链滴

GOF 设计模式小白教程之状态模式

作者：valarchie
原文链接：https://ld246.com/article/1567428121994
来源网站：链滴

状态模式（State）

定义：

对有状态的对象，把复杂的“判断逻辑”提取到不同的状态对象中，允许状态对象在其内部状态发生变时改变其行为。

通俗解释：

玩过英雄联盟的同学都知道卡牌大师这个英雄。卡牌大师有一个技能是根据卡牌的颜色攻击对手的话会有不同的效果。这种效果体现在攻击力，耗蓝量，还有特殊效果。并且出牌颜色是随机的，所以这就有状态的流转。如果我们采用if-else方法的话，就需要在攻击方法，耗蓝量，以及特殊效果上依次用f-else判断当前的牌的颜色来写代码。但这就违反了开闭原则。我们应该将这些“业务逻辑”抽取出，放进一个一个的状态类当中。状态模式和策略模式很相像，状态模式主要不同在于对象内部状态的转，并且行为是随着状态自动改变的。而策略模式是通过动态的切换策略来更改当前类的行为。

代码：

抽象状态类：卡牌类，拥有攻击力，耗蓝量，特殊效果等方法。

```
public abstract class Card {
    // 攻击力
    public abstract void attack();
    // 消耗蓝量
    public abstract void consumeMana();
    // 特殊效果
    public abstract void effect();

}
```

具体状态类：黄牌、红牌、蓝牌各自有不同的实现

```
public class YellowCard extends Card {
    @Override
    public void attack() {
        System.out.println("黄牌攻击力：30！");
    }

    @Override
    public void consumeMana() {
        System.out.println("黄牌消耗魔法值：20！");
    }

    @Override
    public void effect() {
        System.out.println("黄牌造成敌人眩晕！");
    }
}

public class BlueCard extends Card {
```

```
    @Override
    public void attack() {
        System.out.println("蓝牌攻击力：10！");
    }

    @Override
    public void consumeMana() {
        System.out.println("蓝牌消耗魔法值：10！");
    }

    @Override
    public void effect() {
        System.out.println("蓝牌回复魔法值50！");
    }
}

public class RedCard extends Card {
    @Override
    public void attack() {
        System.out.println("红牌攻击力：20！");
    }

    @Override
    public void consumeMana() {
        System.out.println("红牌消耗魔法值：30！");
    }

    @Override
    public void effect() {
        System.out.println("红牌造成范围伤害！");
    }
}
```

环境角色：卡牌大师类，拥有甩牌技能和切牌技能。每发一张牌，下张牌的颜色就会切换，相当于状的流转。

```
public class CardMaster {

    private Card[] cards = {new YellowCard(), new RedCard(), new BlueCard()};
    private Card currentCard = cards[0];
    // 发动卡牌技能
    public void flushCard() {
        System.out.println("+++发动卡牌技能！+++");

        currentCard.attack();
        currentCard.consumeMana();
        currentCard.effect();

        swichCard();

    }
    // 每次发完牌之后，卡牌颜色会随机变换
    public void swichCard() {
```

```
        Random random = new Random();
        int index = random.nextInt(3);
        currentCard = cards[index];

    }

}
```

测试状态模式

```
public class TestState {
    public static void main(String[] args) {

        CardMaster cardMaster = new CardMaster();

        cardMaster.flushCard();
        cardMaster.flushCard();
        cardMaster.flushCard();
        cardMaster.flushCard();
        cardMaster.flushCard();

    }

}
```

运行结果：重复调用甩牌技能，会根据不同颜色的牌进行不同攻击。

```
+++发动卡牌技能！+++
黄牌攻击力：30！
黄牌消耗魔法值：20！
黄牌造成敌人眩晕！
+++发动卡牌技能！+++
红牌攻击力：20！
红牌消耗魔法值：30！
红牌造成范围伤害！
+++发动卡牌技能！+++
黄牌攻击力：30！
黄牌消耗魔法值：20！
黄牌造成敌人眩晕！
+++发动卡牌技能！+++
蓝牌攻击力：10！
蓝牌消耗魔法值：10！
蓝牌回复魔法值50！
```

解析：

1. 状态模式将与特定状态相关的行为局部化到一个状态中，并且将不同状态的行为分割开来，满足“一职责原则”。

2. 减少对象间的相互依赖。将不同的状态引入独立的对象中会使得状态转换变得更加明确，且减少对间的相互依赖。

3. 有利于程序的扩展。通过定义新的子类很容易地增加新的状态和转换。