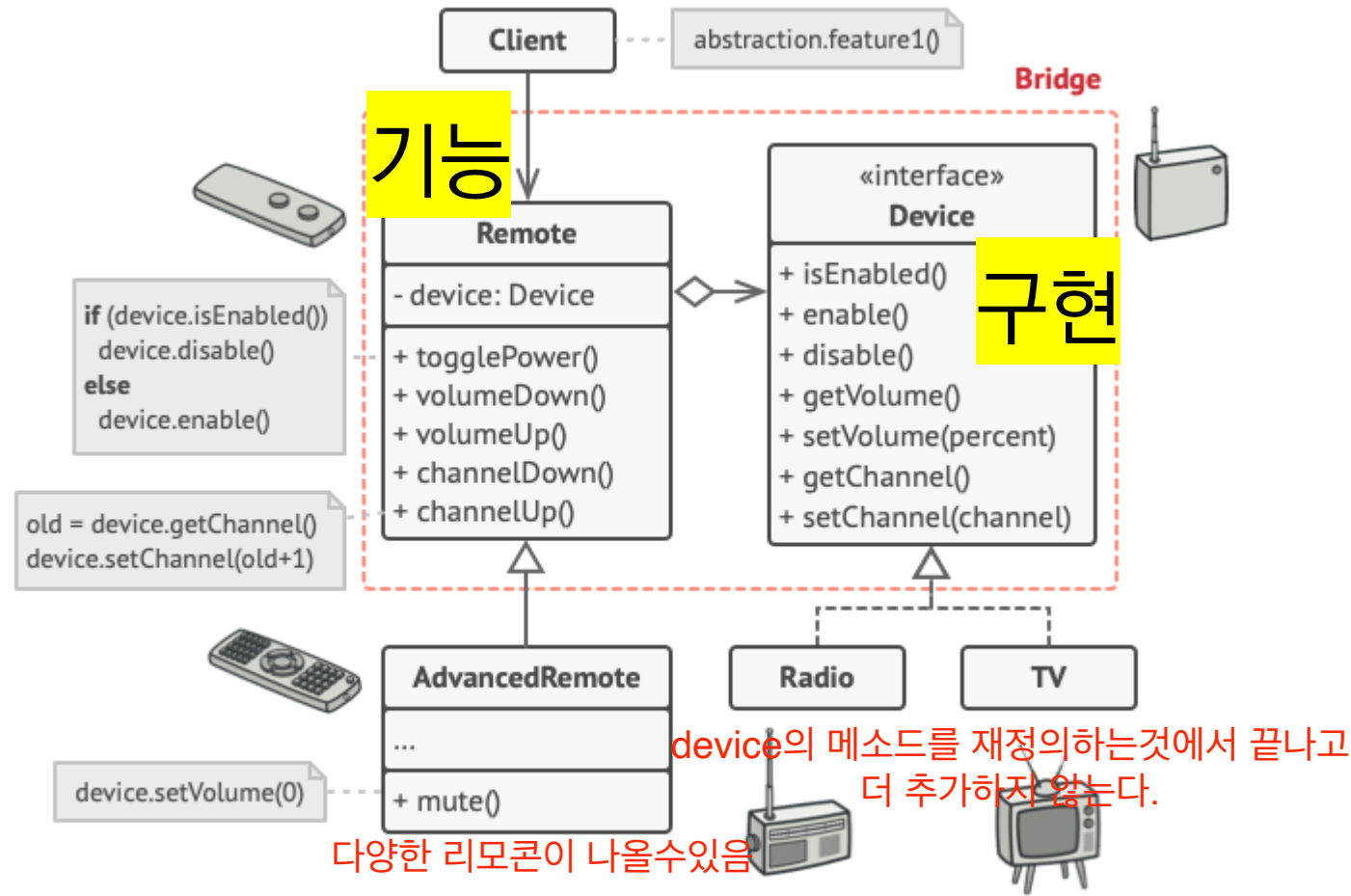
01. 브릿지 패턴이란?



- 서로 떨어져 있는 두 장소를 이어주는 다리처럼, Bridge pattern은 **기능** 클래스 계층과 구현 클래스 계층 사이의 ‘다리’를 놓아주는 패턴!

