



LANZHOU UNIVERSITY OF TECHNOLOGY

设计模式总结

报告完成人： 侯嘉欣

学号： 1817700154

专业班级： 软件1班

学 院： 软件学院

指导教师： 马维俊

目录

**[1、简单工厂模式 3](#_Toc2698)**

[1.1简单工厂模式的优点: 3](#_Toc17652)

[1.2简单工厂模式的缺点: 3](#_Toc15125)

**[2、策略模式 3](#_Toc28742)**

[2.1策略模式的优点： 3](#_Toc13027)

[2.2策略模式的缺点： 4](#_Toc2477)

**[3、装饰模式 4](#_Toc6619)**

[3.1装饰模式的优点： 4](#_Toc6749)

[3.2装饰模式的缺点 4](#_Toc31138)

**[4、代理模式 4](#_Toc7587)**

[4.1代理模式的优点 4](#_Toc32679)

[4.2代理模式的缺点 4](#_Toc30009)

**[5、工厂方法模式 4](#_Toc7587)**

[5.1工厂方法模式的优点 4](#_Toc32679)

[5.2工厂方法模式的缺点 4](#_Toc30009)

**[6、原型模式 4](#_Toc7587)**

[6.1原型模式的优点 5](#_Toc32679)

[6.2原型模式的缺点 5](#_Toc30009)

**[7、建造者模式 5](#_Toc7587)**

[7.1建造者模式的优点 5](#_Toc32679)

[7.2建造者模式的缺点 5](#_Toc30009)

**[8、观察者模式 5](#_Toc7587)**

[8.1观察者模式的优点 5](#_Toc32679)

[8.2观察者模式的缺点 5](#_Toc30009)

**[9、 门面模式 5](#_Toc6891)**

[9.1门面模式的优点 5](#_Toc30386)

[9.2门面模式的缺点 6](#_Toc20026)

**[10、 抽象工厂模式 6](#_Toc19129)**

[10.1抽象工厂模式的优点： 6](#_Toc16611)

[10.2抽象工厂模式的缺点： 6](#_Toc31905)

**1、简单工厂模式**

简单工厂模式，也就是说，到底要实例化谁，将来会不会增加实例化对象，比如增加开根运算，这是很容易变化的地方，应该考虑用一个单独的类来做这个创造实例的过程，这就是工厂。

**1.1简单工厂模式的优点:**

（1）工厂类含有必要的判断逻辑，可以决定在什么时候创建哪一个产品类的实例，客户端可以免除直接创建产品对象的责任，而仅仅“消费”产品；

（2）简单工厂模式通过这种做法实现了对责任的分割，它提供了专门的工厂类用于创建对象。

（3）客户端无须知道所创建的具体产品类的类名，只需要知道具体产品类所对应的参数即可，对于一些复杂的类名，通过简单工厂模式可以减少使用者的记忆量。

（4）通过引入配置文件，可以在不修改任何客户端代码的情况下更换和增加新的具体产品类，在一定程度上提高了系统的灵活性。

**1.2简单工厂模式的缺点:**

（1）由于工厂类集中了所有产品创建逻辑，一旦不能正常工作，整个系统都要受到影响。

（2）使用简单工厂模式将会增加系统中类的个数，在一定程序上增加了系统的复杂度和理解难度。

（3）系统扩展困难，一旦添加新产品就不得不修改工厂逻辑，在产品类型较多时，有可能造成工厂逻辑过于复杂，不利于系统的扩展和维护。4、简单工厂模式由于使用了静态工厂方法，造成工厂角色无法形成基于继承的等级结构。

**2、策略模式**

它定义了算法家族，分别封装起来，让它们之间可以互相替换，此模式让算法的变化，不会影响到使用算法的客户。

**2.1策略模式的优点：**

（1） 策略模式提供了管理相关的算法族的办法。策略类的等级结构定义了一个算法或行为族。恰当使用继承可以把公共的代码转移到父类里面，从而避免重复的代码。

（2） 策略模式提供了可以替换继承关系的办法。继承可以处理多种算法或行为。如果不是用策略模式，那么使用算法或行为的环境类就可能会有一些子类，每一个子类提供一个不同的算法或行为。但是，这样一来算法或行为的使用者就和算法或行为本身混在一起。决定使用哪一种算法或采取哪一种行为的逻辑就和算法或行为的逻辑混合在一起，从而不可能再独立演化。继承使得动态改变算法或行为变得不可能。

