

**实验报告**

|  |  |
| --- | --- |
| **课程名称**: | 设计模式 |
| **开课学期**: | 2020-2021学年 第2学期 |
| **专业**: | 软件工程 |
| **班级年级**: | 2018级 |
| **学生姓名**: | 宋行健 |
| **学号:** | 222018321062006 |
| **实验教师:** | 王晓蒙 |

**计算机与信息科学学院 软件学院**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验项目名称 | | 实验6 结构型模式3 | | | |
| 实验时间 | | 2021/4/25 | Type | \*验证性□设计性 □综合性 | |
| 1. 思路  首先对老师发的代码进行逐句研读，了解代码架构，并绘制类图。通过类图的展示，程序的逻辑逐渐明显。程序主要被封装在ProxyApp进行主流程，在这个类中进行了页面的布局和各种控件响应函数的创建，通过调试运行，这个程序可以通过输入账号密码进行验证登录，在未登录状态不能进行计算。  本程序采用代理方式实现，代理的接口是ICalculator（原IAdd）其对应的被代理对象为Calculator。被代理类通过MyInvocationHandler进行调用，动态代理对象为proxyCalculator，是利用Java的反射技术，在运行时创建一个实现ICalculator接口的新类及其实例。  2. 类图    3. 代码分析——代理模式  本程序采用动态代理的方式实现。代理的接口是ICalculator（原IAdd）其对应的被代理对象为Calculator。  动态代理对象为proxyCalculator。它利用了Java的反射技术，在运行时创建一个实现ICalculator（原IAdd）接口的实例。    图 1 动态生成代理对象  动态代理类MyInvocationHandler，实现了InvocationHandler这个接口。其实例关联到了proxyCalculator，当我们通过代理对象proxyCalculator调用一个方法的时候，这个方法的调用就会被转发为由InvocationHandler这个接口的invoke方法来进行调用，如下图所示：    图 2 MyInvocationHandler类的 invoke方法  另外，在检验登录状态的时候，程序还是用了单例模式的思想，Manager是单例，只创建一次，每次检验时，直接调用其中的方法管理登录状态。    图 3 单例模式Manager  4. 改进  **（一）未登录状态下计算失败弹窗**  原程序在未登录状态下点击“=”进行运算，没有弹窗提示，用户体验感较差。改进后在“=”操作触发时，如果已登录就显示值，未登录则显示弹窗。    图 4未登录状态下计算失败弹窗代码    图 5 未登录状态下计算失败的弹窗  **（二）增加加减乘除四则运算**  原程序中只能计算加法，我将之前IAdd接口改成了ICalculator接口，并增加了减法、乘法、除法。这里我没有重新创建新的代理，而是直接增加了原有代理的功能。其中需要注意的地方是，在运算除法的时候，被除数不能为0，需要先检验一下。    图 6 修改后的ICalculator类    图 7 修改后的Calculator类    图 8 除法除数不为零  另外，为了可以进行四则运算的切换，我添加了一个修改方法的切换按钮。设置了一个int变量func记录当前执行的运算法则，“0-加法，1-减法，2-乘法，3-除法”，并且在每次切换方法后自动进行结果运算，同时修改运算符号。      图 9 切换方法程序代码  5. 运行结果  对于程序的主要界面我没有进行太大的修改，只是添加了一个“修改方法”的按钮，并且随着修改方法，运算符号也会随之改变。另外，我添加了两个消息提示框，分别提示登录信息和除法运算除数不为零。          6. 总结  该项目是动态代理模式的实现，在程序运行时，运用反射机制动态创建而成。首先，通过实现 InvocationHandler 接口创建调用处理器，通过为代理类指定被代理的对象和一组接口来创建动态代理类，之后通过反射机制获得动态代理类的构造函数，构造函数创建动态代理类实例，构造时调用处理器对象作为参数传入。  代理模式主要解决了例如写日志、验证等造成代码混乱的业务需求。另一方面解决了多次使用相同的代码。  在进行程序改进的过程中，我添加了四则运算的功能，没有增加新的接口，而是把原先的IAdd接口改成了ICalculator，并将四个运算方法都放在了里面。通过设置接口中的方法进行四则运算。  通过观察反编译文件，ICalculator中4个方法都插入到了代理中，但是和老师讨论后，把所有的方法都放在一个接口里面是一个不好的模式，这种代理模式耦合度太强，更加好的模式是用多个方法接口，将四则运算分别进行代理。    图 10 反编译代码 | | | | | |
| 教师评阅 | 实验内容（60%）： | | | |  |
| 实验思路（20%）： | | | |  |
| 实验总结（20%）： | | | |  |
| 实验成绩： | | | | |