**设计模式概括**

# 设计模式概括

# 设计模式分类

# 设计模式详细讲解

## 简单工厂

### 定义

### 结构和说明

## 外观模式

### 定义

为子系统中的一组接口提供一个一致的界面，Facade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。

### 结构和说明



### 优缺点

优点：

* 松散耦合

外观模式松散了客户端与子系统的耦合关系，让子系统内部的模块能更容易扩展和维护

* 简单易用

外观模式让子系统更加易用，客户端不再需要了解子系统内部的实现，也不需要跟众多子系统内部的模块进行交互，只需要跟外观交互就可以了，相当于外观类为外部客户使用子系统提供了一站式服务。

* 更好地划分访问的层次

通过合理使用Façade，可以帮助我们更好地划分访问的层次。有些方法是对系统外的，有些方法是系统内部使用的。把需要暴露给外部的功能集中到外观中，这样既方便客户端使用，也很好地隐藏了内部的细节。

缺点：

* 过多的或者是不太合理的Façade也容易让人迷惑。到底是调用Façade好呢，还是直接调用模块好。

### 思考外观模式

* 本质

外观模式的本质是：封装交互，简化调用

* 对设计原则的体现

最少知识原则

* 何时选用外观模式
  + 如果你希望为一个复杂的子系统提供一个简单接口的时候，可以考虑使用外观模式。使用外观对象来实现大部分客户需要的功能，从而简化客户的使用。
  + 如果想要让客户程序和抽象类的实现部分松散耦合，可以考虑使用外观模式，使用外观对象来将这个子系统与它的客户分离开来，从而提高子系统的独立性和可移植性。
  + 如果构建多层结构的系统，可以考虑使用外观模式，使用外观对象作为每层的入口，这样可以简化层间调用，也可以松散层次之间的依赖关系。

### 相关模式

* 中介者模式
* 单例模式
* 抽象工厂

## 适配器模式

### 定义

### 结构和说明

### 优缺点

### 思考适配器模式

### 相关模式