Integer的源码解读

# Integer的自动装箱

在Integer内部封装了一个**Integer数组**，默认的Integer值为-128至127；

在这个范围的int值，自动装箱成Integer时，同int值对应相同的Integer对象。

但是对于new Integer(int)生成的Integer对象时不同的对象。

题目：

/\*\*

\*判断-128 -- 127的Integer对象是否相等

\* Integer内部封装了-128到127的Integer池

\*/

@Test

public void test1( ){

int[] nums = {-129,-128,1,127,128,129};

//通过new的方式创建：全是false

for(int num :nums){

**integerObjectIsTheSameNew**(num);//全是false

}

//自动装箱：-128到127是返回的是一个对象，其他的通过new实现

for(int num:nums){

**integerObjectIsTheSameAutoBoxing**(num);//根据大小而定

}

}

/\*\*

\* 自动装箱：调用的是Integer.valueOf方法：valueOf源码解读一下

\*/

private void integerObjectIsTheSameAutoBoxing(int i) {

Integer a = i;

Integer b = i;

System.out.println("自动装箱："+ i +" --> "+( a == b ));//true or false?

}

/\*\*

\* 通过new Integer()创建对象

\* @param i

\*/

private void integerObjectIsTheSameNew(int i) {

Integer a = new Integer(i);

Integer b = new Integer(i);

System.out.println("new的方式："+ i +" --> "+( a == b ));//true or false?

}

结果：

new的方式：-129 --> false

new的方式：-128 --> false

new的方式：1 --> false

new的方式：127 --> false

new的方式：128 --> false

new的方式：129 --> false

自动装箱：-129 --> false

自动装箱：-128 --> true

自动装箱：1 --> true

自动装箱：127 --> true

自动装箱：128 --> false

自动装箱：129 --> false

# 题目2

/\*\*

\* 定义一个swap函数，实现交换a和b的值，a、b值在-128和127范围内。

\* 由于Java中都是值传递，Integer做形参时接收到只是Integer的intValue。

\*/

@Test

public void test3() {

Integer a = 1;

Integer b = 2;

System.out.println("a = " +a+"; b = "+b);

swap(a,b);//十分暴力，修改了Integer内部封装的Integer数组-128至127

System.out.println("a = " +a+"; b = "+b);

for(int num = 1;num<4;num++){

Integer i = num;

System.out.println(" num = "+num+"; Integer num = " + i.intValue());

}

}

/\*\*

\* 通过反射机制实现交换两个值,只适用于-128至127的整型；

\* 因为只有-128至127的Integer是保持同一个对象，即形参与实参是同一个对象

\* @param b

\*/

private void swap(Integer a, Integer b) {

Field field = null;

try {

field = Integer.class.getDeclaredField("value");

field.setAccessible(true);//强攻

int temp = a.intValue();

field.set(a, b);

field.setInt(b, temp);

} catch (Exception e) {

e.printStackTrace();

}

}

结果：

a = 1; b = 2

a = 2; b = 1

**num = 1; Integer num = 2**

**num = 2; Integer num = 1**

num = 3; Integer num = 3

通过结果看出，利用反射机制暴力改变了Integer内部维护的Integer数组的值。