# 反射

## 一、反射

注意：下面的各种方式的导包，都是导的reflect类型的包，即**java.lang.reflect.xxx**，很容易导成其它类型的包。

反编译：从.class文件到.java文件

通过反射机制访问java类的属性、方法、构造器

每个类在装载到虚拟机时，会生成一个对应的Class类型的对象，该对象中封装了当前类的信息。

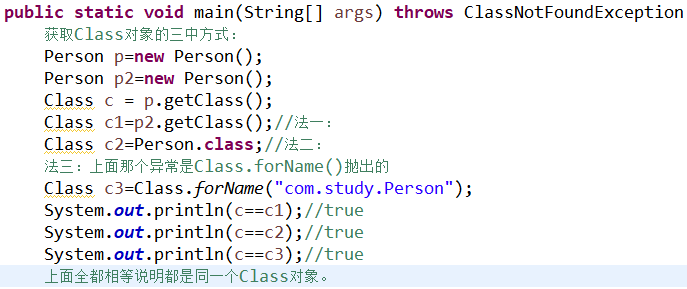
### 1.获取Class对象的三种方式

**(下面这三个都不会执行构造块和构造器，因为不是创建对象)**

**⑴类名.class**：这种方法**不会加载类**(因此不会执行静态代码块)，且**不用抛出异常**。

**⑵Class.forName(“类的全名”)：**这种**会加载类**，这种**要抛出异常**。Class.forName(String classname)，返回值为Class类型，并且括号内为String类型的类名(必须为类的全名)。**注意**，当自己改变包名或类名时，括号内的包名或类名不会自动修改，要自己手动修改，否则出现ClassNotFoundException错误。

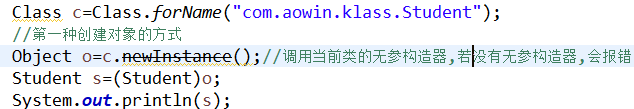
**⑶Object.getClass()：会加载类**。**一般创建一个对象，然后用对象引用.getClass()。**



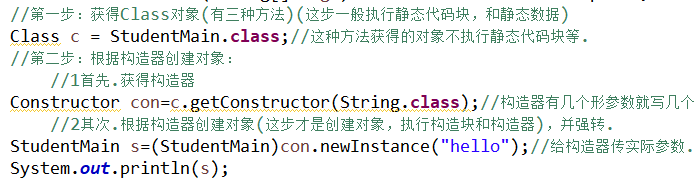
### 2.反射的方式创建对象

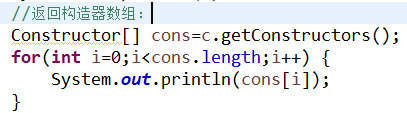
**⑴法一**：通过Class对象的**newInstance()**方法

该方法在版本9开始被弃用了

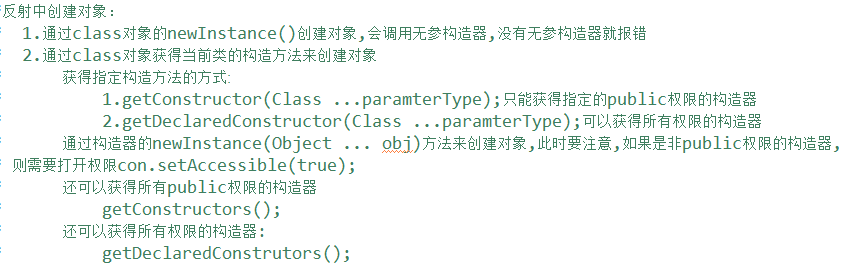


**⑵法二**：**通过反射方式获得当前类的构造器创建对象**





当类的构造器的权限为private时，要将上面的**getConstructor()**更改为**getDeclaredConstructor()**，并且接着在下面将访问权限更改为true，**con.setAccessible(true)**，该方法为对应着的Constructor的方法，其余的一样。



**getConstructor(Class...parameterTypes)**：获得当前类中的public权限的构造器。而**getDeclaredConstructor(Class...parameterTypes)**能拿到当前类中任意权限的构造器，包括private、protected和默认修饰的构造器。返回值为Constructor，括号内的参数为构造器的形参，类型为Class类型的，即用形参.class表示，如String.class。这个方法要抛出异常。

