Adapter Pattern

서로 다른 인터페이스(API)를 갖는 클래스들을 연결하여 바꿔서 재이용.

- 마치 220V 전원 110V 전원으로 개조시키는 것을 비유.
- '이미 제공된 API'와 '필요한 API' 사이의 '차이'를 없애주는 Design Pattern.
  - 기존에 제공된 API를 수정하지 않고 필요한 API에 맞추려는 Design Pattern.
    - 구 버전과의 호환성
  - 다른 용도로 감싸기때문에 'Wrapper' Pattern이라고도 불림.

Architecture, Modeling, DesignPattern, Structional

상속을 이용한 구현

Adaptee

```
/**
 * 이미 제공되는 API. 개조될 대상.
/*
public class OldVerAPI {
    private String str;

    public OldVerAPI(String str) {
        super();
        this.str = str;
    }
    public void showMessage() {
        System.out.println(str);
    }
}
```

Target, Adapter

```
/**
 * 필요한, 변환될 것. Target
  */
public interface NewVerAPI {
    void printMessage();
    void printVersion();
}
```

```
/**
```

```
 * 필요한, 변환될 것의 구현체. Adapter
 */
public class NewVerAPIImpl extends OldVerAPI implements NewVerAPI {

    public NewVerAPIImpl(String str) {
        super(str);
    }

    @Override
    public void printMessage() {
        showMessage();
    }

    @Override
    public void printVersion() {
        System.out.println("v1.0");
    }
}
```

Client

```
/**
 * Target 역할의 Method를 이용하여 수행.
 */
public class Main {
    public static void main(String[] args) {
        NewVerAPI p = new NewVerAPIImpl("test");

        p.printMessage();
        p.printVersion();
    }
}
```

위임을 이용한 구현

From:
http://ledyx.xyz/ - **Ledyx WIKI**

Permanent link:
http://ledyx.xyz/doku.php?id=design_pattern:adapter_pattern&rev=1509096762

Last update: **2021/02/07 03:15**