极客学院
jikexueyuan.com

设计模式之最后总结

设计模式之最后总结 — **课程概要**

- 设计模式的三个分类
- 对象设计的六大原则
- 用模式来思考

设计模式的三个分类

设计模式的三个分类

- 什么是设计模式
- 设计模式的三个分类
- 各分类中模式的关键点

设计模式的三个分类 — **什么是设计模式**

模式：在某些场景下，针对某类问题的某种通用解决方案

场景：项目环境

问题：约束条件，项目目标等

解决方案：通用、可以复用的设计，解决约束，达到目标

设计模式的三个分类 — **设计模式的三个分类**

创建型模式：对象实例化的模式，创建型模式解耦了对象的实例化过程

结构型模式：把类或对象结合在一起形成更大的结构

行为型模式：类和对象如何交互，及划分责任和算法

设计模式的三个分类— **各分类中模式的关键点**

简单工厂：一个工厂类根据传入的参量决定创建出哪一种产品类的实例

工厂方法：定义一个创建对象的接口，让子类决定实例化哪一个类

抽象工厂：创建相关或依赖对象的家族，而无需明确指定具体类

单例模式：某个类只能有一个实例，提供一个全局访问点

生成器模式：封装一个复杂对象的构建过程，并可以按步骤构造

原型模式：通过复制现有的实例来创建新的实例

设计模式的三个分类 — **各分类中模式的关键点**

适配器模式：将一个类的方法接口转换成客户希望的另外一个接口

组合模式：将对象组合成树形结构以表示“部分-整体”的层次结构

装饰模式：动态地给对象添加新的功能

代理模式：为其他对象提供一个代理以控制对这个对象的访问

蝇量模式：通过共享技术有效地支持大量细粒度的对象

外观模式：提供统一的方法来访问子系统的一群接口

桥接模式：将抽象部分与它的实现部分分离，使它们都可以独立地变化

设计模式的三个分类 — **各分类中模式的关键点**

模板模式：定义一个算法结构，而将一些步骤延迟到子类中实现

解释器模式：给定一个语言， 定义它的文法的一种表示，并定义一个解释器

策略模式：定义一系列的算法，把它们封装起来， 并且使它们可相互替换

状态模式：允许一个对象在其内部状态改变时改变它的行为

观测者模式：对象间的一对多的依赖关系

备忘录模式：在不破坏封装性的前提下，保存对象的内部状态

中介者模式：用一个中介对象来封装一系列的对象交互

命令模式：将命令请求封装为一个对象，使得可用不同的请求来进行参数化

设计模式的三个分类 — **各分类中模式的关键点**

访问者模式：在不改变数据结构的前提下，增加作用于一组对象元素新的功能

责任链：请求发送者和接收者之间解耦，使的多个对象都有机会处理这个请求

迭代器：一种遍历访问聚合对象中各个元素的方法，不暴露该对象的内部结构

对象设计的六大原则

对象设计的六大原则 — **对象设计六大原则**

- 组合复用原则
- 依赖倒置原则
- 开闭原则
- 迪米特法则
- 里氏替换原则
- 单一职责原则

对象设计的六大原则 — **组合复用原则**

多用组合，少用继承

找到变化部分，抽象，封装变化

区分“Has-A”与“Is-A”

对象设计的六大原则 — **依赖倒置原则**

依赖：成员变量、方法参数、返回值

要依赖于抽象，不要依赖于具体

高层模块不应该依赖低层模块，二者都应该依赖其抽象

抽象不应该依赖具体，具体应该依赖抽象

针对接口编程，不要针对实现编程

以抽象为基础搭建的结构比具体类搭建的结构要稳定的多

在java中，抽象指的是接口或者抽象类，具体就是具体的实现类

对象设计的六大原则 — **开闭原则**

对扩展开放，对修改关闭

通过扩展已有软件系统，可以提供新的功能

修改的关闭，保证稳定性和延续性

对象设计的六大原则 — **里氏替换原则**

所有引用基类的地方必须能透明地使用其子类对象

子类在扩展父类功能时不能破坏父类原有的功能

使用继承时，遵循里氏替换原则：

子类可以实现父类的抽象方法，但不能覆盖父类的非抽象方法。

当子类重载父类方法时，方法的形参要比父类方法的参数更宽松

当子类实现父类的抽象方法时，方法的返回值要比父类更严格

里氏替换原则是设计整个继承体系的原则

对象设计的六大原则 — **迪米特法则**

一个对象应该与其他对象保持最少的了解。只与直接朋友交谈。

成员变量、方法参数、方法返回值中需要的类为直接朋友

类与类之间的关系越密切了解越多，耦合度越大

尽量降低类与类之间的耦合

外观模式、中介者模式

接口隔离原则：一个类对另一个类的依赖应该建立在最小的接口上

对象设计的六大原则 — **单一职责原则**

类应该只有一个导致类变更的理由

即一个类只负责一项职责

降低类的复杂度

提高系统的可维护性

修改时降低风险溢出

用模式来思考

用模式来思考

- 保持简单
- 设计模式非万能
- 何时需要模式
- 重构的时间就是模式的时间

用模式来思考 — **保持简单**

尽可能用最简单的方式解决问题

简单而弹性的设计，一般使用模式是最好的方法

用模式来思考 — **设计模式非万能**

模式是通用问题的经验总结

使用模式时要考虑它对其他部分的影响

不需要预留任何弹性的时候，删除掉模式

平衡与妥协

用模式来思考 — **何时需要模式**

找出设计中会变化的部分，通常就是需要考虑模式的地方

重构时

用模式来思考 — **重构的时间就是模式的时间**

重构就是改变代码来改进组织方式的过程

利用模式来重构

极客学院

jikexueyuan.com

中国最大的IT职业在线教育平台