Java原始数据类型与包装类面试题

基础概念

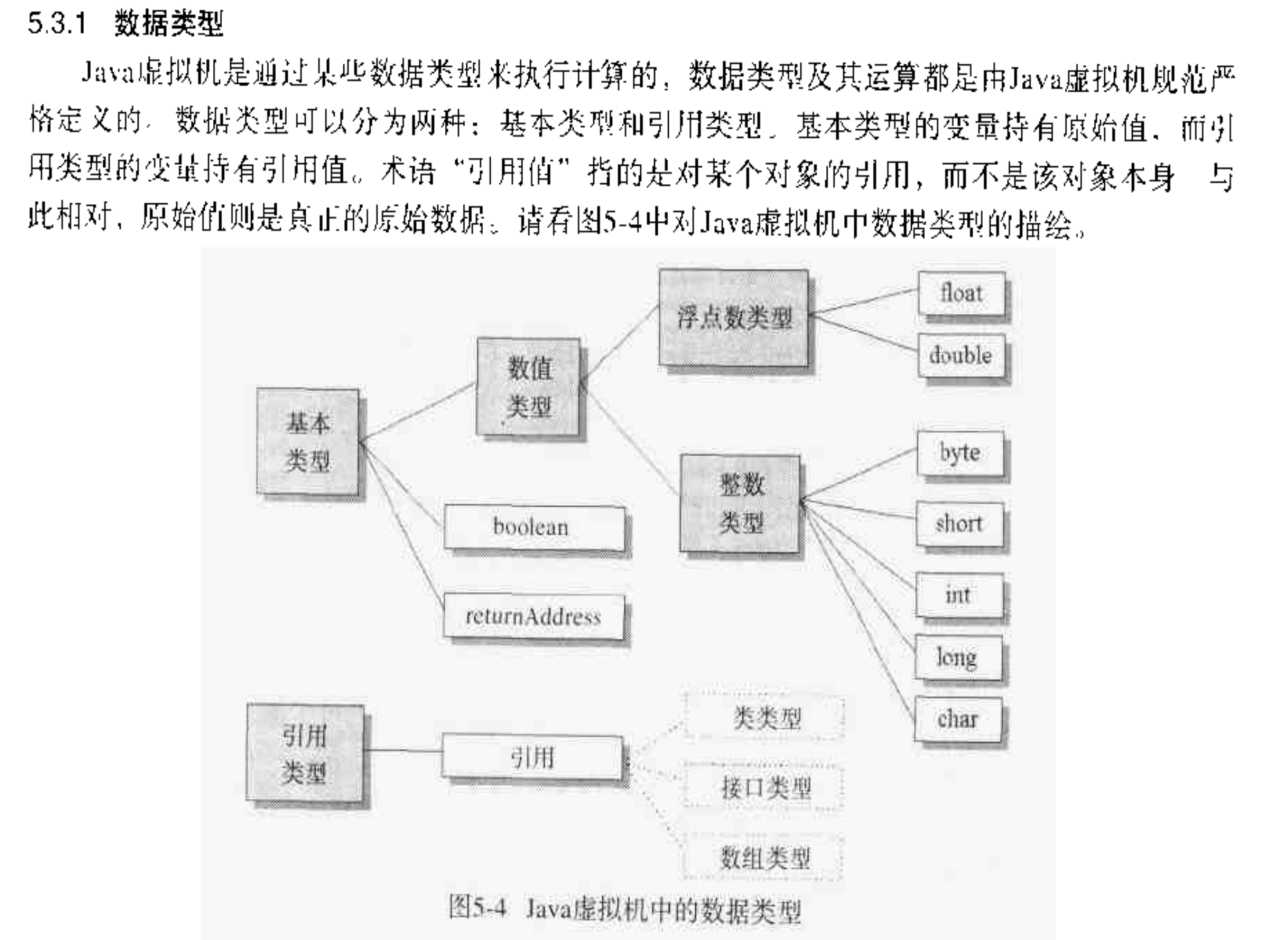
**Java中有哪些原始数据类型？**

整数类型 : byte short int long

浮点型 : double float

布尔类型 : boolean

字符类型 : char



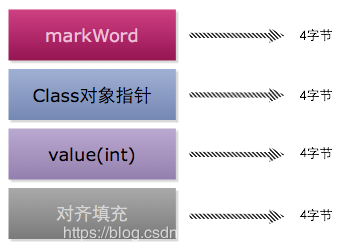
**什么是包装类？为什么需要它们？**

//因为泛型和集合不支持基本数据类型，所以只能用包装类（用来将基本数据打包成类）列出所有原始类型对应的包装类

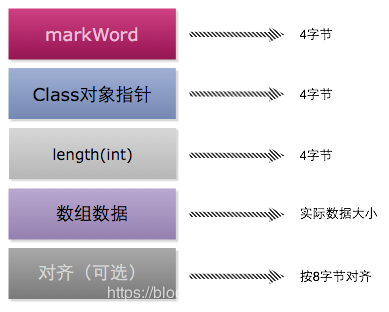
**为什么包装类是基础类型的四倍？**

[java对象头 MarkWord-CSDN博客](https://blog.csdn.net/qq_26542493/article/details/90938070)