（3） 使用策略模式可以避免使用多重条件转移语句。多重转移语句不易维护，它把采取哪一种算法或采取哪一种行为的逻辑与算法或行为的逻辑混合在一起，统统列在一个多重转移语句里面，比使用继承的办法还要原始和落后。

**2.2策略模式的缺点：**

（1）客户端必须知道所有的策略类，并自行决定使用哪一个策略类。这就意味着客户端必须理解这些算法的区别，以便适时选择恰当的算法类。换言之，策略模式只适用于客户端知道所有的算法或行为的情况。

（2）策略模式造成很多的策略类，每个具体策略类都会产生一个新类。有时候可以通过把依赖于环境的状态保存到客户端里面，而将策略类设计成可共享的，这样策略类实例可以被不同客户端使用。换言之，可以使用享元模式来减少对象的数量。

**3、装饰模式**

装饰模式（Decorator）：动态地给一个对象添加一些额外的职责，就增加功能来说，装饰模式比生成子类更为灵活

**3.1装饰模式的优点：**

（1）装饰模式与继承关系的目的都是要扩展对象的功能，但是装饰模式可以提供比继承更多的灵活性。装饰模式允许系统动态决定“贴上”一个需要的“装饰”，或者除掉一个不需要的“装饰”。继承关系则不同，继承关系是静态的，它在系统运行前就决定了。

（2）通过使用不同的具体装饰类以及这些装饰类的排列组合，设计师可以创造出很多不同行为的组合。

## **3.2装饰模式的缺点**

由于使用装饰模式，可以比使用继承关系需要较少数目的类。使用较少的类，当然使设计比较易于进行。但是，在另一方面，使用装饰模式会产生比使用继承关系更多的对象。更多的对象会使得查错变得困难，特别是这些对象看上去都很相像。

**4、代理模式**

代理模式（Proxy），为其他对象提供一种代理以控制对这个对象的访问。

**4.1代理模式的优点**

(1)职责清晰，真实的角色就是实现实际的[业务逻辑](https://baike.baidu.com/item/%E4%B8%9A%E5%8A%A1%E9%80%BB%E8%BE%91" \t "https://baike.baidu.com/item/%E4%BB%A3%E7%90%86%E6%A8%A1%E5%BC%8F/_blank)，不用关心其他非本职责的事务，通过后期的代理完成一件完成事务，附带的结果就是编程简洁清晰。

(2)代理对象可以在客户端和目标对象之间起到中介的作用，这样起到了中介的作用和保护了目标对象的作用。

(3)高扩展性

**4.2代理模式的缺点**

静态代理虽然效率较高，但其也有不可避免的缺陷。它需要创建多个代理类的问题，还有需要被代理对象实现某个接口的问题。

**5、工厂方法模式**

工厂方法模式（Factory Method），定义一个用于创建对象的接口，让子类决定实例化哪一个类。工厂方法使一个类的实例化延迟到其子类。满足了ocp原则，增加新的类需要修建新的工厂，增加了代码量，如果同时需要修改多个工厂类的时候会很麻烦，而简单工厂模式只需要修改一个类，工厂方法模式是升级版的简单工厂模式

**5.1工厂方法模式的优点：**

易于交换产品系列；使具体的创建实例过程与客户端分离   
**5.2工厂方法模式的缺点：**

实现增加功能时，需改动的地方比较多

**6、原型模式**

原型模式（Prototype），用原型实例指定创建对象的种类，并且通过拷贝这些原型创建新的对象。

**6.1原型模式的优点：**

创建通过复制，可以提高新建的效率   
**6.2原型模式的缺点：**

克隆位与类内部，不易于修改源代码

**7、建造者模式**

建造者模式（Builder），将一个复杂对象的构建与它的表示分离，使得同样的构建过程可以创建不同的表示。

**7.1建造者模式的优点：**

使建造代码与表示代码分离，由于建造者隐藏了该产品是如何组装的，所以若需要改变一个产品的内部表示，只需要再定义一个具体的建造者就可以。   
**7.2建造者模式的缺点：**

