Java设计模式之Command模式计算机等级考试 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/646/2021_2022_Java_E8_AE_BE_E8_AE_A1_c97_646092.htm Command 模式是最让我疑惑的一个模式,我在阅读了很多代码后,才感觉隐约掌握其大概原理,我认为理解设计模式最主要是掌握起原理构造,这样才对自己实际编程有指导作用.Command 模式实际上不是个很具体,规定很多的模式,正是这个灵活性,让人有些confuse. Command 定义 不少Command 模式的代码都是针对图形界面的,它实际就是菜单命令,我们在一个下拉菜单选择一个命令时,然后会执行一些动作. 将这些命令封装成在一个类中,然后用户(调用者)再对这个类进行操作,这就是Command 模式,换句话说,本来用户(调用者)是直接调用这些命令的,如菜单上打开文档(调用者),就直接指向打开文档的代码,使用Command 模式,就是在这两者之间增加一个中间者,将这种直接关系拗断,同时两者之间都隔离,基本没有关系了. 显然这样做的好处是符合封装的特性,降低耦合度,Command 是将对行为进行封装的典型模式,Factory 是将创建进行封装的模式,从Command 模式,我也发现设计模式一个"通病":好象喜欢将简单的问题复杂化, 喜欢在不同类中增加第三者,当然这样做有利于代码的健壮性 可维护性 还有复用性. 如何使用? 具体的Command 模式代码各式各样,因为如何封装命令,不同系统,有不同的做法.下面事例是将命令封装在一个Collection 的List 中,任何对象一旦加入List 中,实际上装入了一个封闭的黑盒中,对象的特性消失了,只有取出时,才有可能模糊的分辨出: 典型的Command 模式需要有一个接口.接口中有一个统一的方法,这就是"将命令/请求封装为对

象": public interface Command { public abstract void execute ( ). } 具体不同命令/请求代码是实现接口Command,下面有三个具体命令 public class Engineer implements Command { public void execute( ) { //do Engineers command } } public class Programmer implements Command { public void execute( ) { //do programmers command } } 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com