

Decorator Pattern

장식과 내용물을 동일시해서 장식을 여러 겹 중복되게 하기. (예를 들어 케이크에 초콜릿으로 코팅하면 초콜릿 케이크, 생크림으로 코팅하면 생크림 케이크)

- 장식이 되는 기능을 하나씩 추가하면서 좀 더 목적에 맞는 객체 만들기

[Architecture](#), [Modeling](#), [DesignPattern](#), [Structural](#)

Component

```
package component;
import java.util.stream.IntStream;

public abstract class Display {
    public abstract int getColumns();
    public abstract int getRows();
    public abstract String getRowText(int row);
    public final void show() {
        IntStream.range(0, getRows()).forEach(row ->
        System.out.println(getRowText(row)));
    }
}
```

ConcreteComponent

```
package component;

public class StringDisplay extends Display {
    private String str;

    public StringDisplay(String str) {
        this.str = str;
    }

    @Override
    public int getColumns() {
        return str.getBytes().length;
    }

    @Override
    public int getRows() {
        return 1;
    }
}
```

```
@Override
public String getRowText(int row) {
    return row == 0? str : null;
}
}
```

Decorator

```
package decorator;
import component.Display;

public abstract class Border extends Display {
    protected Display display; // 내용물
    public Border(Display display) {
        this.display = display;
    }
}
```

ConcreteDecorator

```
package decorator;

import java.util.stream.IntStream;
import component.Display;

public class FullBorder extends Border {

    public FullBorder(Display display) {
        super(display);
    }
    @Override
    public int getColumns() {
        return display.getColumns() + 2; // 양쪽 장식물 길이 더한 값
    }

    @Override
    public int getRows() {
        return display.getRows() + 2; // 상하 장식물 길이 더한 값
    }

    @Override
    public String getRowText(int row) {
        StringBuilder sb = new StringBuilder();
```

```

// 상하 사이인 경우
    if(0 < row && row < display.getRows() + 1) {
        sb.append('|');
        sb.append(display.getRowText(row - 1));
        sb.append('|');
    }
    else {
        sb.append('+');
        IntStream.range(0, display.getColumns()).forEach(i ->
sb.append(' '));
        sb.append('+');
    }
    return sb.toString();
}
}

```

Client

```

import component.Display;
import component.StringDisplay;
import decorator.FullBorder;
import decorator.SideBorder;

public class Main {
    public static void main(String[] args) {
        Display component = new StringDisplay("Hello world");
        Display decorator1 = new SideBorder(component, '#');
        Display decorator2 = new FullBorder(decorator1);
        component.show();
        decorator1.show();
        decorator2.show();
        System.out.println();
        Display composite = new SideBorder(
            new FullBorder(
                new FullBorder(
                    new SideBorder(
                        new FullBorder(
                            new StringDisplay("Hello world")
                        ) , '*'
                    ) , '/'
                ) , '/'
            ) , '/'
        );
        composite.show();
    }
}

/*
Hello world
#Hello world#

```

```
+-----+
|#Hello world#|
+-----+

/+-----+/
/|+-----+|/
/| |+*+-----+*| | /
/| |+*|Hello world|*| | /
/| |+*+-----+*| | /
/| |+-----+|/
/+-----+/
*/
```

From:
<http://wiki.ledyx.xyz/> - Ledyx WIKI

Permanent link:
http://wiki.ledyx.xyz/doku.php?id=design_pattern:decorator_pattern&rev=1509376690

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