**newInstance(object...initargs)**：这里介绍构造器的newInstance()方法，Class的该方法已经被弃用了。该返回值为一个Object，即对象。括号内的参数为初始化构造器的实参，即给类的构造器传入的实际参数。这个方法要抛出异常。

**getConstructors()：**括号里没有参数，返回当前类中所有public类型的构造器，并且是一个数组的形式，注意并不会返回父类的构造器。**getDeclaredConstructors()：**则以数组的形式返回当前类中所有权限的构造器，注意并不会返回父类的构造器。这里可以不用打开权限。

### 3.反射的方式获得对象的属性

c**.getField(String name)**：注意这是Class类型的方法，返回的是Field属性，括号内的参数为字符串类型的属性名称。一般用类似Field f=c.getField(“xxx”)，接收返回的属性。该方法只能返回被public权限修饰的属性。若对象的属性被protected、默认或者private修饰，则要用**c.getDeclaredField(String name)**，其中被private修饰的还要修改访问权限，改为**f.setAccessible(true)，**该方法为对应着的Field方法，Constructor同样有这个方法。

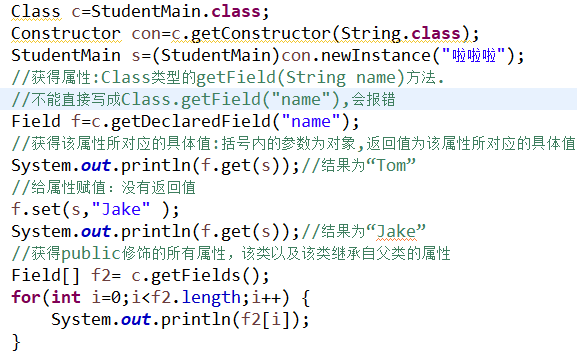
c**.getFields()**：Class类型的方法，该方法括号内没有参数，且以数组的形式返回**该类所有的public修饰的属性以及继承自父类的public的属性**。如Field[] f2=c.getFields()。**c.getDeclaredFields()**方法则以数组的形式返回该类任意权限的属性，但**不包括继承父类的属性**。这里可以不用打开权限。

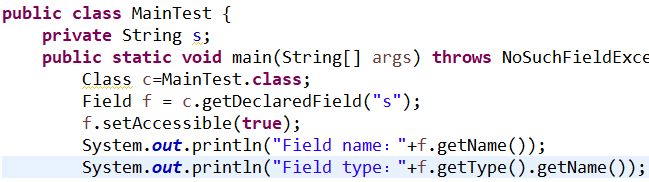
f**.get(Object obj)**：获得属性具体的值，返回的是该属性所对应的具体的属性值。括号内的参数为对象。该方法为Field的方法。

f**.set(Object obj,Objec value)**：给属性赋值。括号内的参数，第一个为对象，第二个为给对应的属性赋具体的值。该方法属于Field，没有返回值。

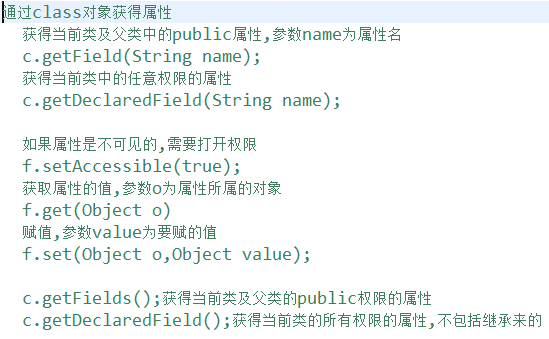
Field**.getName()**：返回值为String类型，返回的是这个属性的名字，如private String s，返回的是s。

