## Headfirst 设计模式—设计原则

|  |  |  |
| --- | --- | --- |
| 模式名称 | 定义 | 其他 |
| 策略模式(strategy) | 策略模式定义了一系列的算法，并将每一个算法封装起来，而且使它们还可以相互替换。策略模式让算法独立于使用它的客户而独立变化。 |  |
| 观察者模式(Observer) | 观察者模式(有时又被称为发布(publish)-订阅(Subscribe)模式、模型-视图(View)模式、源-收听者(Listener)模式或从属者模式）：一个目标物件管理所有相依于它的观察者物件，并且在它本身的状态改变时主动发出通知。这通常透过呼叫各观察者所提供的方法来实现。此种模式通常被用来实现事件处理系统。 |  |
| 装饰者模式  (Decorator) | 动态的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有弹性的替代方案。 |  |
| 工厂方法模式  (FactoryMethod) | 定义了一个创建对象的接口,但是由子类决定实例化的类是哪一个。工厂方法把实例化推迟到子类。 |  |
| 抽象工厂模式 | 提供一个接口用于创建相关或依赖对象的家族，而不需要明确指定具体类。 | — |
| 命令模式(Command) | 将”请求”封装成对象，以便使用不同的请求、队列或日志来参数化其他对象。命令模式也支持可撤销的操作。 |  |
| 适配器模式(Adapter) | 将一个类的接口转换成客户期望的接口。适配器让原来不兼容的类可以合作无间。 |  |
| 外观模式  (Facade) | 提供了统一的接口，用来访问子系统的一群接口。外观定义了一个高层接口，让子系统更容易使用！ |  |
| 模板方法(Template Method) | 在一个方法中定义一个方法的骨架，而且将一些步骤延迟到子类中。模板方法可以在子类不改变算法结构的情况下，重新定义某些算法的步骤。 |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | 组合(has-a)关系：在新类中new 另外一个类的对象，以添加该对象的特性。  委托(代理):在代理类中创建某功能的类，调用类的一些方法以获得该类的部分特性. |  |

1. 找出应用中可能需要变化之处，把他们独立出来，不要和那些不需要变化的代码混在一起。
2. 针对接口编程，而不是针对实现编程。
3. 为了交互对象之间的松耦合而努力。
4. 开放—关闭原则：类应该对扩展开放，对修改关闭
5. 依赖倒置原则(dependency inversion principle)要依赖抽象，不要依赖具体实现
6. 最少知识原则: 迪米特法则（Law of Demeter）又叫作最少知道原则（Least Knowledge Principle 简写LKP），就是说一个对象应当对其他对象有尽可能少的了解,不和陌生人说话。英文简写为: LoD
7. 好莱坞原则:别调用我们，我们调用你！