Design Pattern

《设计模式》这本书从顶层对软件设计经验进行系统的总结。

# 1 设计模式基础

## 什么是设计模式

设计模式由pattern name 、problem、solution、consequences四个步骤构成。Christopher Alexander说过：“每一个模式描述了一个在我们周围不断重复发生的问题，以及该问题的解决方案的核心。这样，你就能一次又一次地使用该方案而不必做重复的劳动。”

## 23个设计模式

|  |  |
| --- | --- |
| 模式名称 | 效果 |
| Abstract Factory | 将一个类的接口转换成客户希望的另外一个接口。解决类接口不兼容的情况 |
| Bridge | 将抽象部分与实现部分分离，使其可分别独立变化 |
| Builder | 将复杂对象的构建与其表示分离，的构造过程可以创建不同的表示 |
| Chain of Responsibility | 解决请求的发送者与接收者之间的耦合，而使多个对象都有机会处理该请求将请求对象连城一条链，并沿该条链传递该请求，直到有一个对象处理它 |
| Command | 将请求分装为一个对象，支持请求双方的记录、排队、清除等操作 |
| Composite | 将对象组合成树形结构，表示“部分-整体”的层次结构；使得客户对单个对象和符复合对象具有一致性 |
| Decorator | 动态地给一个对象添加额外的职责。相比于生成子类更灵活 |
| Facade | 为系统中的一组接口提供一个一致的界面，从顶层上设计接口，更为便利的使用其子系统 |
| Factory Method | 定义一个用于创建对象的接口，让子类决定将哪一个类实例化 |
| Flyweight | 运用共享技术有效地支持大量细粒度的对象（粒度就是取不同大小的对象-将原来“粗粒度”的大对象分割成为若干“细粒度”的小对象，或者把若干小对象合并成一个大的粗粒度对象，进行研究） |
| Interpreter | 给定一种语言，定义其文法、解释器 |
| Mediator | 采用中介对象来封装一系列的对象交互，中介者使各对象不需要显示地相互引用，从而使其耦合松散，而且可以独立地改变它们之间的交互 |
| Memento | 在不破坏封装性的前提下，捕获一个对象的内部状态，并在该对象之外保存这个状态，用于恢复对象的原始状态 |
| Observer | 定义对象间的一种一对多的依赖关系，以便当一个对象的状态发生改变时，所有依赖它的对象都得到通知并自动刷新 |
| Prototype | 用原型实例指定创建对象的种类，并且通过拷贝这个原型来创建新的对象 |
| Proxy | 为其它对象提供一个代理以控制这个对象的访问 |
| Singleton | 保证一个类仅有一个实例，并提供一个访问它的全局访问点 |
| State | 允许一个对象在其内部状态改变时改变它的行为，对象看起来视乎修改了它所属的类 |
| Strategy | 定义一系列的算法，把它们一个个封装起来，并且使它们可相互替换。算法的变化可独立于其客户 |
| Template Method | 定义一个算法中的算法骨架，而将一些步骤延迟到子类中。子类可以不改变算法的结构即可重定义该算法的某些特定步骤。 |
| Visitor | 表示一个作用域某对象结构中的各元素的操作。在不改变各元素类的前提下定义作用于这些元素的新操作。 |

## 23个设计模式分类