Field**.getType()**：返回值为一个Class对象，该对象用来标识这个属性对象所声明的类型。







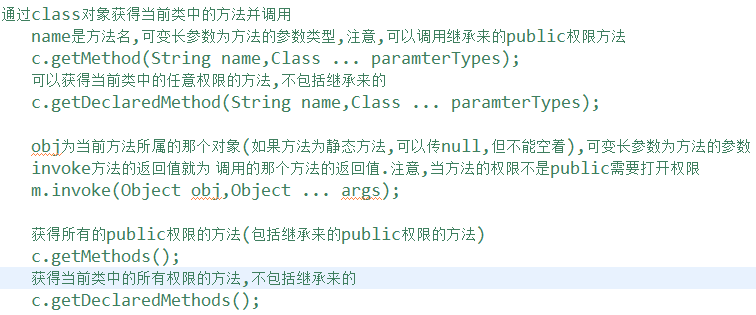


### 4.反射的方式获得对象的方法

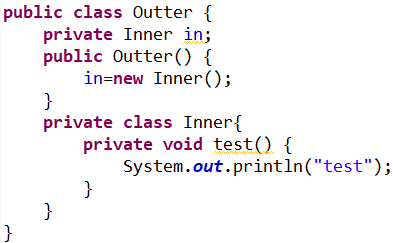
c**.getMethod(String name, class...parameterTypes)**：Class类型的方法，返回值为Method类型。括号内的参数：String name(以字符串的形式表示方法名)、class...parameterTypes(为方法内的形参以.class表示)，若没有形参，则可以不能写。如：Method m=c.getMethod(“func1”, String.class)。该方法还能够调用父类的方法。该方法只能调用public修饰的方法，可以是**本类的方法或者继承自父类的方法**。而c**.getDeclaredMethod(String name, class...parameterTypes)**则可以调用本类中任意权限修饰的方法，但是**不能是继承来的方法**。如果调用的是private修饰的方法，则要修改进入的权限，**m.setAccessible(true)**。

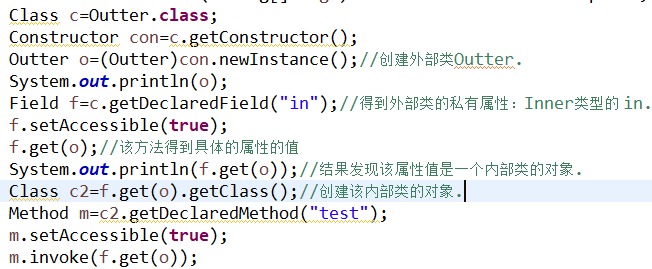
m**.invoke(Object obj, Object...args)**：Method类型的方法，**该方法是调用对象的方法**，返回值为调用方法的返回值。括号内的参数：Object obj(为对象)、**Object..args(为方法的可变长的实参)**。**若是静态方法，则可以传对象也可以传null，但是不能为空着**。invoke方法的返回值就是调用那个的方法的返回值，若调用的该方法没有返回值，则结果为null。

c**.getMethods()**：括号内没有参数，返回一个Method类型的数组。获得当前类及其父类的所有public权限修饰的方法。如Method[] m=c.getMethods()。c**.getDeclaredMethods()**则获得当前类中所有权限修饰的方法，**不包括继承自父类的方法**。这里可以不用打开权限。



**例子：访问内部类的方法**





### 5.isInstance和instanceof

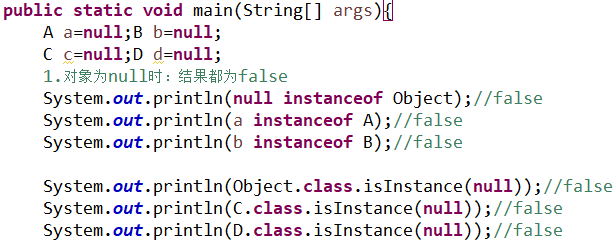
**isInstance(Object obj)**：**是Class类的一个方法**。用法：**B.class .isInstance(b)**。返回值为boolean类型，括号内的参数为对象。该方法**用于判断参数对象是否属于对应的类(可以是本身类也可以是父类(类或者接口))**，若属于则为true，否则为false。

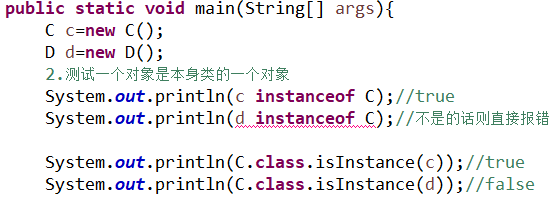
**instanceof**关键字：**是一个操作符。**用法：**object instanceof Object**。**用于判断对象是否是这个类的实例或者这个类的子类的实例**。注意这里的**of是小写**的。如果object是这个类或子类的实例，则返回true，否则返回false。只要object为null，结果都是false。









上面报错的原因：显示不兼容C类和D类。

