链滴

Java 五道输出易错题解析（避免小错误）

作者：Ethan

原文链接：https://ld246.com/article/1523514265274

来源网站：链滴

<img src="https://ld246.com/images/img-loading.svg" alt="" data-src="https://b3logfile
com/bing/20180214.jpg?imageView2/1/w/960/h/520/interlace/1/q/100">
收集了几个易错的或好玩的 Java 输出题，分享给大家，以后在编程学习中稍微注意下就 OK 了

<h3 id="1-看不见的空格-"><a href="https://ld246.com/forward?goto=http%3A%2F%2Fwww.
nblogs.com%2Flanxuezaipiao%2Fp%2F4170157.html%231" target="_blank" rel="nofollow ug
">1. 看不见的空格？</h3>
下面的输出会正常吗？
package basic;
public class IntegerTest {
<pre><code class="highlight-chroma"><span class="highlight
cl">public static void main(String\[\] args) {
    System.out.print
n(Integer.parseInt("1"));
    System.out.print
n(Integer.parseInt("2"));
</code></pre>
}
}
解析：将上面代码复制下（不要自己手敲）在自己的环境里运行看看，是不是输出下面错误来了

<blockquote>
1
Exception in thread “main” java.lang.NumberFormatException: For input string: “2”
at java.lang.NumberFormatException.forInputString(Unknown Source)
at java.lang.Integer.parseInt(Unknown Source)
at java.lang.Integer.parseInt(Unknown Source)
at basic.IntegerTest.main(IntegerTest.java:7)
</blockquote>
竟然说第二条语句有问题，表面上完全看不出来任何问题是不是！
实际上这里的错误原因涉及到一个概念 —零宽度空格，可能有人接触过，但相
更多的人甚至都没听过，什么是零宽度空格？它实际上是一个 Unicode 字符，是一个空格
关键是它没有宽度，因此我们一般肉眼看不到。但可以在 vim 下看到，上面的第二条语句
的 2 前面就有一个零宽度空格，放到 vim 中打开后你会发现是下面这样的语句：
System.out.println(Integer.parseInt("2"));
Unicode 规范中定义，每一个文件的最前面分别加入一个表示编码顺序的字符，这个字符的名
叫做”零宽度非换行空格“（ZEROWIDTHNO-BREAKSPACE），用 <code>FEFF</code> 表示。
正好是两个字节，而且 FF 比 FE 大 1。因此下面的语句会输出 65279，刚好是 <code>FEFF</code
。
System.out.println((int)"2".charAt(0));
<h3 id="2-类静态成员初始化"><a href="https://ld246.com/forward?goto=http%3A%2F%2Fw
w.cnblogs.com%2Flanxuezaipiao%2Fp%2F4170157.html%232" target="_blank" rel="nofollow
ugc">2. 类静态成员初始化</h3>
下面的程序能编译通过么？如果通过，说结果并解释，不能编译，说错误原因。
class A
{
public static int X;
static { X = B.Y + 1;}
}
public class B
{
public static int Y = A.X + 1;
static {}
public static void main(String[] args) {

System.out.println("X = "+A.X+", Y = "+B.Y);
}
}
解析：这个程序能正确运行，类的运行过程如下：
<blockquote>
首先加载主类 B，初始化静态成员 Y，发现需要类 A 的信息，于是加载类 A，初始化静态成员 X 也用到 B 类信息，由于此时 B 类的 Y 还未成功加载因此这里是默认值 0，从而到 A 类的 X 为 1，然后返回到 B 类，得到 Y 为 2。
</blockquote>
<h3 id="3-装箱拆箱的实际过程">3. 装箱拆箱的实际过程</h3>
关于自动装箱，相信大部分人都明白是怎么一回事，但真的完全明白了嘛？
先看下面的代码：
Short s1 = 1;
Short s2 = s1;
System.out.println(s1 == s2);
谁都知道当然打印 true 了。现在加一句试试：
Short s1 = 1;
Short s2 = s1;
s1++;
System.out.println(s1 == s2);
还是 true 吗？No，这次输出成了 false。WHY？难道 s1 和 s2 引用的不是同一个对象吗？有这疑问的说明你对自动装箱拆箱的过程还不是非常清楚，实际上上面的代码可以翻译为下面的代码（实执行过程，要掌握）：
Short s1 = new Short((short)1);
Short s2 = s1;
short tempS1 = s1.shortValue();
tempS1++;
s1 = new Short(tempS1);
System.out.println(s1 == s2);
哦，原来如此，这下明白了，因此我们在使用自动装箱的时候小心点为妙。
<h3 id="4-你自以为是的异常">4. 你自以为是的异常</h3>
先来两句代码：
NullTest myNullTest = null;
System.out.println(myNullTest.getInt());
相信很多人看到这段代码时，都会自以为是的说：<code>NullPointerException</code>。果如此吗？你还没看到 NullTest 这个类是如何定义的呢。现在看看这个类的定义：
class NullTest {
public static int getInt() {
return 1;
}
}
发现 <code>getInt()</code> 方法体没有任何类变量和类方法的使用，因此这里会正常输出 1.
记住：类变量和类方法的使用，仅仅依赖引用的类型。即使引用为 null，仍然可以调用。从良好践的角度来看，明智的做法是使用 <code>NullTest.getInt()</code> 来代替 <code>myNullTest.gtInt()</code>，但谁不不能保证不会碰到这样的代码，因此还是小心为妙。
<h3 id="5-变长参数和数组-如何变通-">5. 变长参数和数组，如何变通？</h3>
变长参数特性带来了一个强大的概念，可以帮助开发者简化代码。不过变长参数的背后是什么呢

Basically，就是一个数组。
public void calc(int... myInts) {}
calc(1, 2, 3);
编译器会将前面的代码翻译成类似这样：
int[] ints = {1, 2, 3};
calc(ints);
不过这里有两点需要注意：
- 当心空调用语句，这相当于传递了一个 null 作为参数。
calc();
等价于
int[] ints = null;
calc(ints);
- 当然，下面的代码会导致编译错误，因为两条语句是等价的：
public void m1(int[] myInts) { ... }
public void m1(int... myInts) { ... }