Integer只有一个int类型的成员变量value，所以其对象实际数据部分的大小是4个字节，然后再在后面填充4个字节达到8字节的对齐，所以可以得出Integer对象的大小是16个字节。



对象的内存结构，需要注意数组的内存结构和普通对象的内存结构稍微不同，因为数据有一个长度length字段，所以在对象头后面还多了一个int类型的length字段，占4个字节，接下来才是数组中的数据



**自动装箱和拆箱是什么？**

装箱指自动把基本数据类型包装成对应的包装类，拆箱指的是把包装类转换成相对应的基本数据类型

**自动装箱是在哪个Java版本引入的？**

JDK1.5；Java SE5

**数值类型相关问题**

**int和Integer有什么区别？**

1. 初始值区别：Int初始值为0，Integer初始值为null。
2. 存储方式的区别：基本类型在栈中直接存储的具体数值，而包装类型则存储的是堆中的引用,包装类型需要占用更多的内存空间（堆中占一块内存，栈中也占用一块内存）*。*
3. 占用大小的区别int占用4个字节大小固定。除了存储一个整数值外，它还需要额外的内存来存储对象的头信息。在64位JVM中，对象头通常占用16字节；而在32位JVM中，对象头可能占用8字节（近似值，具体大小还和JVM的实现细节有关）。
4. 包装类使用==判断的是是否引用同一个对象即包装类的值。
5. 存储逻辑的区别：Integer进行自动装箱时，如果数字不在 -128 至 127这个范围之内，会创建新的对象。如果数字在 -128 至 127 之间时，会直接使用缓存中的对象，从IntegerCache中获取，不用再创建新的对象。

**Integer.valueOf(127) == Integer.valueOf(127)的结果是什么？为什么？**

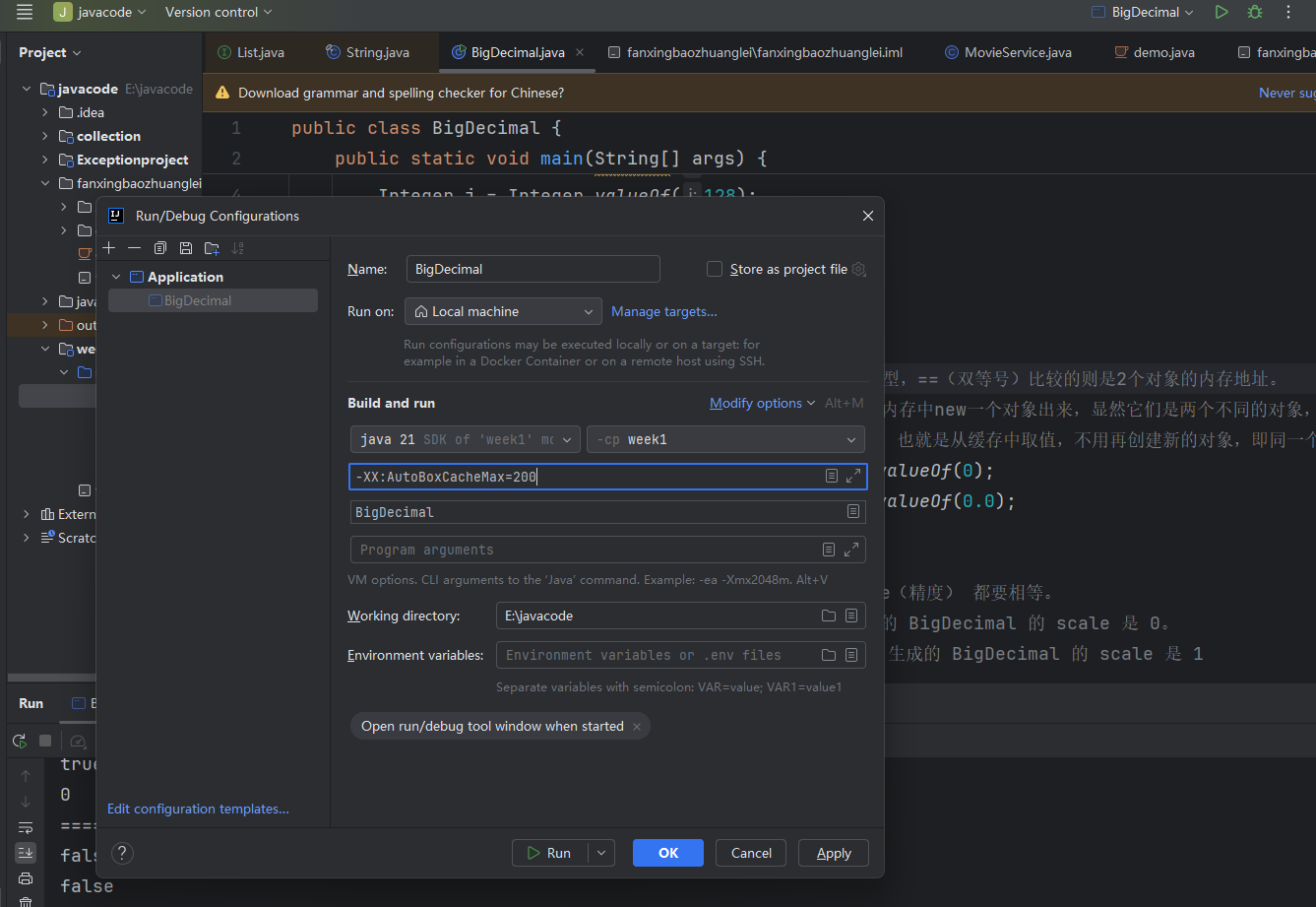
//应该是true因为对于包装类型，==（双等号）比较的是2个对象的内存地址。此时同一个数对应的地址应该是一样的。

