设计模式学习笔记

[零、基本 2](#_Toc26093253)

[1. 参考资料 2](#_Toc26093254)

[2. 分类 2](#_Toc26093255)

[3. 设计原则 3](#_Toc26093256)

[一、策略模式(Strategy Pattern) 3](#_Toc26093257)

[二、工厂模式(Factory Pattern) 4](#_Toc26093258)

[1. 简单工厂 4](#_Toc26093259)

[2. 工厂模式 4](#_Toc26093260)

[3. 抽象工厂 4](#_Toc26093261)

[三、模板模式(Template Pattern) 4](#_Toc26093262)

[四、策略&模板&工厂 4](#_Toc26093263)

[五、观察者模式(Observer Pattern) 4](#_Toc26093264)

[六、单例模式(Singleton Pattern) 4](#_Toc26093265)

[七、装饰者模式(Decoration Pattern) 5](#_Toc26093266)

[八、命令模式(Command Pattern) 5](#_Toc26093267)

[九、适配器模式(Adapter Pattern) 5](#_Toc26093268)

[十、外观模式(Facade Pattern) 5](#_Toc26093269)

[十一、代理模式(Proxy Pattern) 5](#_Toc26093270)

[十二、适配器&外观&代理 5](#_Toc26093271)

[十三、迭代器模式(Iterator Pattern) 5](#_Toc26093272)

[十四、组合模式(Composite Pattern) 5](#_Toc26093273)

# 零、基本

## 参考资料

1. Head First 设计模式

## 分类

* 1. 创建型

单例、工厂、抽象工厂、建造者、原型

* 1. 结构型

适配器、装饰、桥接、组合、享元、代理、外观

* 1. 行为型

观察者、模版、策略、访问者、中介者、解释器、迭代器、状态、命令、备忘录、职责链

## 设计原则

* 1. 找出应用中可能需要变化之处，把它们独立出来，不要和那些不需要变化的代码混在一起
  2. 为了交互对象之间的松耦合设计而努力
  3. 多用组合，少用继承
  4. 依赖倒转原则(Dependency Inversion Principle, DIP)：针对接口编程，不针对实现编程
  5. 开闭原则(Open-Closed Principle, OCP)：类应该对扩展开放，对修改关闭。应尽量在不修改原有代码的情况下进行扩展
  6. 里氏代换原则(Liskov Substitution Principle, LSP)：所有引用基类（父类）的地方必须能透明地使用其子类的对象。要依赖抽象，不要依赖具体类
  7. 迪米特法则(Law of Demeter, LoD)：一个软件实体应当尽可能少地与其他实体发生相互作用，最少知识原则，只和朋友交谈
  8. 好莱坞原则：不要给我们打电话，我们会给你打电话
  9. 接口隔离原则(Interface Segregation Principle, ISP)：使用多个专门的接口，而不使用单一的总接口，即客户端不应该依赖那些它不需要的接口。
  10. 单一职责原则(Single Responsibility Principle, SRP)：单一责任，一个类应该只有一个引起变化的原因

# 一、策略模式(Strategy Pattern)

1. 定义

定义了算法族，分别封装起来，让他们之间可以互相替换，此模式让算法的变化独立于算法的客户。

1. 例子

A screenshot of a cell phone

Description automatically generated

1. In Spring boot
2. 实际应用场景

# 二、工厂模式(Factory Pattern)

## 简单工厂

* 1. 定义

简单工厂模式是由一个工厂对象决定创建出哪一种产品类的实例，是一种创建型模式，但不属于23种GOF设计模式之一，比较像一种编程习惯。

* 1. 例子

## 工厂模式

* 1. 定义

定义了一个创建对象的接口，但由子类决定改创建对象是什么，把类的实例化推迟到子类，来达到将对象创建过程封装的目的。

* 1. 例子

## 抽象工厂

* 1. 定义

提供一个借口，用于创建相关或依赖对象的家族，而不需要依赖具体的类

* 1. 例子

1. 工厂&抽象工厂
2. 实际应用场景

# 三、模板模式(Template Pattern)

1. 定义

在一个方法中定义一个算法的骨架，而将一些步骤延迟到子类中。使得子类可以在不改变算法结构的情况下，重新定义算法的某些步骤。

1. 例子
2. In spring boot
3. 实际应用场景

# 四、策略&模板&工厂

1. 区别
2. 综合运用
3. 实际应用场景

# 五、观察者模式(Observer Pattern)

1. 定义

定义了对象之间的一对多依赖。

1. 例子
2. In spring boot
3. 实际运用场景

# 六、单例模式(Singleton Pattern)

1. 定义
2. 五种写法
   1. 饿汉式
   2. 懒汉一
   3. 懒汉二(单重锁)
   4. 懒汉三(双重锁)
   5. 懒汉四(双重锁+ volatile)
   6. 静态内部类
   7. 枚举单例
3. 应用场景

# 七、装饰者模式(Decoration Pattern)

1. 定义
2. 例子
3. In Java

# 八、命令模式(Command Pattern)

1. 定义
2. 例子
3. In Linux

# 九、适配器模式(Adapter Pattern)

# 十、外观模式(Facade Pattern)

# 十一、代理模式(Proxy Pattern)

# 十二、适配器&外观&代理

# 十三、迭代器模式(Iterator Pattern)

# 十四、组合模式(Composite Pattern)