Java引进了八个基本类型,来支持数值计算,Java这么做的原因主要是工程上的考量,因为使用基本类型能够在执行效率以及内存使用两方面提升软件性能.

1.1 Java虚拟机的boolean类型

Java语言规范中,boolean类型的值只有两种可能,它们分别是"true"和"false"来表示,

在Java虚拟机规范中,boolean类型则被映射成int类型,具体来说,"true"被映射为整数1,而"false"被映射为整数0.这个编码约束了Java字节码的具体实现.

对于存储boolean数组的字节码,Java虚拟机许哟啊保证实际存入的值是整数1或者0.

类型	值域	默认值	虚拟机内部符号
boolean	{false, true}	false	Z
byte	[-128, 127]	0	B
short	[-32768, 32767]	0	S
char	[0, 65535]	'\u0000'	C
int	[-2^31, 2^31-1]	0	I
long	[-2^63, 2^63-1]	0L	J
float	~[-3.4E38, 3.4E38]	+0.0F	F
double	~[-1.8E308, 1.8E308]	+0.0D	D

Java的基本类型都有对应的值域和默认值,可以看到,byte,short,int,long,float以及double的值域一次扩大,而且前面的值域被后面的值域所包含,因此,从前面的基本类型转换至后面的基本类型,无需强制转换,尽管他们的默认值看起来不一样,但在内存中都是0.

boolean和char是唯二的无符号类型.char是非负数.

声明为byte,char,short的局部变量,可以存储超出他们取值范围的数值,

Java的浮点数采用IEEE 754浮点数格式,以float为例,浮点类型通常由两个0,+0.0F以及-0.0F.前者在Java里是0,后者是符号位为1,其他位为0的浮点数.在内存中等同于十六进制整数0x8000000.尽管他们的内存数值不同,但是在Java中+0.0F == -0.0F会返回真.

有了+0.0F和-0.0F这两个定义后,我们可以定义浮点数中的正无穷和负无穷.正无穷就是任意正浮点数(不包括+0.0F)除以+0.0F得到的值,而负无穷是任意正浮点数除以-0.0F得到的值,在Java中,正无穷和负无穷是有确切的值,在内存中分别等同于十六进制整数0x7f800000和0xff800000.NaN除了"!="始终返回true,所有其他比较结果都会返回false.

1.1 Java基本类型的大小

Java虚拟机每调用一个Java方法,便会创建一个栈帧,这种栈帧有两个主要的组成部分,分别是局部变量区,以及字节码的操作数栈,这里的局部变量是广义的,除了普遍意义下的局部变量之外,他还包含实例方法的"this指针"以及方法所接收的参数.

在Java虚拟机规范中,局部变量区等价于一个数组,并且可以用正整数来索引.除了long,double值需要两个数组单元来存储之外,其他基本类型以及引用类型的值均占用一个数组单元.

也就是说boolean,byte,char,short这四种类型,在栈上占用的空间和int是一样的,和引用类型也是一样的,因此在32位虚拟机HotSpot中,这些类型在栈上将占用4个字节;而在64位的HotSpot中,他们将占用8个字节.

当然，这种情况仅存在于局部变量，而并不会出现在存储于堆中的字段或者数组元素上。对于 byte、char 以及 short 这三种类型的字段或者数组单元，它们在堆上占用的空间分别为一字节、两字节，以及两字节，也就是说，跟这些类型的值域相吻合。

因此，当我们将一个 int 类型的值，存储到这些类型的字段或数组时，相当于做了一次隐式的掩码操作。举例来说，当我们把 0xFFFFFFFF（-1）存储到一个声明为 char 类型的字段里时，由于该字段仅占两字节，所以高两位的字节便会被截取掉，最终存入“\uFFFF”。

boolean 字段和 boolean 数组则比较特殊。在 HotSpot 中，boolean 字段占用一字节，而 boolean 数组则直接用 byte 数组来实现。为了保证堆中的 boolean 值是合法的，HotSpot 在存储时显式地进行掩码操作，也就是说，只取最后一位的值存入 boolean 字段或数组中。

讲完了存储，现在我来讲讲加载。Java 虚拟机的算数运算几乎全部依赖于操作数栈。也就是说，我们需要将堆中的 boolean、byte、char 以及 short 加载到操作数栈上，而后将栈上的值当成 int 类型来运算。

对于 boolean、char 这两个无符号类型来说，加载伴随着零扩展。举个例子，char 的大小为两个字节。在加载时 char 的值会被复制到 int 类型的低二字节，而高二字节则会用 0 来填充。

对于 byte、short 这两个类型来说，加载伴随着符号扩展。举个例子，short 的大小为两个字节。在加载时 short 的值同样会被复制到 int 类型的低二字节。如果该 short 值为非负数，即最高位为 0，那么该 int 类型的值的高二字节会用 0 来填充，否则用 1 来填充。