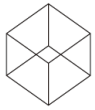
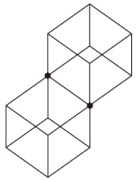
Bridge 패턴



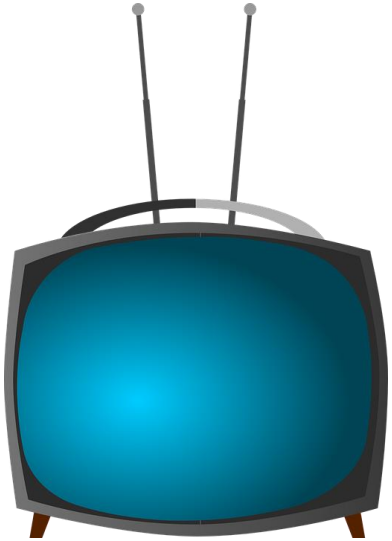
JAVA 객체 지향 디자인 패턴

UML과 GoF 디자인 패턴 핵심 10가지로 배우는



Tv 리모콘

Remote



Tv

Tv 소스코드

```
public class Tv {
    private boolean on = false;
    private int volume = 30;
    private int channel = 1;

    public boolean isEnabled() { return on;}
    public void enable() { on = true; }

    public void disable() { on = false; }

    public int getVolume() { return volume; }

    public void setVolume(int volume) {
        if (volume > 100) {
            this.volume = 100;
        } else if (volume < 0) {
            this.volume = 0;
        } else {
            this.volume = volume;
        }
    }
```

```
public int getChannel() { return channel; }

    public void setChannel(int channel) { this.channel = channel; }
    public void printStatus() {
        System.out.println("-----------------------------------");
        System.out.println("| I'm TV set.");
        System.out.println("| I'm " + (on ? "enabled" : "disabled"));
        System.out.println("| Current volume is " + volume + "%");
        System.out.println("| Current channel is " + channel);
        System.out.println("-----------------------------------\n");
    }
}
```

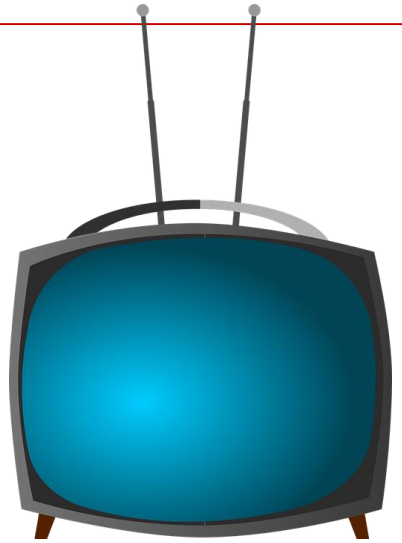
Remote 소스코드

```
public class Remote {
    protected Tv device;

    public Remote(Tv device) {
        this.device = device;
    }

    public void power() {
        System.out.println("Tv Remote: power toggle");
        if (device.isEnabled()) {
            device.disable();
        } else {
            device.enable();
        }
    }

    public void volumeDown() {
        System.out.println("Tv Remote: volume down");
        device.setVolume(device.getVolume() - 10);
    }

    public void volumeUp() {
        System.out.println("Tv Remote: volume up");
        device.setVolume(device.getVolume() + 10);
    }
```

