State Pattern

객체를 상태로 표현하여 그 상태를 변화시켜 행위를 변경.

Architecture, Modeling, Design Pattern, Behavioral

전등을 On, Off 상태로 전환한다고 가정.

State

```
public interface State {
    void turnOn(Light light);
    void turnOff(Light light);
}
```

ConcreteState

```
public enum OnState implements State {
    INSTANCE; // Singleton

    @Override
    public void turnOn(Light light) {
        System.out.println("Already turned on.");
    }

    @Override
    public void turnOff(Light light) {
        light.setState(OffState.INSTANCE);
        System.out.println("Turned off!");
    }
}
```

```
public enum OffState implements State {
    INSTANCE;

    @Override
    public void turnOn(Light light) {
        light.setState(OnState.INSTANCE);
        System.out.println("Turn on!");
    }
```

```
    @Override
    public void turnOff(Light clock) {
        System.out.println("Already turned off.");
    }
}
```

Context

State 전환을 하는 주체

```
public class Light {
    private State state;

    public Light() {
        setState(OffState.INSTANCE);
    }

    public void setState(State state) {
        this.state = state;
        System.out.println(state.getClass().getName());
    }

    public void turnOn() {
        this.state.turnOn(this);
    }

    public void turnOff() {
        this.state.turnOff(this);
    }
}
```

Client

```
public class Client {
    public static void main(String... args) {
        Light light = new Light();
        light.turnOff();
        light.turnOn();
        light.turnOn();
        light.turnOff();
    }
}
```

```
designpattern.flyweight.OffState
Already turned off.
designpattern.flyweight.OnState
Turn on!
Already turned on.
designpattern.flyweight.OffState
Turned off!
```

From:
https://wiki.ledyx.xyz/ - **Ledyx WIKI**

Permanent link:
https://wiki.ledyx.xyz/doku.php?id=design_pattern:state_pattern

Last update: **2021/02/07 03:29**