【案例12-1】：重写toString()方法

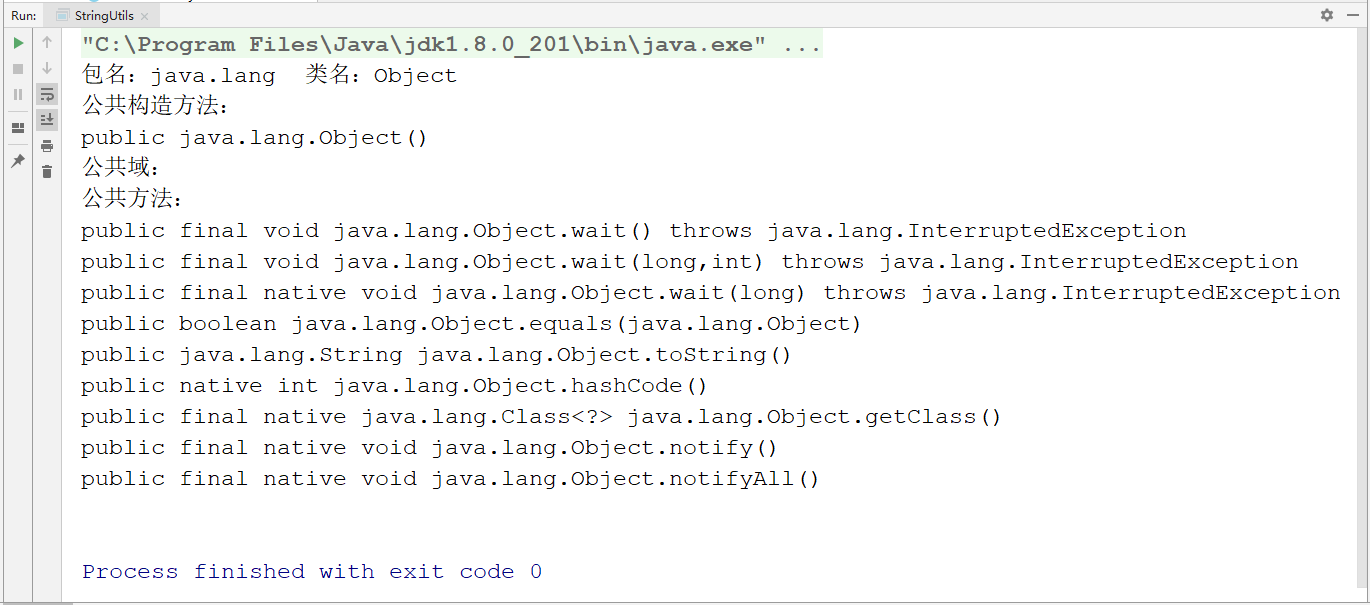
**【案例介绍】**

**1.案例描述**

为了方便输出对象，Object类提供了toString()方法。但是该方法的默认值是由类名和哈希码组成的，实用性并不强。通常需要重写该方法以提供更多的对象信息。

本案例要求使用反射重写类的toString()方法，并通过反射输出类的包名、类名、类的公共构造方法、类的公共域和类的公共方法。

**2.运行结果**



运行结果

**【案例目标】**

* 学会分析“利用反射重写toString()方法”实现的逻辑思路。
* 能够独立完成“利用反射重写toString()方法”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）通过任务的描述可知，此程序需要利用反射重写toString()方法，因此，需要先创建一个类，并在该类中定义两个方法，一个是toString()方法，用于输出类的包、类的名字、类的公共构造方法、类的公共域和类的公共方法等信息；另一个是main()方法，用来进行测试。

（2）由于是重写Object类的toString()方法，因此需要给toString()方法传递一个Object对象。

（3）由于需要利用反射输出类的包、类的名字、类的公共构造方法、类的公共域和类的公共方法，故需要先通过Object对象.getClass()获得代表该类的Class对象，再通过类的Class对象. getPackage()获得类所在的包，通过类的Class对象.getSimpleName()获得类的简单名称，通过类的Class对象.getDeclaredConstructors()获得所有代表构造方法的Constructor数组，遍历数组，判断如果是访问控制符为“public”即为公共构造方法。通过类的Class对象.getDeclaredFields()获得代表所有域的Field数组，遍历数组，判断如果是访问控制符为“public”即为公共域。通过类的Class对象.getDeclaredMethods()获得代表所有方法的Method[]数组，遍历数组，判断如果是访问控制符为“public”即为公方法。

**【案例实现】**

StringUtils.java

1. import java.lang.reflect.Constructor;
2. import java.lang.reflect.Field;
3. import java.lang.reflect.Method;
4. import java.lang.reflect.Modifier;
5. public class StringUtils {
6. @SuppressWarnings("unchecked")
7. public String toString(Object object) {
8. // 获得代表该类的Class对象
9. Class clazz = object.getClass();
10. // 利用StringBuilder来保存字符串
11. StringBuilder sb = new StringBuilder();
12. // 获得类所在的包
13. Package packageName = clazz.getPackage();
14. // 输出类所在的包
15. sb.append("包名：" + packageName.getName() + "\t");
16. String className = clazz.getSimpleName(); // 获得类的简单名称
17. sb.append("类名：" + className + "\n"); // 输出类的简单名称
18. sb.append("公共构造方法：\n");
19. // 获得所有代表构造方法的Constructor数组
20. Constructor[] constructors = clazz.getDeclaredConstructors();
21. for (Constructor constructor : constructors) {
22. String modifier =
23. Modifier.toString(constructor.getModifiers());// 获得方法修饰符
24. if (modifier.contains("public")) {// 查看修饰符是否含“public”
25. sb.append(constructor.toGenericString() + "\n");
26. }
27. }
28. sb.append("公共域：\n");
29. // 获得代表所有域的Field数组
30. Field[] fields = clazz.getDeclaredFields();
31. for (Field field : fields) {
32. String modifier = Modifier.toString(field.getModifiers());
33. if (modifier.contains("public")) {// 查看修饰符是否含“public”
34. sb.append(field.toGenericString() + "\n");
35. }
36. }
37. sb.append("公共方法：\n");
38. // 获得代表所有方法的Method[]数组
39. Method[] methods = clazz.getDeclaredMethods();
40. for (Method method : methods) {
41. String modifier = Modifier.toString(method.getModifiers());
42. // 查看修饰符是否含有“public”
43. if (modifier.contains("public")) {
44. sb.append(method.toGenericString() + "\n");
45. }
46. }
47. return sb.toString();
48. }
49. public static void main(String[] args) {
50. System.out.println(new StringUtils().toString(new Object()));
51. }
52. }

上述代码中，第50行代码的new StringUtils().toString(new Object())创建了一个StringUtils对象并调用了其toString()方法，并向toString()方法中传递了一个Object对象。第9行代码通过Object对象.getClass()获得代表该类的Class对象；第13行代码通过Object对象. getPackage()获得类所在的包；第20行代码通过Object对象. getDeclaredConstructors()获得所有代表构造方法的Constructor[]数组，第21-27行代码为遍历Constructor数组，判断如果是访问控制符为“public”即为公共构造方法。第30行代码通过类的Class对象.getDeclaredFields()获得代表所有域的Field[]数组，第31-36行代码为遍历Field数组，判断如果是访问控制符为“public”即为公共域。第39行代码通过类的Class对象.getDeclaredMethods()获得代表所有方法的Method[]数组，第40-48行代码为遍历Method[]数组，判断如果是访问控制符为“public”即为公方法。