如果产品内部变化复杂，会增加系统的难度和运行成本

**8、观察者模式**

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。

**8.1观察者模式的优点：**

（1）观察者模式在被观察者和观察者之间建立一个抽象的耦合。被观察者角色所知道的只是一个具体观察者列表，每一个具体观察者都符合一个抽象观察者的接口。被观察者并不认识任何一个具体观察者，它只知道它们都有一个共同的接口。由于被观察者和观察者没有紧密地耦合在一起，因此它们可以属于不同的抽象化层次。如果被观察者和观察者都被扔到一起，那么这个对象必然跨越抽象化和具体化层次。

（2）观察者模式支持广播通讯。被观察者会向所有的登记过的观察者发出通知，

**8.2观察者模式的缺点**

（1）如果一个被观察者对象有很多的直接和间接的观察者的话，将所有的观察者都通知到会花费很多时间。

（2）如果在被观察者之间有循环依赖的话，被观察者会触发它们之间进行循环调用，导致系统崩溃。在使用观察者模式是要特别注意这一点。

（3）如果对观察者的通知是通过另外的线程进行异步投递的话，系统必须保证投递是以自恰的方式进行的。

（4）虽然观察者模式可以随时使观察者知道所观察的对象发生了变化，但是观察者模式没有相应的机制使观察者知道所观察的对象是怎么发生变化的。

1. **门面模式**

门面模式要求一个子系统的外部与其内部的通信必须通过一个统一的门面对象进行。门面模式提供一个高层次的接口，使得子系统更易于使用。就如同医院的接待员一样，门面模式的门面类将客户端与子系统的内部复杂性分隔开，使得客户端只需要与门面对象打交道，而不需要与子系统内部的很多对象打交道。

**9.1门面模式的优点**

1. 它可以对客户屏蔽子系统组件，因而减少了客户处理对象的数目并使得子系统使用起来更加方便。
2. 它实现了子系统与客户之间的松耦合关系，而子系统内部的功能组件往往紧耦合的。松耦合关系使得子系统的组件变化不会影响到它的客户。Facade模式有助于建立层次结构系统，也有助于对对象之间的依赖关系分层。Facade模式可以消除复杂的循环依赖关系。用Facade可以降低编译依赖性，限制重要系统中较小的变化所需的重编译工作。Facade模式同样也有利于简化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。
3. 如果应用需要，它并不限制它们使用子系统类。因此你可以让客户程序在系统易用性和通用性之间加以选择。

**9.2门面模式的缺点**

1. 当你要为了一个复杂子系统提供一个简单接口时。
2. 由于抽象类的实现部分与客户程序之间存在着很大的依赖性。引入facade将这个子系统与客户以及其他的子系统分离，可以提高子系统的独立性和可移植性。
3. 当你需要构建一个层次结构的子系统时，使用facade模式定义子系统中每层的入口点。如果子系统之间是相互依赖的，你可以让它们仅通过facade进行通讯，从而简化了它们之间的依赖关系。
4. **抽象工厂模式**

抽象工厂模式（Abstract Factory），提供一个创建一系列相关或相互依赖对象的接口，而无需指定它们具体的类。

****10.1抽象工厂模式的优点：****

抽象工厂模式除了具有工厂方法模式的优点外，最主要的优点就是可以在类的内部对产品族进行约束。所谓的产品族，一般或多或少的都存在一定的关联，抽象工厂模式就可以在类内部对产品族的关联关系进行定义和描述，而不必专门引入一个新的类来进行管理。

****10.2抽象工厂模式的缺点：****

产品族的扩展将是一件十分费力的事情，假如产品族中需要增加一个新的产品，则几乎所有的工厂类都需要进行修改。所以使用抽象工厂模式时，对产品等级结构的划分是非常重要的。