1. 反射获取普通方法
2. 获取方法

在进行反射处理的时候也可以通过反射来获取之中的全部方法，但是需要提醒的是如果反射调用这些方法必须有一个前提条件:类之中必须提供有实例化对象。

Class类里面提供如下方法:

·获取全部方法：public Method[] getMethods​()

throws SecurityException

·获取指定方法：public Method getMethod​(String name,

Class<?>... parameterTypes)

throws NoSuchMethodException,

SecurityException

·获取本类指定方法：public Method getDeclaredMethod​(String name, Class<?>... parameterTypes)

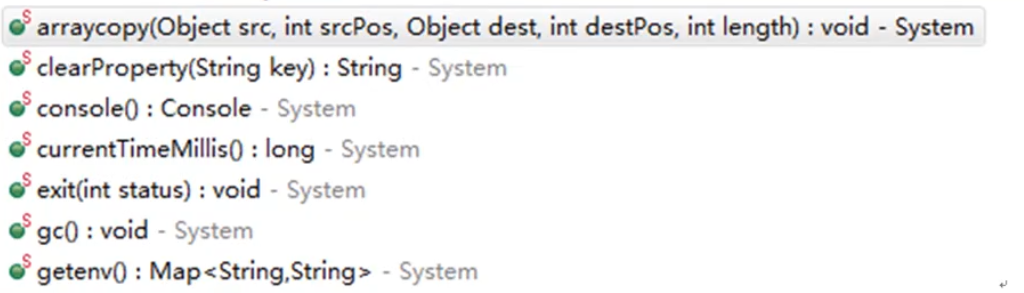
throws NoSuchMethodException,

SecurityException

* 获取本类全部方法：public Method[] getDeclaredMethods​()throws SecurityException

范例:获取全部方法

|  |
| --- |
| package com.company.反射机制;  import com.company.反射机制.entity.Person;  import java.lang.reflect.Method;  public class JavaDemo4 {  public static void main(String[] args) {  Class<?> aClass = Person.class;  { //获取全部方法包括父类的方法  Method[] method = aClass.getMethods();  for (Method temp:method) {  System.*out*.println(temp);  }  }  System.*out*.println("\_\_\_\_\_\_\_\_");  {  //获取本类方法  Method[] methods = aClass.getDeclaredMethods();  for (Method temp:  methods) {  System.*out*.println(temp);  }  }  } } |

但是需要注意的是，这个时候的方法信息获取是依靠Method类之中提供的toString（）方法完成的。而我们可以自定义拼凑方法信息的展示信息。

这种方法我们只需要知道能够通过反射获取方法结构即可，不需要做过多的深入了解，但是Method中有一个非常重要的方法:

·public Object invoke(Object obj，Object… args);

|  |  |
| --- | --- |
| **Class Method**  [java.lang.Object](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/lang/Object.html)  [java.lang.reflect.AccessibleObject](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/lang/reflect/AccessibleObject.html)  [java.lang.reflect.Executable](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/lang/reflect/Executable.html)  java.lang.reflect.Method | public [Object](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/lang/Object.html) invoke​([Object](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/lang/Object.html) obj, [Object](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/lang/Object.html)... args) throws [IllegalAccessException](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/lang/IllegalAccessException.html), [IllegalArgumentException](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/lang/IllegalArgumentException.html), [InvocationTargetException](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/lang/reflect/InvocationTargetException.html) |

在Person类里面为name 属性追加getter setter方法。

|  |
| --- |
| **public class Person extends Abstract Base implements IChannlMessage ,IMessage{**  **//其它操作略  private String name ;  public Person() {  System.*out*.println("-----Person构造方法------");  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }**  **//其它操作略**  **}** |

现在我们通过反射机制来实现Person类中getter 与 setter 方法的调用处理。

范例:在不导入指定（Person）开发包的情况下实现属性的配置。

|  |
| --- |
| **package com.company.反射机制;  import com.company.反射机制.entity.Person ;  import java.lang.reflect.Method;  public class JavaDemo5 {  public static void main(String[] args) throws Exception {  Class<?> aClass = Class.*forName*("com.company.反射机制.entity.Person");  String value = "小强";  Object instance = aClass.getDeclaredConstructor().newInstance();  String setMethodName = "setName";  Method setMethod = aClass.getDeclaredMethod(setMethodName, String.class);  setMethod.invoke(instance , value);  String getMethodName = "getName" ;  Method getMethod = aClass.getDeclaredMethod(getMethodName);  Object invoke = getMethod.invoke(instance);  System.*out*.println(invoke);    } }** |