01. 브릿지 패턴이란?

- 구현(implementation)으로부터 기능(추상, abstraction) 레이어를 분리하여 이 둘이 서로 독립적으로 변화할 수 있도록 한다
- 즉, 기능과 구현에 대해서 두 개를 별도의 클래스로 구현한다



01. 브릿지 패턴이란?

기능클래스 : 내려올수록 기능 추가
구현클래스 : 이 구현에는 라디오가 있다, tv가
있다 라는 식으로 분리시키는 역할

- 기능(추상) 클래스(Abstraction)?

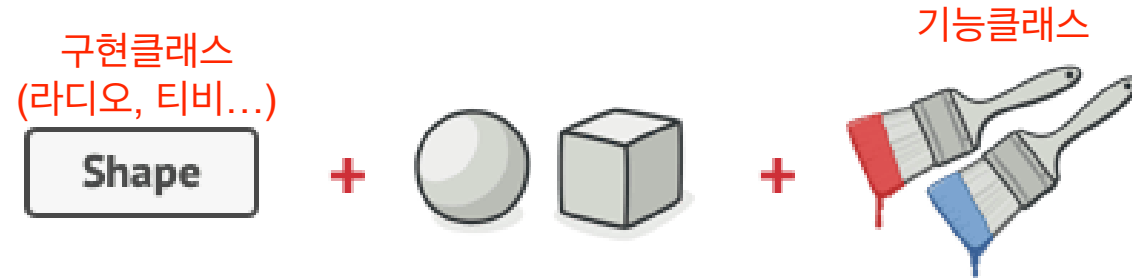
최상위 클래스가 추상클래스

- 상위 클래스는 기본적인 기능을 가지고 있음
- 하위 클래스는 기본적인 기능에 새로운 기능이 추가됨
- A 추상클래스를 상속받는 B 클래스 -> 기능 클래스 계층

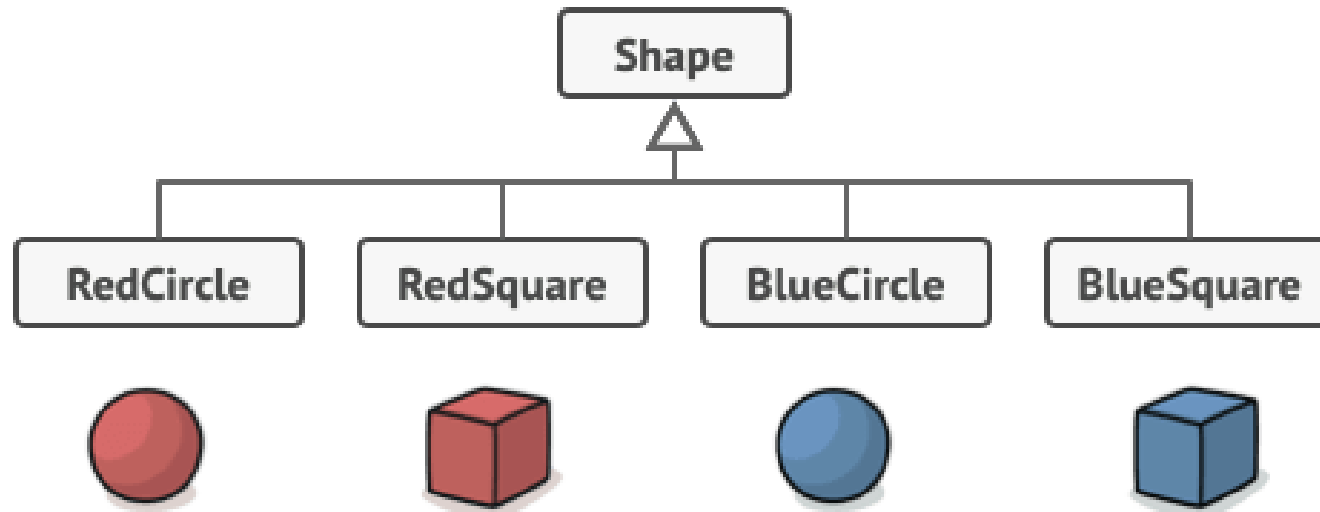
- 구현 클래스(Implementation)?

