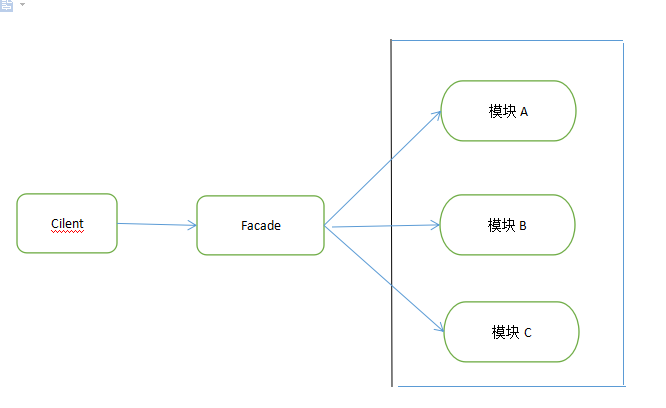
## [java设计模式之外观模式（门面模式）](https://www.cnblogs.com/lthIU/p/5860607.html)

　　针对外观模式，在项目开发和实际运用中十分频繁，但是其极易理解，下面就简要介绍一下。

## **一、概念介绍**

　　外观模式（Facade）,他隐藏了系统的复杂性，并向客户端提供了一个可以访问系统的接口。这种类型的设计模式属于结构性模式。为子系统中的一组接口提供了一个统一的访问接口，这个接口使得子系统更容易被访问或者使用。

## **二、角色及使用场景**



　　简单来说，该模式就是把一些复杂的流程封装成一个接口供给外部用户更简单的使用。这个模式中，设计到3个角色。

　　1）.门面角色：外观模式的核心。它被客户角色调用，它熟悉子系统的功能。内部根据客户角色的需求预定了几种功能的组合。

　　2）.子系统角色:实现了子系统的功能。它对客户角色和Facade时未知的。它内部可以有系统内的相互交互，也可以由供外界调用的接口。

　　3）.客户角色:通过调用Facede来完成要实现的功能。

### **使用场景：**

　　1- 为复杂的模块或子系统提供外界访问的模块；

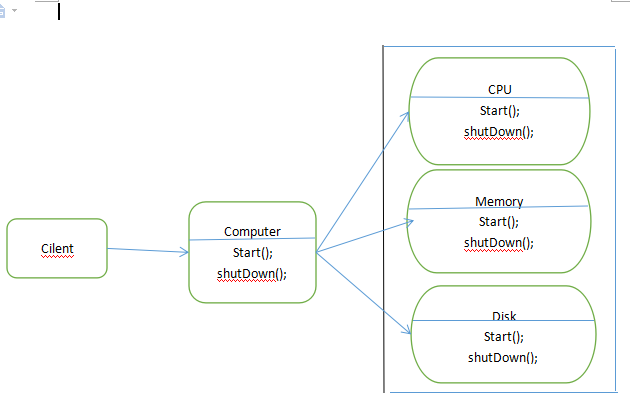
　　2- 子系统相互独立；

　　3- 在层析结构中，可以使用外观模式定义系统的每一层的入口。

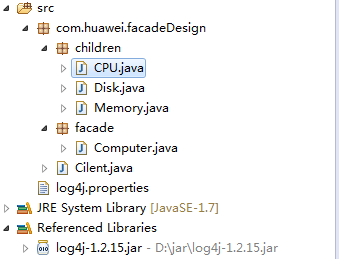
## **三、实例**

　　下面，我们就通过一个简单的例子来实现该模式。

　　每个Computer都有CPU、Memory、Disk。在Computer开启和关闭的时候，相应的部件也会开启和关闭，所以，使用了该外观模式后，会使用户和部件之间解耦。如：



#### **包体的创建：**



**代码实现**

首先是子系统类：

成。客户端只需要跟Facade交互就可以了。

## **四、优点**

### **- 松散耦合**

　　使得客户端和子系统之间解耦，让子系统内部的模块功能更容易扩展和维护；

### **- 简单易用**

　　客户端根本不需要知道子系统内部的实现，或者根本不需要知道子系统内部的构成，它只需要跟Facade类交互即可。

### **- 更好的划分访问层次**

　　有些方法是对系统外的，有些方法是系统内部相互交互的使用的。子系统把那些暴露给外部的功能集中到门面中，这样就可以实现客户端的使用，很好的隐藏了子系统内部的细节。

## **五、延伸及应用（补充）**

　　...