Java设计模式学习心得 PDF转换可能丢失图片或格式，建议阅读原文

https://www.100test.com/kao_ti2020/645/2021_2022_Java_E8_AE_BE_E8_AE_A1_c104_645222.htm 整个设计模式贯穿一个原理:面对介面编程，而不是面对实现，（面向物件编程应该改面向介面编程）。目标原则是:降低耦合，增强灵活性。 一、创建模式 1. 设计模式之Factory(工厂方法和抽象工厂) 使用工厂模式就象使用new一样频繁. 2. 设计模式之Prototype(原型) 用原型实例指定创建物件的种类，且通过拷贝这些原型创建新的物件。 3. 设计模式之Builder 汽车由车轮 方向盘 发动机很多部件组成，同时，将这些部件组装成汽车也是一件杂的工作，Builder模式就是将这两种情况分开进行。 4. 设计模式之Singleton(单态) 保一个类只有一个实例,提供一个访问它的全局访问点 二、结构模式 1. 设计模式之Facade 可扩展的使用JDBC针对不同的资料库编程,Facade提供了一种灵活的实现。 2. 设计模式之Proxy 以Jive例,剖析代理模式在用户级别授权机制上的应用 3. 设计模式之Adapter 使用类再生的两个方式:组合(new)和继承(extends),这个已经在"thinking in java"中提到过。 4. 设计模式之Composite 就是将类用树形结构组合成一个单位。你向别人介绍你是某单位，你是单位元元中的一个元素，别人和你做买卖，相当於和单位做买卖。文章中还对Jive再进行了剖析。 5. 设计模式之Decorator Decorator是个油漆工,给你的东东的外表刷上美丽的色。 6. 设计模式之Bridge 将"牛郎织女"分开(本应在一起,分开他们,形成两个介面),在他们之间搭建一个桥(动态的结合) 7. 设计模式之Flyweight 提供Java运行性能,降低小而大量重复的类的开销

。 三、 行模式 1. 设计模式之Template 实际上向你介绍了什麼要使用Java 抽象类,该模式原理简单,使用很普遍。 2. 设计模式之Memento 很简单一个模式,就是在记忆体中保留原来资料的拷贝。 3. 设计模式之Observer 介绍如何使用Java API提供的现成Observer 4. 设计模式之Chain of Responsibility 各司其职的类串成一串,好象击鼓传花,当然如果自己能完成,就不要推委给下一个。 5. 设计模式之Command 什麼是将行封装,Command 是最好的说明。 6. 设计模式之State 状态是编程中经常碰到的实例,将状态物件化,设立状态变换器,便可在状态中轻切换。 7. 设计模式之Strategy 不同演算法各自封装,用户端可随意挑选需要的演算法。 8. 设计模式之Mediator Mediator很象十字路口的红绿灯,每个车辆只需和红绿灯交互就可以。 9. 设计模式之Interpreter 主要用来对语言的分析,应用机会不多。 10. 设计模式之Visitor 访问者在进行访问时,完成一系列实质性操作,而且还可以扩展。 11. 设计模式之Iterator 这个模式已经被用来遍Collection中物件。使用频率很高。在Java中无需专门阐述,在大多数场合也无需自己制造一个Iterator,只要将物件装入Collection中，我们就直接可以使用Iterator模式。 