- 최상위 클래스가 인터페이스, 이 인터페이스는 자식 클래스에서 구체화될 메소드들이 정의되어 있음
- 자식 클래스에서 부모 인터페이스를 구현하여 사용
- A 인터페이스를 구현한 B 객체 -> 구현 클래스 계층

02. 분리하지 않는다면?



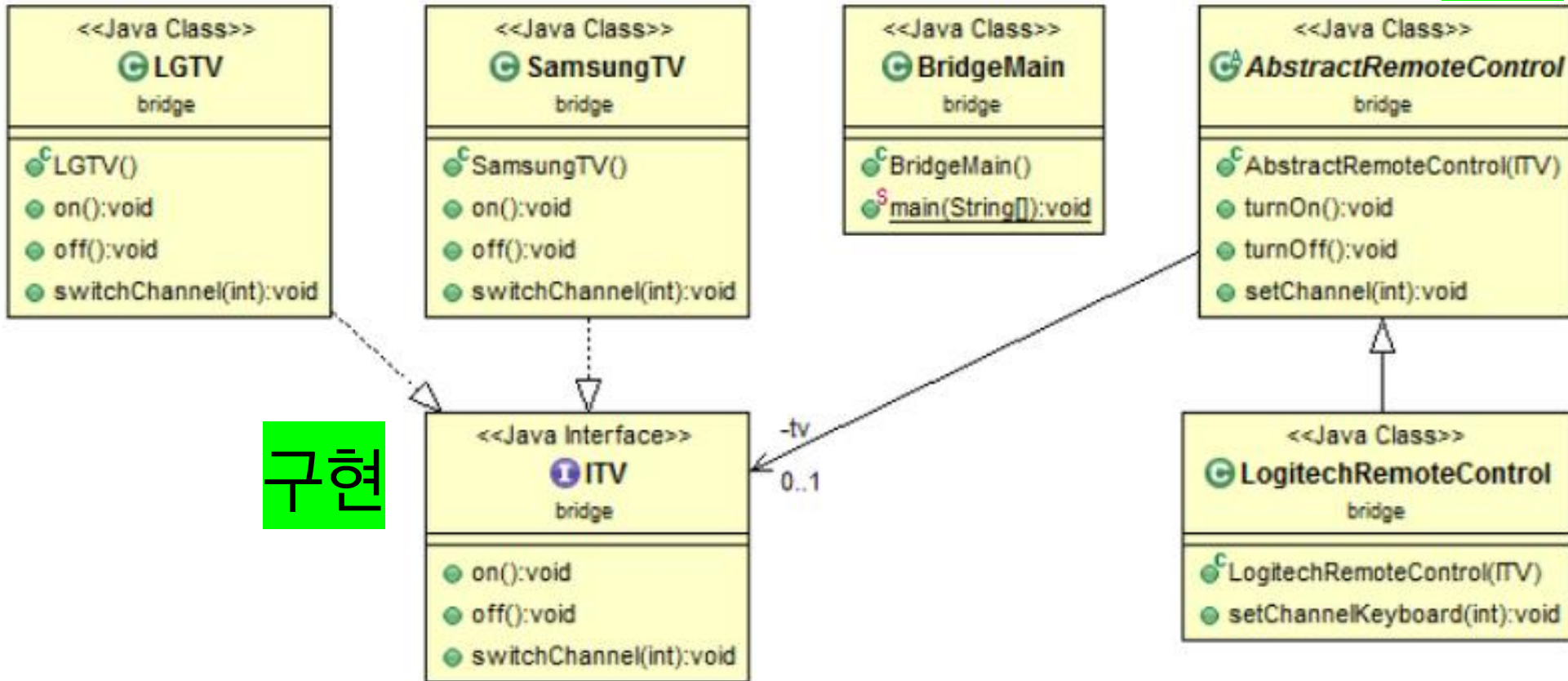
따로 만들지 않으면 빨강원, 빨강네모... 등으로 다 만들어야한다.
도형이 하나 추가될때도 모든 색깔로 다 만들어야한다.



