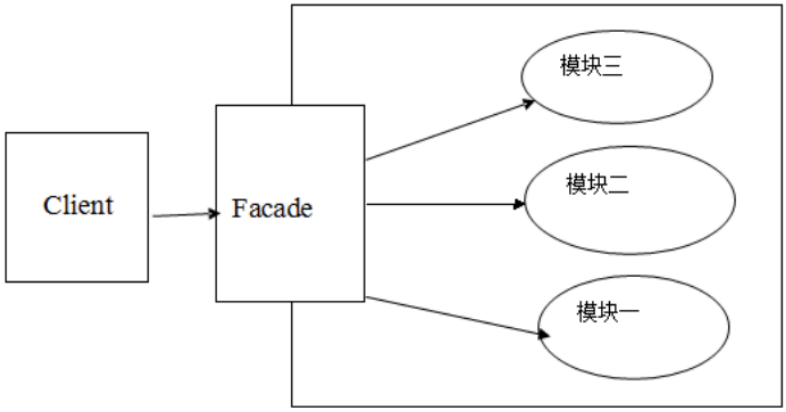
定义:将子系统中的一组接口提供一个一致的界面(稳定)，外观模式定义了一个高层接口，这个接口使得这一子系统更加容易使用(复用)。GOF



要点总结: 从客户程序角度看,外观模式简化了整个组件系统的接口,对于组件内部与外部客户程序来说,达到了一种"解耦"的效果. 内部子系统的任何变化不影响Facade接口的变化. 外观模式更注重从架构的层次去看整个系统. 而不是单个类的层次.

外观模式的主要用途就是为子系统的复杂处理过程提供方便的调用方法，使得子系统更加容易被使用。Façade对象通常属于单实例模式。

```
#include<iostream>
using namespace std;
class Shop
{
public:
    void BuyFood(){
        cout << "买菜" << endl;
    }
};
class Cook
{
public:
    void doCook(){
        cout << "做菜" << endl;
    }
};
class Clean{
public:
    void doClean(){
        cout << "晚餐完毕,收拾餐盘,打扫卫生" << endl;
    }
};
class Facade_HaveDinner{
public:
    Shop objShop;
    Cook objCook;
    Clean objClean;
public:
    void doHaveDinner(){
        cout << "下午啦,开始准备晚餐" << endl;
        objShop.BuyFood();
        objCook.doCook();
        objClean.doClean();
```

```
    }
};
int main(){
    Facade_HaveDinner objHavedinner;
    objHavedinner.doHaveDinner();
    return 0;
}
```