# JVM中的基本数据类型

版本：2018/9/9-1(18:00)



## 问题汇总

> 1. 总结基本数据类型相关所有可能的问题，用于自我检测和查漏补缺。
> 2. 【☆】标记的是补充问题，直接给答案。其余问题答案都在文中。

1. Java语言规范中如何定义boolean?
2. JVM规范是如何定义boolean类型的?
3. JVM规范约束了Java字节码的具体实现，导致编译成的class文件中，除了字段和传入参数，看不到boolean类型的痕迹
4. JVM中基本类型的值域和默认值
5. Java有哪些基本数据类型？(8种)
6. Java中所有的基本数据类型都是有符号的，并且没有无符号数?
7. Java中只有两个基本类型是无符号类型?
8. char如果用负数去赋值会出现什么情况?
9. 无符号类型有什么用?
10. Java的浮点类型采用IEEE 754浮点数格式
11. Java中如何获取到-0.0F?
12. 如何定义浮点数中的正无穷?
13. 如何定义浮点数中的负无穷?
14. Java中的正无穷和负无穷在内存中的具体值是什么?
15. 如果一个数超过了正无穷和负无穷的范围，有属于什么浮点数呢?
16. 标准的NaN是什么?

17. NaN的比较?
18. 解释器的解释栈帧(Interpreted frame)
19. JVM规范中局部变量区等价于一个数组，并且可以使用正整数来索引
20. boolean、byte、char、short在栈上所占用的字节数?
21. 堆中的字段或者数组元素中，数据类型的所占空间
22. 将int类型的值存储到byte、char、short等类型中是如何操作的?
23. boolean字段和boolean数组存储的特殊性
24. JVM如何进行算数运算?
25. boolean、char的运算伴随着零扩展
26. byte、short的运算伴随着符号扩展
27. 【☆】为什么boolean、byte、char、short在解释器栈上(局部变量区)和堆上的空间不一样?
    > 局部变量区中会浪费空间，主要是因为变长数组不好控制，浪费空间，但是可以统一使用下标来计算地址。

28. 【☆】Java中8种基本数据类型的默认值在内存中都是0，如何具体区分是哪个类型?
    > 内存中不区分，Java程序将其解释为什么类型就是什么类型。

# boolean类型(4)

1、Java语言规范中如何定义boolean?

> 1. boolean的取值只有两种：true或者false
> 2. 然而这两个符号无法被虚拟机直接使用

2、JVM规范是如何定义boolean类型的?

> 1. JVM中boolean类型被映射为int类型
> 2. true映射为整数1
> 3. false映射为整数0

3、JVM规范约束了Java字节码的具体实现，导致编译成的class文件中，除了字段和传入参数，看不到boolean类型的痕迹

4、如何通过asmtools更改字节码?

```
public class Main {
    public static void main(String c[]) {
        boolean flag = true;
        if(flag) System.out.println("Hello, Java!");
        if(flag == true) System.out.println("Hello, JVM!");
    }
}
```

> 1-正常运行

```
// 编译成Main.class文件
javac .\Main.java
// 运行Main
java Main

//显示结果
Hello, Java!
Hello, JVM!
```

> 2-更改字节码

```
$ java -cp /path/to/asmtools.jar org.openjdk.asmtools.jdis.Main Foo.class > Foo.jasm.1
$ awk 'NR==1,/iconst_1/{sub(/iconst_1/, "iconst_2")} 1' Foo.jasm.1 > Foo.jasm
$ java -cp /path/to/asmtools.jar org.openjdk.asmtools.jasm.Main Foo.jasm
$ java Foo

// 显示结果
Hello. Java!
```

> 3-if（flag）比较时ifeq指令做是否为零判断，常数2仍为true，打印输出.
> 4-if（true == flag）比较时if_cmpne做整数比较，iconst_1是否等于flag，比较失败，不再打 印输出

# 基本数据类型(14)

1、JVM中基本类型的值域和默认值

| 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 |

> 1. 基本类型的默认值在内存中都是0
> 2. 前面的基本类型转换到后面的基本类型，不需要强制转换！

2、Java有哪些基本数据类型？(8种)

> boolean
> byte
> short
> char
> int
> long
> float
> double

3、Java中所有的基本数据类型都是有符号的，并且没有无符号数?

