适配器模式与外观模式

# 适配器模式

将一个类的接口，转换成客户期望的另一个接口。

## 适配器模式的使用场合：

当需要使用一个现有的类而其接口并不符合你的需要时，就使用适配器模式。

## 适配器模式的意图：

将一个接口转换成另一个接口，即通过改变接口来符合客户的期望。

## 适配器模式的复杂度

实现一个适配器，可能需要一番功夫，也可能不费功夫，视**目标接口的大小与复杂度**而定。

## 适配器模式有两种形式

### 对象适配器

### 类适配器

类适配器需要**多重继承**，在java中无法实现。Java只允许单继承。

# 外观模式

## 使用场合

当需要简化并统一一个很大的接口或者一群复杂的接口时，使用外观。

## 外观模式的意图

将客户从一个复杂的子系统中解耦出来，统一和简化接口。

## 外观模式的实现

实现一个外观，需要将子系统组合进外观中，然后将工作委托给子系统进行。

可以为一个子系统实现一个以上的外观。

# 适配器模式、装饰者模式、外观模式区别

适配器模式：将一个接口转成另一个接口。（广义的接口）

装饰者模式：不改变接口，但是会增加功能。

外观模式：让接口更简单。

适配器的意图是要转换接口，而外观的意图是要统一和简化接口。

**适配器将一个对象包装起来以改变其接口；装饰着将一个对象包装起来以增加新的行为和责任；而外观将一群对象“包装”起来以简化其接口。**

# 示例：

## 适配器模式的例子：

三脚插头如何与两脚插头连接。显然需要利用适配器。

## 外观模式的例子

家庭的智能遥控器：可以控制空调、电视、冰箱、灯光等。如果需要单独控制，需要利用命令模式，但是如果想一键开启或关闭所有电器，可以利用外观模式。外观将所有的对象封装起来，一起执行。