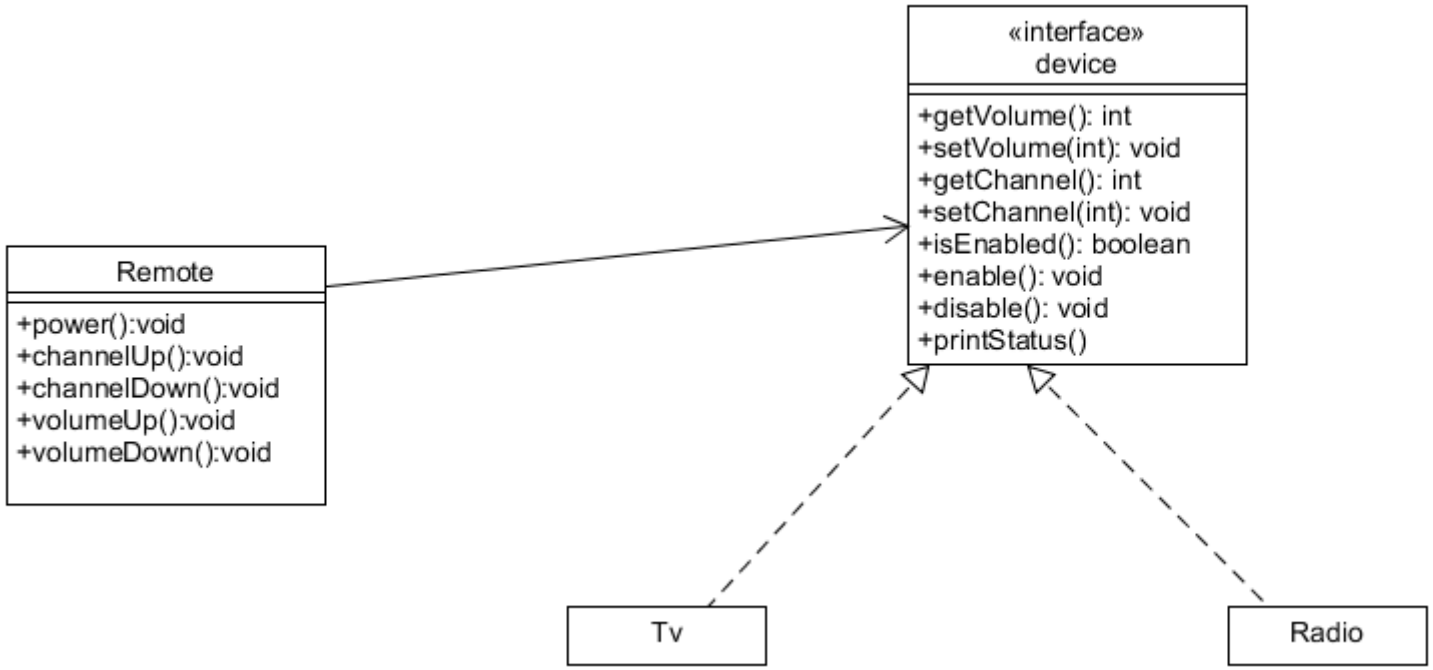
```
    public void channelDown() {
        System.out.println("Tv Remote: channel down");
        device.setChannel(device.getChannel() - 1);
    }


    public void channelUp() {
        System.out.println("Tv Remote: channel up");
        device.setChannel(device.getChannel() + 1);
    }
}
```

Radio 겸용 Tv 리모콘

Remote



클래스 다이어그램


«interface»
device
+getVolume(): int
+setVolume(int): void
+getChannel(): int
+setChannel(int): void
+isEnabled(): boolean
+enable(): void
+disable(): void
+printStatus()
Remote
+power():void
+channelUp():void
+channelDown():void
+volumeUp():void
+volumeDown():void
Tv
Radio

