****

**面向对象程序设计上机实验报告**

实验7：类的动态创建和注入

**学院名称 智能与计算学部**

**专 业 软件工程**

**学生姓名 陈昊昆**

**学 号 3021001196**

**年 级 2021级**

**班 级 3班**

**时 间 2022年 4月23日**

1. **实验目的**
2. 理解配置文件的读写；
3. 理解反射、注入；
4. 练习使用DEBUG工具调试程序；
5. **实验内容**
6. 系统中有若该类等待构造（这些类的名字在程序运行的时候动态的从配置文件中读取），配置文件格式类似如下：

#系统的配置文件

#这个文件里面可能有注释，以#开头表示注释

#这个文件肯定以UTF-8编码

#每个类的配置部分由如若干行描述构成

#每个描述如下： [类的完整路径名].[属性]="一个字符串"/或者一个整数，其中属性可能是任意小写单词

#测试的时候，要考虑文件是否存在乱码的问题

#com.huawei.classroom.student.h18.Apple 是类名; color/weight 是属性名,"red"/500是属性

#如果属性以""开头结尾则表示一个字符串属性，否则肯定是一个整数的属性

com.huawei.classroom.student.h18.Apple.color="红色"

com.huawei.classroom.student.h18.Apple.weight=500

com.huawei.classroom.student.h18.Car.price=200000

com.huawei.classroom.student.h18.Car.color="黑色"

com.huawei.classroom.student.h18.Car.brand="大众"

......

1. 程序从配置文件中动态的读取每个类的名字、属性名、属性值，动态构造出对象，并通过测试用例。
2. **程序实现**

根据指定的配置文件动态创建类

**public** **class** MyClassFactory {

**private** Param param=**null**;

**public** MyClassFactory(String configFile) **throws** FileNotFoundException, IOException {

// 构造函数，根据指定配置文件名创建 Param 对象

param=**new** Param(configFile);

}

**public** <T> T createInstance( Class<T> clazz) **throws** InstantiationException, IllegalAccessException, IllegalArgumentException, InvocationTargetException, NoSuchMethodException, SecurityException, ClassNotFoundException{

// 根据指定类名创建实例，并将实例属性值设置为配置文件中的值

String className=clazz.getName();

T t = clazz.newInstance();

Map<String,Object> props=param.getParamMap(className) ;

Iterator<String> it=props.keySet().iterator();

**while**(it.hasNext()) {

String key=it.next();

Object value=props.get(key);

String methName="set"+key.substring(0,1).toUpperCase()+key.substring(1,key.length());

dynamicInvokeMethod(t,methName,value);

}

**return** t;

}

**public** Object dynamicInvokeMethod (Object obj, String methodName, Object... values)

**throws** IllegalAccessException, IllegalArgumentException, InvocationTargetException, NoSuchMethodException,

SecurityException, ClassNotFoundException {

// 动态调用指定方法，并传入参数

Class[] classes = **new** Class[values.length];

**for** (**int** i = 0; i < values.length; i++) {

classes[i] = values[i].getClass();

}

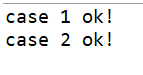
Method method = obj.getClass().getMethod(methodName, classes);

**return** method.invoke(obj, values);

}

}

1. **实验结果**



1. **实验中遇到的问题及解决方法**

这次实验中，我理解了通过读入配置文件，实现对类的动态创建的调用。

创建实例时缺少默认构造函数导致的异常：如果 clazz 没有默认构造函数，则调用 clazz.newInstance() 时会抛出 InstantiationException 异常，可以在调用前判断 clazz 是否有默认构造函数，如果没有则抛出异常或使用其他方式创建实例。