Java基础篇-基本类型

java学习都是从基本类型开始的，日常工作中基本类型也是经常用的东西。不过基本类型的知识点确认已经掌握了吗？

来看看那接下来的几个问题吧

boolean类型的长度是多少？

void是基本类型吗？

int i=0;Integer j=I;这个装箱过程中，基本类型是怎么赋值给引用类型的？

基本类型存储在栈上，什么时候存储在堆上呢？

这些问题的答案包含在接下来的内容里，让我们一块来研究下吧

1、什么是基本类型

Java中对象的类型分为基本类型和引用类型，那么怎么区分基本类型和引用类型呢。

常用简单区分引用类型和基本类型得方法是：基本类型存储在栈上；引用类型是引用存储在栈上，数据存储在堆上。根据数据存储位置来区分类型。这种方法实际上存在一些错误。

看下接下来的代码：

public void demo1()  
{  
 int no1=0;  
}

no1存储在demo1方法的方法栈中，这个没有什么疑问，可是接下来的情况呢

public class Demo2 {  
 int no2 =1;  
}

首先Demo2是引用类型，所以实例化对象demo2的数据存储位置是堆，no2是int类型，如果no2数据存储在栈中，demo2是如何找到栈中的no2呢，如果这里还有一个引用指向no2的引用吗？显然是不可能这么设计的，所以no2的实际存储位置是堆中。

所以，基本类型的存储位置分为这两种情况：

1、基本类型作为方法中的局部变量时，存储在方法栈上；

2、作为对象属性的时候，对象存储在堆上，那么基本类型就会存储在堆上。（**这里强调对象存储在堆上，这个涉及编译优化的，暂不详述**）

个人观点，区分基本类型和引用类型的关键还是它们的数据存储结构，

1、基本类型数据结构组成：创建时分配的存储空间，并且存储的是基本类型包含的值；2、引用类型的数据结构组成：栈上的引用和堆上的数据，在堆上分配的存储空间，存储的是对象的值，在栈上分配的存储空间，存储引用指向堆中的地址。

现在弄清楚了基本类型是什么，接下来具体说一下基本类型吧

2、基本类型的分类、长度和默认值

基本类型分为四类：整形、布尔型、泛型和字符型

它们的情况如下表

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 长度（字节） | 默认值 | 取值范围 |
| byte | 1 | 0 | -128～ 127 |
| short | 2 | 0 | -32768 ～ 32767 |
| int | 4 | 0 | -2147483648～2147483647 |
| long | 8 | 0 | -9223372036854774808～9223372036854774807 |
| float | 4 | 0.0 | 3.402823e+38～1.401298e-45（e+38 表示乘以10的38次方，而e-45 表示乘以10的负45次方） |
| double | 8 | 0.0 | 1.797693e+308～4.9000000e-324 |
| char | 2 | \u000 |  |
| boolean | 官方没有明确定义长度 | flase | true false |

其中值得研究一下的是boolean，因为官方没有明确给出boolean长度，实际的长度是由相应的虚拟机来设计的。boolean表示是、否两种情况，只需要一位就可以表示。在jvm中基本类型分为：数值类型、boolean类型和returnAddress三种类型。boolean作为一种单独的类型，个人认为它是一种比较特殊的情况。

不过在jvm中的最小单位是字节，也就是说boolean不会少于8位，比较让人接受的说法是，boolean在编译过程中使用int表示，即占用4个字节。而boolean数组中使用一个字节表示。

说完了基本类型的长度和存储，接下来说一下基本类型的使用吧

3、拆箱和装箱

拆箱，即把包装类型转换为基本类型

装箱，即把基本类型转换为包装类型

拆箱和装箱是在日常代码中经常用到的，下面代码，就是一个拆箱和装箱的过程

public void demo()  
{  
 //装箱  
 Integer no1 = 10;  
 //拆箱  
 int no2 = no1;  
}

这种代码已经很熟悉了，不过no1赋值10这个过程中，一个基本类型是怎么变成引用类型的呢？

这就要从java代码的语法糖说起了

语法糖：指计算机语言中添加的某种语法，这种语法对语言的功能并没有影响，但是更方便程序员使用。（Java中有很多被加糖的语法，如for循环、String的拼接等等，有兴趣的话可以了解下。）

也就是说Integer no1 = 10;这行代码被加糖了，具体做了什么呢。很简单反编译一下class文件看下这行代码的真实面目就好了，没有加糖前的代码是这样的

Integer no1 = Integer.*valueOf*(10);

int no2 = no1.intValue();

这下就明白了拆箱和装箱过程的整个过程了

这里还有一个关于包装类型的比较巧妙地设计，来看下valueOf这个函数

public static Integer valueOf(int i) {  
 if (i >= IntegerCache.*low* && i <= IntegerCache.*high*)  
 return IntegerCache.*cache*[i + (-IntegerCache.*low*)];  
 return new Integer(i);  
}

IntegerCache.*low 是-128，*IntegerCache.*hig是127，也就是说，如果int值是-128-127之间的时候，返回的结果是*IntegerCache.*cache这个缓存池中的对象，如果不在这个范围内就返回一个新的Integer对象。*

*所以面试过程中，这个经常会问*

Integer no1 = 10;  
Integer no2 = 10;  
Integer no3 = 200;  
Integer no4 = 200;  
  
System.*out*.println(no1==no2);  
System.*out*.println(no3==no4);

上边这块代码输出结果是什么？