03. 예제

다른 기능 추가하지 않고 그대로 재정의만

기능



구현

가장 기본적으로 있는 기능만 존재

03. 예제_구현 클래스 계층도

```
7
8 interface TVI{ //구현의 클래스 계층의 최상위 인터페이스
9     public void TVon();
10    public void TVoff();
11    public void changeChannel(String channel);
12 }
13
14 class SamsungTV implements TVI{ //구현 클래스 계층
15     public void TVon() {
16         System.out.println("SamsungTV on");
17     }
18     public void TVoff() {
19         System.out.println("SamsungTV off");
20     }
21     public void changeChannel(String channel) {
22         System.out.println("SamsungTV Channel : "+channel);
23     }
24 }
25
```

```
25
26 class LGTV implements TVI{
27
28     public void TVon() {
29         System.out.println("LGTV on");
30     }
31     public void TVoff() {
32         System.out.println("LGTV off");
33     }
34     public void changeChannel(String channel) {
35         System.out.println("LGTV Channel : "+channel);
36     } //구현 클래스 계층
37 }
38
```

03. 예제_기능 클래스 계층도

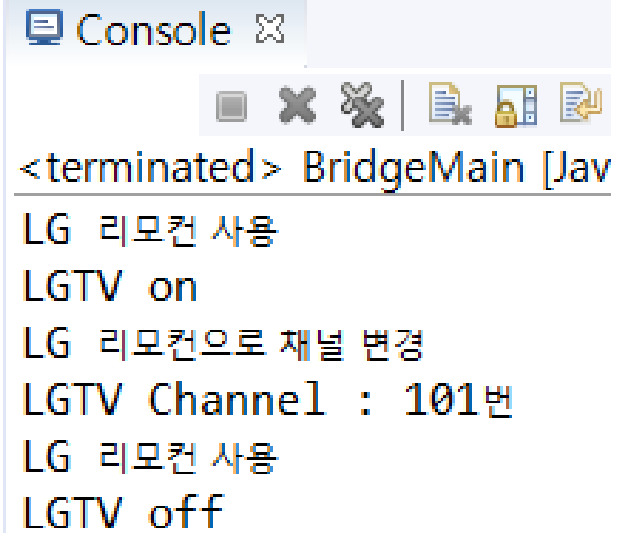
```
7 abstract class RemoteControl {  
8     //기능 계층 클래스의 최상위 추상클래스  
9     protected TVI tv;  
10 public RemoteControl(TVI tv) {  
11     this.tv = tv;  
12 }  
13  
14 abstract public void on();  
15 abstract public void off();  
16 }
```

```
18 class LGRemoteControl extends RemoteControl{  
19     public LGRemoteControl(TVI tv) {  
20         super(tv);  
21     }  
22     public void on() {  
23         System.out.println("LG 리모컨 사용");  
24         tv.TVon();  
25     }  
26     public void off() {  
27         System.out.println("LG 리모컨 사용");  
28         tv.TVoff();  
29     }  
30  
31     public void changechannel(String channel) {  
32         System.out.println("LG 리모컨으로 채널 변경");  
33         tv.changeChannel(channel);  
34     }  
35     //엘지 리모컨에만 새롭게 추가된 기능.  
36 }
```


03. 예제_main

```
public class BridgeMain {  
  
    public static void main(String[] args) {  
        LGTV LGtv = new LGTV();  
        LGRemoteControl LGRemotecontrol = new LGRemoteControl(LGtv);  
        //기능 클래스 계층의 최상위 클래스에 구현 클래스 계층의 클래스를 등록함.  
        LGRemotecontrol.on();  
        LGRemotecontrol.changechannel("101번");  
        LGRemotecontrol.off();  
    }  
}
```

기능 구현시 티비를 건드리지 않고 리모콘을 건드리면 티비까지 기능 실행 가능



```
Console  
<terminated> BridgeMain [Java]  
LG 리모컨 사용  
LGTV on  
LG 리모컨으로 채널 변경  
LGTV Channel : 101번  
LG 리모컨 사용  
LGTV off
```

04. 실제 사용 예시

- 기능(추상) 클래스 : 다양한 사용자를 위한 화면 - ex) 사용자용, 관리자용
- 구현 클래스 : 윈도우, 리눅스, 맥OS에서 구현 가능

