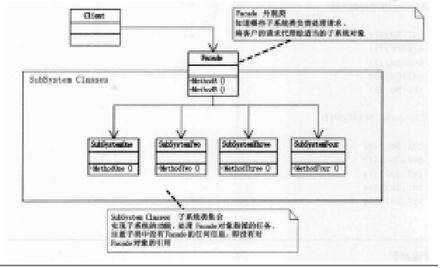
Facade

- 定义
  - 为子系统中的一组接口提供一个一致的界面，此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。
- UML图
  - 

- 代码示例
- vs 装饰模式、适配器模式、代理模式
  - 1）适配器模式强调的是转换接口。举例：JDBC是java定义的数据库操作规范，其已经定义好了操作数据库的接口，假设一个数据库厂商提供了一套操作数据库的接口，但是该接口并不符合JDBC的规范，也就是说JDBC规范规定的接口和数据库厂商提供的接口不一致。这个时候就可以使用适配器模式将数据库厂商的接口转换成JDBC规范要求的接口。其他三种模式均不提供接口转换的功能。

    2）外观模式强调的是包装多个对象，以简化他们的接口。想象这样一种场景：你家的电器都是智能控制的，当你回家的时候你要打开空调、打开电视、打开热水器等等，这些都需要你自己一个个的去操作，这个时候你就会有这样一种需求，就是当你到家的时候只要进行一个操作，就能依次打开空调、电视、热水器等。这个时候就可以使用外观模式将空调、电视、热水器等的接口进行包装，只对外提供一个按钮。其他三种模式均强调的是对一个对象的包装。

    3）装饰者模式强调的是为被装饰对象增加额外的行为。举例：java.io包。

    4）代理模式强调的是对被代理对象进行控制。这些控制体现在很多方面，比如安全、权限控制等。

    适配器模式和外观模式容易和其他模式进行区分。下面重点比较装饰者模式和代理模式。

    5）装饰者模式和外观模式主要的区别就是，装饰者模式从来不创建被装饰的对象，它总是添加新功能到已经存在的对象上面；而代理模式在被代理对象不存在的时候会创建被代理对象。

    6）装饰者模式可以通过嵌套装饰添加多重额外功能，而代理模式一般不推荐使用嵌套代理。