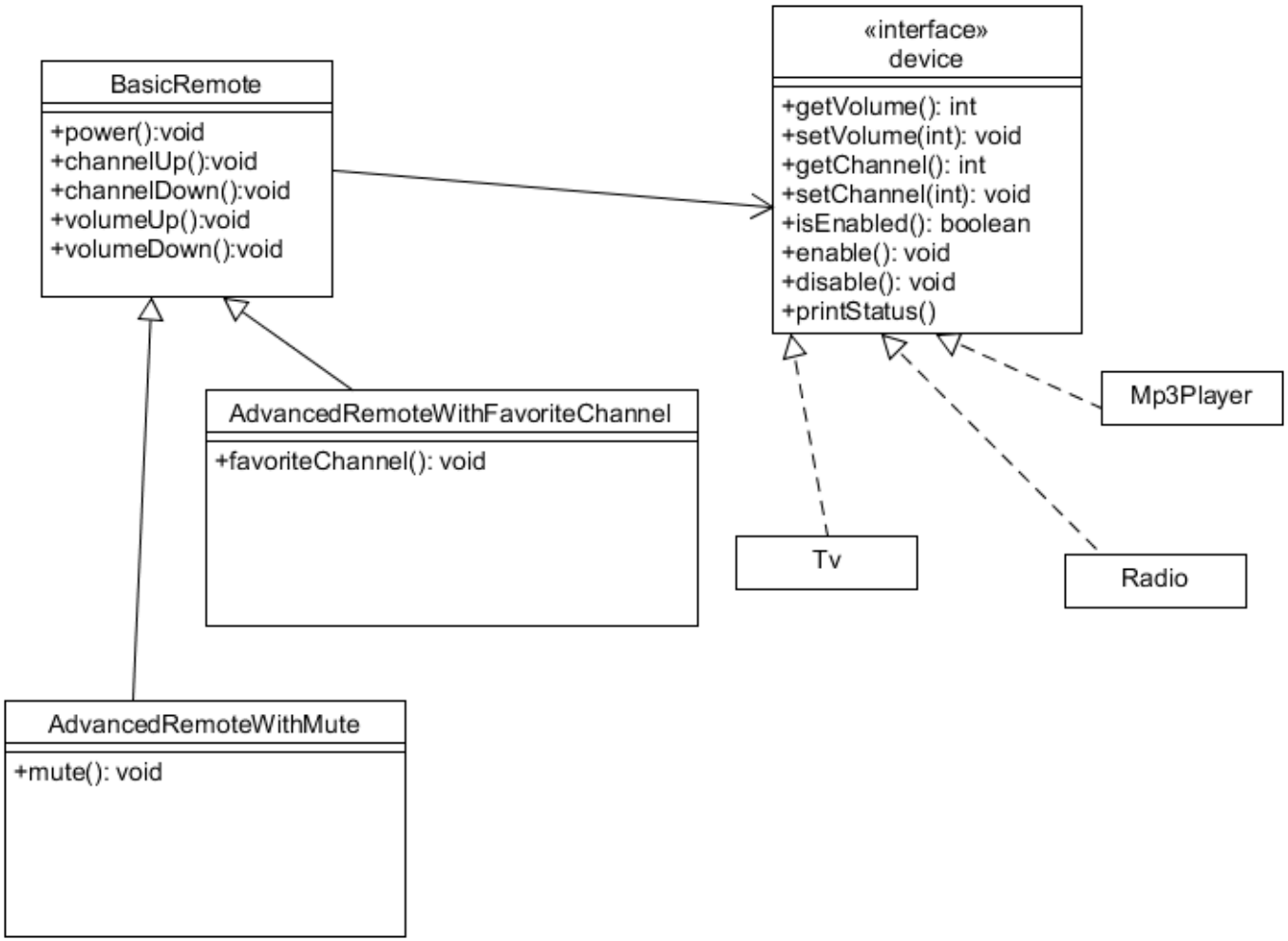
소스 코드

```
public class Remote {
    protected Device device;

    public Remote(Radio device) {
        this.device = device;
    }
…
```

```
public class Tv implements Device {
    private boolean on = false;
    private int volume = 30;
    private int channel = 1;
```

```
public class Radio implements Device {
    private boolean on = false;
    private int volume = 30;
    private int channel = 1;
…
```

