OOAD第十一章

1. 职责驱动设计（Responsibility-Driven Design）

   对象的认知职责（Knowing responsibilities）包括：

   对私有封装数据的认知 knowing about private encapsulated data

   对相关对象的认知 knowing about related objects

   对其能够导出或计算的事物的认知 knowing about things it can derive or calculate

   对象的行为职责（Doing responsibilities）包括：

   初始化其他对象中的动作 initiating action in other objects

   控制和协调其他对象中的活动 controlling and coordinating activities in other objects

   自身执行一些行为，如创建或计算 doing something itself, such as creating an object or doing a calculation

2. GoF设计模式

   包含23种模式

   创建型

   Abstract Factory（抽象工厂）、Singleton（单实例）

   结构型

   Adapter（适配器）、Composite（组合）、Facade（外观）

   行为型

   Observer（观察者）、Strategy（策略）

适配器

通过中介适配器对象，将构件的原有接口转换为其他接口

适配器模式的宗旨是，保留现有类提供的服务，向客户提供接口，以满足客户的期望

对于一个已经存在的类，如果它的接口与现有系统的需求不同时，可以考虑用适配器模式

简单工厂

是GoF抽象工厂（Abstract Factory）模式的简化

问题：当有特殊考虑时，应该由谁来负责创建对象？

解决方案：创建称为工厂的纯虚构对象来处理这些创建职责

分离复杂创建的职责，并将其分配给内聚的帮助者对象

单实例类（Singleton）

问题：如何只创建唯一实例的类即“单实例类”？如何使对象具有全局可见性并且单点访问？

解决方案：对类定义静态方法用以返回单实例。

单实例类只能有一个实例

单实例类必须自己创建自己的唯一实例

单实例类必须给所有其他对象提供这一实例

策略

问题：如何设计变化但相关的算法或政策，才能使这些算法或政策具有可变的能力?

解决方案：在单独的类中分别定义每种算法/政策/策略，并且使其具有共同接口

策略模式对应于解决某一个问题的一个算法族，允许用户从该算法族中选择一个算法解决一个问题

本质——分离算法，选择实现

组合

问题：如何能够像处理非组合（原子）对象一样，（多态地）处理一组对象或具有组合结构的对象呢?

解决方案：定义组合和原子对象的类，使它们实现相同的接口

组合模式希望用户可以忽略组合与单个对象的不同，统一地使用组合结构中的所有对象

外观

问题：为了降低复杂性，常常将系统划分为若干个子系统。如何做到各个子系统之间的通信和相互依赖关系达到最小?

解决方案：对子系统定义唯一的接触点——使用外观对象封装子系统。该外观对象提供了唯一和统一的接口，并负责与子系统构件进行协作。

外观模式为子系统的一组接口提供一个一致的界面，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

观察者

观察者模式定义了一种一对多的依赖关系，让多个观察者对象同时监听某一个主题对象。这个主题对象在状态发生变化时，会通知所有观察者对象，使它们能够自动更新自己

3. 状态机图

结构

状态（state）是指对象在事件发生之间某时刻所处的情形

事件（event）是指一件值得注意的事情的发生

转换（transition）是两个状态之间的关系。它表明当某事件发生时，对象从先前的状态转换到后来的状态

使用原则

状态无关和状态依赖对象

如果一个对象对某事件的响应总相同，则认为此对象对于该事件状态无关

如果对于所有事件，对象的响应总是相同的，则该对象是一个状态无关对象

状态依赖对象是指对事件的响应根据对象的状态或模式而不同

考虑为具有复杂行为的状态依赖对象而不是状态无关对象建立状态机图