> 错误！有两个特例

4、Java中只有两个基本类型是无符号类型?

> 1. boolean, 0~1
> 2. char, 0~65535

# char

5、char如果用负数去赋值会出现什么情况?

> 1. 会提示需要强制类型转换
> 2. 例如-1，实际的值就是65535。会从另一头开始算。
> 3. Java编译器生成的字节码会遵守JVM规范对编译器的约束。
> 4. 不要担心变量会超出取值范围。

```
char c = (char)-1;
```

6、无符号类型有什么用?

> 1. char可以认定为非负数。
> 2. 可以用做数组的索引。

# float

7、Java的浮点类型采用IEEE 754浮点数格式

> 1. float有两个0，一个是+0.0F,一个是-0.0F
> 2. +0.0F在Java内存中是0
> 3. -0.0F在Java内存中是0x8000000(符号位1，其余为0)
> 4. +0.0F == -0.0F, Java中结果为true

8、Java中如何获取到-0.0F?

> Float.iniBitsToFloat(0x8000000)

9、如何定义浮点数中的正无穷?

> 任意正浮点数（不包括 +0.0F）除以 +0.0F得到的值

10、如何定义浮点数中的负无穷?

> 任意正浮点数除以 -0.0F 得到的值

11、Java中的正无穷和负无穷在内存中的具体值是什么?

> 1. 正无穷在内存中等同于十六进制整数 0x7F800000(8位)
> 2. 负无穷在内存中等同于十六进制整数 0xFF800000

12、如果一个数超过了正无穷和负无穷的范围，有属于什么浮点数呢?

> 1. 比如0x7F80 0001，对应的都是NaN(Not-a-Number)

13、标准的NaN是什么?

> 1. +0.0F/+0.0F = 0x7FC0 0000, 这就是标准的NaN
> 2. 其余的都是不标准的NaN

```
java.lang.Float.NaN;
java.lang.Double.NaN;
```

14、NaN的比较?

> 1. NaN只有在"!="时会返回true
> 2. 其余的比较都会返回false。

# 基本类型的大小

1、解释器的解释栈帧(Interpreted frame)

> 1. JVM每调用一个Java方法，就会创建一个栈帧。
> 2. 解释器使用的解释帧栈有两个主要的组成部分：局部变量区，字节码的操作数栈
> 3. 该局部变量区除了一般的局部变量外，还包含实例方法的this指针，和实例方法所接受的参数

2、JVM规范中局部变量区等价于一个数组，并且可以使用正整数来索引

> 1. long、double需要两个数组单元来存储。
> 2. 其他的基本类型和引用类型的值均占用一个数组单元。
> 3. boolean、byte、char、short这四种类型在栈上所占用的空间和int是一样的。和引用类型所占空间也是一样的。

3、boolean、byte、char、short在栈上所占用的字节数?

1. 在局部变量中的情况
2. 和int以及引用类型所占空间一样
3. 32位 JVM中是4个字节
4. 64位 JVM中是8个字节

4、堆中的字段或者数组元素中，数据类型的所占空间

1. byte是1个字节
2. char是2个字节
3. short是2个字节

5、将int类型的值存储到byte、char、short等类型中是如何操作的?

1. 存储到这些类型和数组时，相当于做了一次隐式的掩码操作。
2. 0xFFFF FFFF(-1)存储到char类型的字段时，高两位的字节会被截取掉，会存入 `\uFFFF`

6、boolean字段和boolean数组存储的特殊性

1. HotSpot中boolean字段占用一个字节
2. boolean数组采用byte数组实现
3. 为了保证堆中boolean值的合法性，HotSpot存储时进行显式的掩码操作，只取最后一位的值存入到boolean字段和数组中

# 算数运算

1、JVM如何进行算数运算?

1. JVM的算数运算都依赖于 操作数栈
2. 堆中的boolean、byte、char、short加载到操作数栈上
3. 然后将栈上的值当做int类型进行运算

2、boolean、char的运算伴随着零扩展

1. boolean、char是无符号类型
2. 加载时将char的两字节的值复制到int类型的低二字节，高二字节补0

3、byte、short的运算伴随着符号扩展

1. byte、short为有符号类型
2. 如果是正数，高字节用0填充
3. 如果是负数，高字节用1填充

# 参考资料