고급 기능 추가

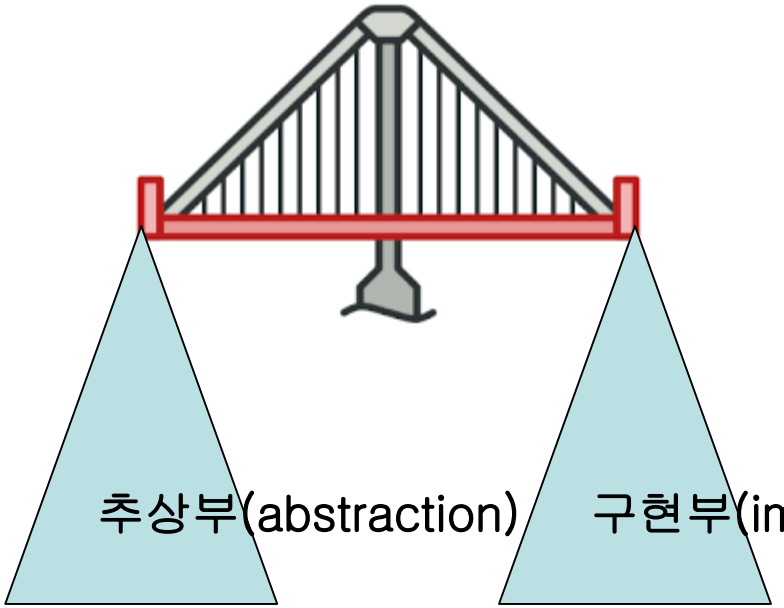

«interface»
device
+getVolume(): int
+setVolume(int): void
+getChannel(): int
+setChannel(int): void
+isEnabled(): boolean
+enable(): void
+disable(): void
+printStatus()
BasicRemote
+power():void
+channelUp():void
+channelDown():void
+volumeUp():void
+volumeDown():void
AdvancedRemoteWithFavoriteChannel
+favoriteChannel(): void
AdvancedRemoteWithMute
+mute(): void
Mp3Player
Tv
Radio

소스코드

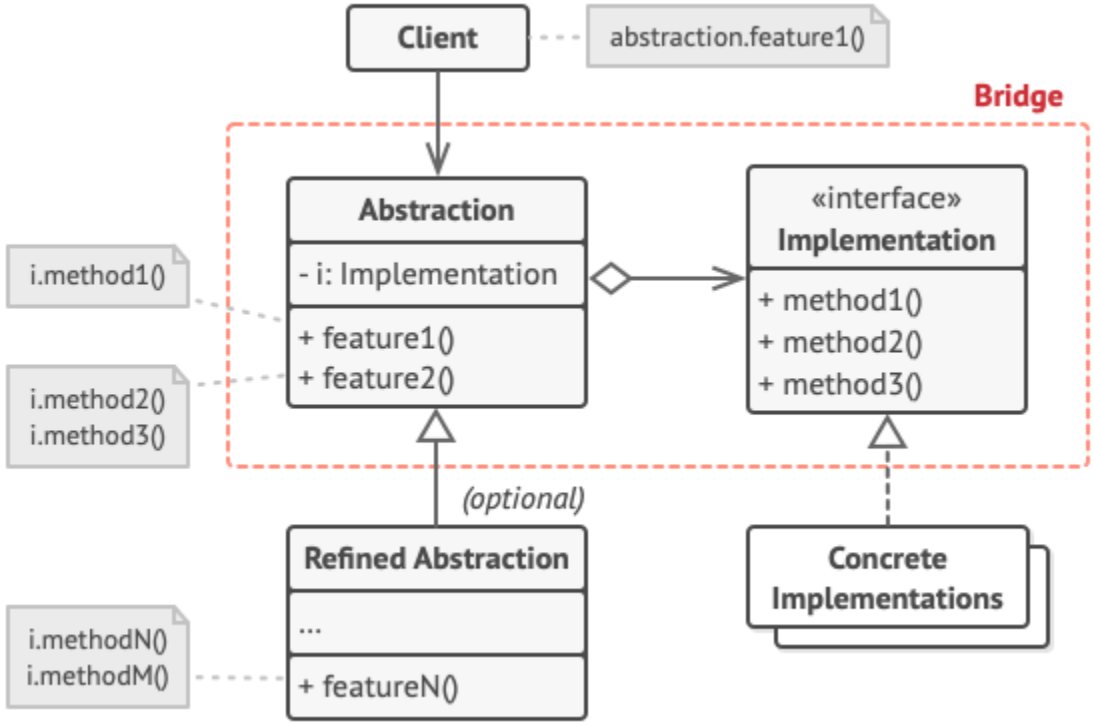
- E-class 첨부된 소스 코드 참조

Bridge pattern

- 추상부와 구현부를 분리하여 각각 독립적으로 변경할 수 있도록 하는 structural patten이다.
  - 추상부(abstraction): high-level function 제공.
  - 구현부(implementation): low-level function 제공
  - 추상부에서 제공하는 기능을 실현하기 위해 구현부로 위임(delegation)



브리지 패턴 구조


Client
abstraction.feature1()
Bridge
Abstraction
- i: Implementation
+ feature1()
+ feature2()
«interface»
Implementation
+ method1()
+ method2()
+ method3()
i.method1()
i.method2()
i.method3()
(optional)
Refined Abstraction
...
+ featureN()
Concrete
Implementations
i.methodN()
i.methodM()