转载请注明链接：https://blog.csdn.net/feather_wch/article/details/50281655

> 本文以面试题的形式归纳总结，可以直接学习和背诵。
> 鸣谢: 本文基础部分归纳总结自《Head First 设计模式》

如果有帮助，请点个赞，万分感谢！

外观模式

版本：2018/8/24-1(23:24)

- 外观模式
  - 介绍
  - 特点
  - 实例
  - 设计原则
  - 外观和适配器

介绍

1、外观模式是什么?

> 1. 外观模式-Facade Pattern
> 2. 外观模式的作用就是让一个接口更简单。
> 3. 使用组合实现外观模式。

2、外观模式的定义

> 1. 外观模式给子系统的一系列接口提供了一个统一的接口。
> 2. 外观模式定义了一个高层接口，让子系统更容易使用。

特点

3、外观模式的优点和缺点?

> 1. 能提供全部的功能(能直接操纵底层的功能)，也提供了一个简化的接口。
> 2. 将用户实现和底层系统解耦。
> 3. 解耦，减少系统维护成本
> 4. 缺点：创造了更多的包装类，会增加复杂度，开发时间和运行性能。

实例

4、外观模式实例

> 内容：家庭影院要播放电影，需要灯光，幕布，DVD等各种设备的开关，非常复杂。这里就利用Facade模式来实现家庭影院。
>
> Device设备:

```
public class DvdPlayer {
        public void on() {
                System.out.println("DVD on");
        }
        public void off() {
                System.out.println("DVD off");
        }
        public void play(String movie) {
                System.out.println("DVD is playing " + movie);
        }
}
public class Light {
        public void on() {
                System.out.println("Light on");
        }
        public void lightToRed() {
                System.out.println("Light red");
        }
        public void off() {
                System.out.println("Light off");
        }
}
```

> 家庭影院(外观)：HomeTheaterFacade

```
public class HomeTheaterFacade {
        DvdPlayer dvdPlayer;
        Light light;
        public HomeTheaterFacade(DvdPlayer dvdPlayer, Light light) {
                this.dvdPlayer = dvdPlayer;
                this.light = light;
        }
    //观看电影
        public void watchMovie(String movie) {
                light.on();
                light.lightToRed();
                dvdPlayer.on();
                dvdPlayer.play(movie);
        }
        //结束电影
        public void endMovie() {
                light.off();
                dvdPlayer.off();
        }
}
```

> 测试：电影播放

```
HomeTheaterFacade homeTheaterFacade = new HomeTheaterFacade(new DvdPlayer(), new Light());
// 这样通过外观模式提供的简单接口，完成了复杂的功能。
homeTheaterFacade.watchMovie("Harry Potter");
homeTheaterFacade.endMovie();
```

设计原则

5、最少知识原则是什么?

> 只和你的密友谈话-要减少对象之间的交互，只留下几个“密友”。

6、如何遵循最少知识原则

> 1. 不影响太多的对象
> 2. 将方法调用保持在界限内

7、如何不影响太多的对象？(可以调用的方法)

> 1. 该对象本身，的方法
> 2. 作为参数传入方法的对象，的方法
> 3. 该方法创建的任何对象，的方法
> 4. 该对象任何组件，的方法

8、哪种方法调用影响了设计模式?

> 如果某对象是调用其他对象返回的结果，不要调用该对象的方法。

9、符合最少知识原则实例

```
public class Car{
        Engine engine;
        public void start(Key key) {
                Doors doors = new Doors();
                boolean authorized = key.turns(); //可以调用参数传入对象的方法
                if(authorized) {
                        engine.start(); //可以调用内部对象的方法
                        updateDisplay(); //可以调用本地方法
                        doors.lock(); //可以调用该方法内部创建的Doors对象的方法
                }
        }
        public void updateDisplay() {
                //更新
        }
}
```

10、Java中System.out.println就违反了该原则

外观和适配器

11、Facade和Adapter的区别

> 1. Adapter的目的是将一个接口转换为用户需要的接口。
> 2. Facade的目的是提供一个简化的接口。