【案例12-3】：利用反射实现通过读取配置文件对类进行实例化

**【案例介绍】**

**1.案例描述**

现在有一个项目，项目中创建了一个Person类，在Person类中定义了一个sleep()方法。在工程中还定义了一个Student类继承Person类，在Student类中重写了Person类的sleep()方法。项目有一个配置文件，名称为test.properties，在配置文件中配置了一个className属性和一个methodName属性，className属性值是类的全限定类名，methodName属性值是方法名。

本案例要求通过读取配置文件对类进行实例化，具体如下：

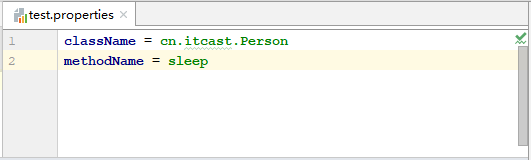
（1）获取test.properties配置文件中的className属性值（类的全限定类名），利用反射对该类进行实例化。

（2）获取test.properties配置文件中的methodName属性值（方法名），利用反射获取对象方法，并执行该方法。

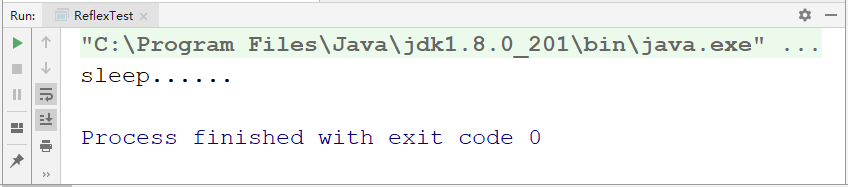
**2.运行结果**

**情景一：**

配置文件中的信息：

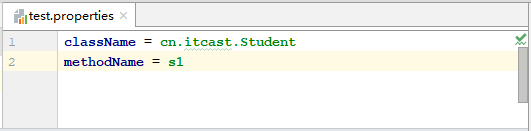


运行结果：

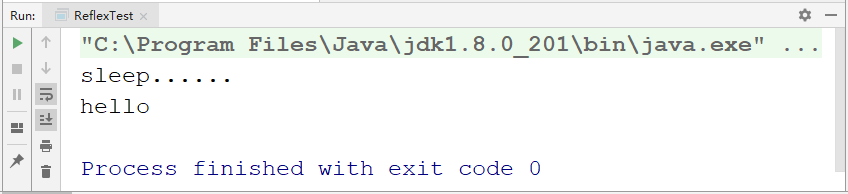


**情景二：**

配置文件中的信息：



运行结果：



**【案例目标】**

* 学会分析“利用反射实现通过读取配置文件信息信息对类进行实例化”程序任务实现的逻辑思路。
* 能够独立完成“利用反射实现通过读取配置文件信息对类进行实例化” 程序的源代码编写、编译及运行。
* 掌握反射机制的应用。

**【案例分析】**

（1）通过任务描述可知，需要先在工程的根目录下创建一个test.properties文件，在配置文件中配置一个className属性和一个methodName属性，className属性值是类的全限定类名，methodName属性值是方法名。

（2）然后创建两个类：Person类和Student类且Student类继承Person类。在Person类中编写一个sleep()方法，在Student类中重写Person类的sleep()方法；

（3）最后编写一个测试类ReflexTest，在main()方法中，编写程序，具体步骤描述如下：

1.利用反射加载配置文件

2.获取配置文件中的数据，获取类的全路径名及方法名

3.根据获取的类的全路径名，利用反射将该类加载进内存

4.创建该类对象

5.根据在配置文件中获取的方法名获取对象方法

6.执行方法

**【案例实现】**

**情景一**

test.properties

1. className = fanshe.Person
2. methodName = sleep

上述配置文件中，定义了classNmae和methodName属性并赋值。

Person.java

1. public class Person {
2. public void sleep() {
3. System.out.println("sleep......");
4. }
5. }

上述代码中，创建了一个Person类，并在Person类中编写了一个sleep方法。

Student.java

1. public class Student extends Person{
2. @Override
3. public void sleep() {
4. super.sleep();
5. System.out.println("呼噜呼噜~~~");
6. }
7. public void s1() {
8. super.sleep();
9. System.out.println("hello");
10. }
11. }

上述代码中，定义了一个Student类并继承了Person类。在Student类中重写了父类Person中的sleep()方法，然后，编写了一个s1()方法。

ReflexTest.java

1. import java.io.IOException;
2. import java.lang.reflect.InvocationTargetException;
3. import java.lang.reflect.Method;
4. import java.util.Properties;
5. public class ReflexTest {
6. public static void main(String[] args) throws IOException,
7. ClassNotFoundException, InstantiationException, IllegalAccessException,
8. NoSuchMethodException, SecurityException, IllegalArgumentException,
9. InvocationTargetException {
10. /\*1、加载配置文件
11. \* 用类名.class.getResourceAsStream("/xx")或者
12. \* 类名.class.getClassLoader().getResourceAsStream("xx");
13. \* 区别在于前者是需要反斜杠，后者不需要
14. \* \*/
15. Properties properties = new Properties();
16. properties.load(RelectTestMain.class.getResourceAsStream("/test.prope
17. rties"));
18. //2、获取配置文件中定义的数据
19. String className = properties.getProperty("className");
20. String methodName = properties.getProperty("methodName");
21. //3、加载该类进内存
22. Class cls = Class.forName(className);
23. //4、创建类对象
24. Object obj = cls.newInstance();
25. //5、获取对象方法
26. Method method = cls.getMethod(methodName);
27. //6、执行方法
28. method.invoke(obj);
29. }
30. }

上述代码中，第15-17行代码，利用反射加载了test.properties配置文件；第19-20行代码，获取配置文件中className和methodName属性的信息；第22行代码，根据获取的类的全路径名，利用反射将该类加载进内存；第24行代码，创建在配置文件中获取的类的对象；第26行代码，根据在配置文件中获取的方法名获取对象方法；第28行，执行该方法。

**情景二**

只需要修改test.properties配置文件即可。

test.properties

1. sclassName = fanshe.Student
2. methodName = s1