package package5;/\*\*

\* package5

\* RunDi

\*/

import java.lang.reflect.Field;

/\*\*

\* @program: RunDi

\* @description:

\* @param:

\* @return:

\* @author zzhd

\*\*/

/\*\*

\* 1. getField(String name) 与 getFields()

\* ①类名.getField(String name) : 获取类特定的方法，参数name指定了属性的名称

\* ②类名.getFields() : 获取类中public类型的属性，返回一个包含某些Field对象的数组，

\* 该数组包含了此Class对象所表示的类或接口的所有可访问的公共字段

\* 区别:

\* ---> getField(name),带参方法，返回的是类特定的方法

\* getFields(),不带参方法，返回Field对象数组，

\* ---> 异常不同

\*

\* 2. getDeclaredFields() 与 getDeclaredField(String name)

\* ①类名.getDeclaredField(String name) : 获取类特定的方法，参数name指定了属性的名称

\* ②类名.getDeclaredFields() : 获取类中所有属性(public protected default private),

\* 单不包括继承的属性，返回Field对象的一个数组

\* 区别:

\* ---> getDeclaredField(name),带参方法，返回的是类特定的方法

\* getDeclaredFields(),不带参方法，返回Field对象数组，

\* ---> 异常不同

\*

\*

\*/

public class UserTest {

public static void main(String[] args) {

Class c = User.class; // 获取User下的类

System.out.println(c); // class package5.User

User user = new User("校长");

// 写法一

// for(Field field : c.getDeclaredFields()){

// //getDeclaredFields() : 获取类中所有的属性(public、protected、default、private)，

// 但不包括继承的属性，返回 Field 对象的一个数组

// System.out.println(field.isAccessible()); // false

// field.setAccessible(true);

// try {

// System.out.println(field.getName() + " " +field.get(user));

// }catch (IllegalAccessException e){ // 非法访问异常：

// e.printStackTrace();

// }catch(IllegalArgumentException e){ // 非法参数异常：

// e.printStackTrace();

// }catch (NullPointerException e){ // 空指针异常

// e.printStackTrace();

// }catch (ExceptionInInitializerError e){ // 初始化异常

// e.printStackTrace();

// }

// }

// 写法二

// try {

// Field[] f = c.getDeclaredFields();

// for (int i = 0; i < f.length; i++) {

// System.out.println(f[i].isAccessible());

// f[i].setAccessible(true);

// System.out.println(f[i].getName() + " " + f[i].get(user));

// }

// }catch (Exception e){

// e.printStackTrace();

// }

// 写法三

try {

// Field[] f1 = c.getFields();

try {

Field f2 = c.getDeclaredField("sName"); // 编号①

// Field f3 = c.getDeclaredField("method1"); // 编号①

// Field f4 = c.getDeclaredField("iAge"); // 编号①

System.out.println(f2); // private java.lang.String package5.User.sName

// System.out.println(f3); // java.lang.NoSuchFieldException: method1

// System.out.println(f4); // public int package5.User.iAge

// System.out.println(f4.isAccessible()); // false

System.out.println(f2.isAccessible()); // 编号②

f2.setAccessible(true); // 编号③

f2.setAccessible(false);

System.out.println(f2.isAccessible());

System.out.println(f2.getName() + " " + f2.get(user)); // 编号④

// System.out.println(f4.getName() + " " + f4.get(user)); // 编号、④ iAge 0

/\*\*

\* 编号解析：

\* 编号①：根据输出结果分析，该语句是用来获取参数sName的属性(public protected default private)

\* + 参数所存在的位置

\* 其中参数是类中的成员属性，不能是方法名，否则报错java.lang.NoSuchFieldException: method1

\* 编号②：isAccessible() 方法

\* ---> 返回值类型为boolean

\* "初理解" ---> 当isAccessible的返回值为false时，那么我们就可以认为不能访问该反射出来的属性值

\* "实验" ---> 将getDeclaredField(String name)参数name的实参类型设置为public类型时，其返回值也是false

\* "如何得到true" ---> 通过Field对象名.isAccessible(true)进行强行设置为true，得到其返回值为true

\* "二次实验" ---> 将getDeclaredField(String name)参数name的实参类型设置为private类型，

\* 不使用Field对象名.isAccessible(true)进行设置，结果报错，

\* "二次实验报错" ---> can not access a member of class package5.User with modifiers "private"

\* 无法访问类package5的成员。修饰符为“private”的用户

\* "结论" ---> Field对象名.isAccessible()的返回值为false时，并不是不能访问此成员属性(变量)

\* 当返回值为true时，表示反射的对象在使用时应该取消了Java语言访问检查

\* 当返回值为false时，表示反射的对象在使用时应该实施了Java语言访问检查

\* ---> 那么实际上isAccessible只是启用了或禁用了访问安全检查的开关，并不是为true就能

\* 访问，为false就不能访问

\* ---> 由于JDK的安全检查耗时较多，所以通过isAccessible(true)的方式关闭安全检查就可以

\* 达到提升反射速度的目的。

\*/

}catch (Exception e){

e.printStackTrace();

}

}catch (SecurityException e){

e.printStackTrace();

}

}

}