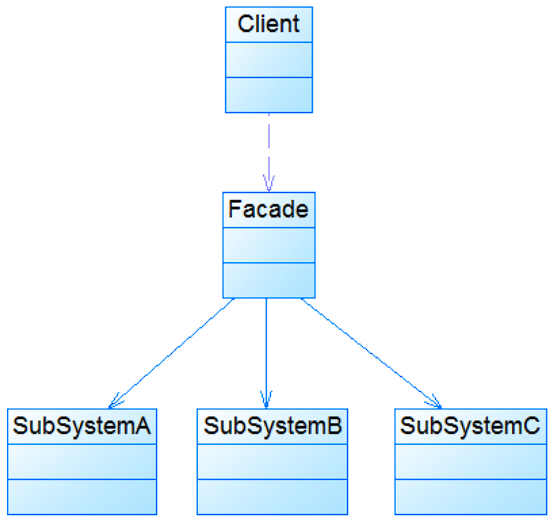
1. 模式定义

外观模式（FacadePattern)：外部与一个子系统的通信必须通过一个统一的外观对象进行，为子系统中的一组接口提供一个一致的界面，外观模式定义了一个高层接口，这个接口使得这一子系统更加容易使用。外观模式又称为门面模式，它是一种对象结构型模式。



**二、**模式动机

引入外观角色之后，用户只需要直接与外观角色交互，用户与子系统之间的复杂关系由外观角色来实现，从而降低了系统的祸合度。

三、例子

网站系统的首页与子栏目

四、优缺点

优点：

·对客户屏蔽子系统组件，减少了客户处理的对象数目并使得子系统使用起来更加容易。通过引入外观模式，客户代码将变得很简单，与之关联的对象也很少。

·实现了子系统与客户之间的松藕合关系，这使得子系统的组件变化不会影响到调用它的客户类，只需要调整外观类即可。

·降低了大型软件系统中的编译依赖性，并简化了系统在不同平台之间的移植过程，因为编译一个子系统一般不需要编译所有其他的子系统。一个子系统的修改对其他子系统没有任何影响，而且子系统内部变化也不会影响到外观对象。

·只是提供了一个访问子系统的统一入口，并不影响用户直接使用子系统类。

缺点：

·不能很好地限制客户使用子系统类，如果对客户访问子系统类做太多的限制则减少了可变性和灵活性。

·在不引入抽象外观类的情况下，增加新的子系统可能需要修改外观类或客户端的源代码，违背了“开闭原则”。

五、适用环境

·当要为一个复杂子系统提供一个简单接口时可以使用外观模式。该接口可以满足大多数用户的需求，而且用户也可以越过外观类直接访问子系统。

·客户程序与多个子系统之间存在很大的依赖性。引入外观类将子系统与客户以及其他子系统解藕，可以提高子系统的独立性和可移植性。

·在层次化结构中，可以使用外观模式定义系统中每一层的入口，层与层之间不直接产生联系，而通过外观类建立联系，降低层之间的藕合度。

六、示例代码