链滴

GOF 设计模式小白教程之策略模式

作者：valarchie

原文链接：https://ld246.com/article/1567431734849

来源网站：链滴

策略模式（Strategy）

定义：

该模式定义了一系列算法，并将每个算法封装起来，使它们可以相互替换，且算法的变化不会影响使算法的客户。策略模式属于对象行为模式，它通过对算法进行封装，把使用算法的责任和算法的实现割开来，并委派给不同的对象对这些算法进行管理。

通俗解释：

玩过英雄联盟的小伙伴都知道有野兽之灵乌迪尔这个英雄，这个英雄拥有不同的形态，例如熊形态，形态，老虎形态。如果代码编写成if-else，通过判断野兽之灵是什么形态来进行什么样的攻击的话，期游戏想继续增加新的形态的话，就会破坏原有的代码。所以我们必须遵循开闭原则，通过将形态抽出来进行解耦。在JDK当中的集合排序，通过传入不同的比较器来进行比较，也是策略模式的体现。

代码：

抽象策略类：拥有攻击的方法

```
public interface Strategy {
    // 攻击
    void attack();

}
```

具体策略类：三种不同的形态

```
public class Tiger implements Strategy {
    @Override
    public void attack() {
        System.out.println("使用老虎形态进行攻击！");
    }
}


public class Tortoise implements Strategy{
    @Override
    public void attack() {
        System.out.println("使用乌龟形态进行攻击！");
    }
}

public class Wolf implements Strategy {
    @Override
    public void attack() {
        System.out.println("使用狼人形态进行攻击！");
    }
}
```

环境角色类：野兽之灵，持有策略对象（动物形态）

```
public class AnimalSoul {
```

```
    // 当前的形态
    private Strategy animal;
    // 使用形态进行攻击
    public void animalAttack() {
        animal.attack();
    }

    public Strategy getStrategy() {
        return animal;
    }

    public void setStrategy(Strategy strategy) {
        this.animal = strategy;
    }
}
```

测试策略模式

```
public class TestStrategy {

    public static void main(String[] args) {

        AnimalSoul animalSoul = new AnimalSoul();
        // 切换老虎形态
        animalSoul.setStrategy(new Tiger());
        animalSoul.animalAttack();
        // 切换乌龟形态
        animalSoul.setStrategy(new Tortoise());
        animalSoul.animalAttack();
        // 切换狼人形态
        animalSoul.setStrategy(new Wolf());
        animalSoul.animalAttack();

    }

}
```

运行结果：

```
使用老虎形态进行攻击!
使用乌龟形态进行攻击!
使用狼人形态进行攻击!
```

解析：

1. 算法可以自由切换。
2. 避免使用多重条件判断。
3. 扩展性良好，遵循开闭原则。