链滴

# 00_ 设计模式概述【每天学一种设计模式】

作者：dianjiu

原文链接：https://ld246.com/article/1592963473166

来源网站：链滴

# 一、为什么使用设计模式?

使用设计模式的目的是为了可重用代码、让代码更容易被他人理解、保证代码可靠性，当然设计模式不是万能的，项目中的实际问题还有具体分析。咱不能为了使用设计模式而使用，而是在分析问题的程中，想到使用某种设计模式能达到咱需要的效果，而且比不使用设计模式更有优势，那么咱该考虑用设计模式了。

# 二、设计模式的分类

## 创建型模式（共5种）：

单例、原型、建造者、工厂方法、抽象工厂。

## 结构型模式（共7种）：

适配器、代理、外观、桥接、组合、享元、装饰者。

## 行为型模式（共11种）：

策略、观察者、模板方法、迭代器、责任链、命令、备忘录、状态、访问者、中介者、解释器。

# 三、23种设计模式的简单理解

## 创建型模式（共5种）

### 1、单例模式

单例（Singleton）是一种常用的设计模式。在Java应用中，单例对象能保证在一个JVM中，该对象有一个实例存在。实现方式主要有饿汉式和懒汉式两种。

## 2、原型模式

该模式的思想就是将一个对象作为原型，对其进行复制、克隆，产生一个和原对象类似的新对象。实方式主要有浅复制和深复制两种。浅复制的关键是super.clone()；而深复制，需要采用二进制流的形写入当前对象，再对其进行读取。

## 3、建造者模式

该模式是将各种产品集中起来进行管理。将很多功能集成到一个类里，这个类可以创造出比较复杂的西。它关注的是创建复合对象，多个部分。

## 4、工厂方法

调用工厂里的方法来生产对象(产品)的。它有3种实现方式：

1)普通工厂模式：就是建立一个工厂类，对实现了同一接口的一些类进行实例的创建)。

2)多个工厂方法模式：是对普通工厂方法模式的改进，在普通工厂方法模式中，如果传递的字符串出，则不能正确创建对象，而多个工厂方法模式是提供多个工厂方法，分别创建对象。

3)静态工厂方法模式：将上面的多个工厂方法模式里的方法置为静态的，不需要创建实例，直接调用可。如果你想使用工厂方法模式可以优先选择：静态工厂方法模式。

## 5、抽象工厂

顾名思义，就是把工厂抽象出来，不同的工厂生产不同的产品。

# 结构型模式（共7种）

## 1、适配器模式

将某个类的接口转换成客户端期望的另一个接口表示，目的是消除由于接口不匹配所造成的类的兼容问题。主要分为三类：类的适配器模式、对象的适配器模式、接口的适配器模式。

1)类的适配器模式：当希望将一个类转换成满足另一个新接口的类时，可以使用类的适配器模式，创一个新类，继承原有的类，实现新的接口即可。

2)对象的适配器模式：当希望将一个对象转换成满足另一个新接口的对象时，可以创建一个Adapter，持有原类的一个实例，在Adapter类的方法中，调用实例的方法就行。

3)接口的适配器模式：当不希望实现一个接口中所有的方法时，可以创建一个抽象类Adapter实现所方法，我们写别的类的时候，继承抽象类即可。

## 2、代理模式

代理模式其实就是多一个代理类出来，替原对象进行一些操作。比如咱有的时候打官司需要请律师，为律师在法律方面有专长，可以替咱进行操作表达咱的想法，这就是代理的意思。有两种实现方式：

态代理(不使用JDK里面的方法)、动态代理(InvocationHandler和Proxy)。

## 3、门面模式

外观模式是为了解决类与类之间的依赖关系的，像spring一样，可以将类和类之间的关系配置到配置件中，而外观模式就是将他们的关系放在一个Facade类中，降低了类类之间的耦合度，该模式中没有及到接口。

## 4、桥接模式

把事物和其具体实现分开(抽象化与实现化解耦)，使他们可以各自独立的变化。桥接模式其实就是一将N*M转化成N+M组合的思想。

## 5、组合模式

组合模式有时又叫部分-整体模式，将对象组合成树形结构来表示“部分-整体”层次结构。

## 6、享元模式

运用共享的技术有效地支持大量细粒度的对象。主要目的是实现对象的共享，即共享池，当系统中对多的时候可以减少内存的开销。在某种程度上，你可以把单例看成是享元的一种特例。

## 7、装饰者模式

动态地将责任附加到对象上，若要扩展功能，装饰者提供了比继承更具有弹性的替代方案。保持接口增强性能。

# 行为型模式（共11种）

## 1、策略模式

用户可以选择不同的策略来进行操作。个人觉得策略模式可以用这个公式：不同的XXX 拥有不同的XX供用户选择。比如说：不同的象棋棋子拥有不同的走法供用户选择。

## 2、观察者模式

在对象之间定义了一对多的依赖关系，这样一来，当一个对象改变状态时，依赖它的对象都会收到通并自动跟新。Java已经提供了对观察者Observer模式的默认实现，Java对观察者模式的支持主要体在Observable类和Observer接口。

http://point9.top/articles/2020/06/30/1593485540463.html

## 3、模板方法

在一个方法中定义了一个算法的骨架，而将一些步骤延迟到子类中。模板方法使得子类可以再不改变法结构的情况下，重新定义算法中的某些步骤。简而言之：模板方法定义了一个算法的步骤，并允许类为一个或多个步骤提供实现。

## 4、迭代器模式

提供了一种方法顺序访问一个聚合对象中的各个元素，而又不暴露其内部的表示。

## 5、责任链模式

有多个对象，每个对象持有下一个对象的引用，形成一条链，请求在这条链上传递，直到某一对象决处理该请求，但是发出者并不清楚最终哪个对象会处理该请求。

## 6、命令模式

将“请求”(命令/口令)封装成一个对象，以便使用不同的请求、队列或者日志来参数化其对象。命令式也支持撤销操作。

## 7、备忘录模式

主要目的是保存一个对象的某个状态，以便在适当的时候恢复对象。

## 8、状态模式

允许对象在内部状态改变时改变它的行为，对象看起来好像修改了它的类。状态模式说白了就是一个像有不同的状态，不同的状态对应不同的行为，它其实是对switch case这样的语句的拓展。

## 9、解释器模式

它定义了对象与对象之间进行某种操作之后会得到什么值。一般主要应用在OOP开发中的编译器的开中，所以适用面比较窄。

## 10、中介者模式

主要用来降低类与类之间的耦合的，因为如果类与类之间有依赖关系的话，不利于功能的拓展和维护因为只要修改一个对象，其它关联的对象都得进行修改。

## 11、访问者模式

把数据结构和作用于结构上的操作解耦合，使得操作集合可相对自由地演化。访问者模式适用于数据构相对稳定而算法又容易变化的系统。访问者模式的优点是增加操作很容易，因为增加操作意味着增新的访问者；而它的缺点就是增加新的数据结构很困难。

# 四、代码实例

# 仓库地址

https://github.com/dianjiu/design-pattern

https://gitee.com/dianjiu/design-pattern

# 模式详解