**Integer.valueOf(128) == Integer.valueOf(128)的结果是什么？为什么？**

//应该是false，当赋的基本数据类型值不在[-128, 127]之间，包装类会去Java堆内存中new一个对象出来，显然它们是两个不同的对象，所以结果false；

**如何修改Integer缓存的上限？**

Idea中点击run->edit configurations->modify options 选中add vmoption 输入-XX:AutoBoxCacheMax=200



**Float和Double的缓存行为与Integer相同吗？**

Float和Doulbe进行自动装箱时，返回的是new出来的一个对象。没有缓存机制。

**布尔类型相关问题**

**Boolean.TRUE == Boolean.valueOf(true)的结果是什么？**

True，因为Boolean.valueOf(true) 会返回 Boolean.TRUE 这个实例。Boolean.TRUE 是一个被缓存的 Boolean 对象，用于表示 true 值，而 Boolean.valueOf(true) 方法也是返回这个已经被缓存的缓存的 Boolean.TRUE。

**Boolean b = new Boolean(true)和Boolean b = true有什么区别？**

Boolean b = new Boolean(true)会一直创建一个为true的新对象；直接引用的是 Boolean 类内部缓存的 Boolean.TRUE 对象，不会创建新的对象。

**Boolean类中缓存了哪些值？**

True/false

**字符类型相关问题**

**Character类中有哪些常用的静态方法？**

isLetter(char c)：检查字符是否是字母。

isDigit(char c)：检查字符是否是数字。

isWhitespace(char c)：检查字符是否为空格、制表符或换行符。

isUpperCase(char c)：检查字符是否为大写字母。

isLowerCase(char c)：检查字符是否为小写字母。

toUpperCase(char c)：将字符转换为大写字母。

toLowerCase(char c)：将字符转换为小写字母。

**char可以存储中文字符吗？为什么？**

可以因为Unicode字符集里面包含了中文

**类型转换与运算**

**int i = Integer.parseInt("123")和int i = Integer.valueOf("123")有什么区别？**

Integer.parseInt("123") ：调用会创建一个新的 int 值（在栈内存中），但如果只是单纯的数值计算和存储，基本没有内存管理开销。

Integer.valueOf("123")创建一个value值在-128~127之间的Integer对象时，直接从缓存中拿到这个对象，所以value值相同的Integer对象都是对应缓存中同一个对象。虽然这里返回的是integer对象但是赋给了一个int所以会自动拆箱。

**如何将字符串"123.45"转换为double？**

double d1 = Double.parseDouble(s4);

double d2 = Double.valueOf(s4);

**Integer i = null; int j = i; 会发生什么？**

报错因为包装类的null不能赋值给int类型

**byte b = 130; 是否正确？为什么？**

byte b = 130; 是不正确的，因为 130 超出了 byte 类型的取值范围。它占用 1 个字节（8 位）的存储空间。byte 类型的取值范围是 -128 到 127。

**short s = 1; s = s + 1; 和 short s = 1; s += 1; 哪个能编译？为什么？**

变量 s 是 short 类型，而赋值操作符 = 左右两边的类型必须一致。

由于 s + 1 为了匹配+1所以会自动升级成 int 类型，直接将其赋值给 short 类型的变量 s 会导致类型不匹配，因此编译器会报错。可以使用强转解决short s=1；s=（short）（s+1）；

后面的可以编译。使用+=java会自动进行类型匹配。相当于强转s=（short）（s+1）。

**内存与性能**

**使用包装类会带来什么性能开销？**

除了存储一个数值以外，包装类还需要额外的内存来存储对象的头信息。在64位JVM中，对象头通常占用16字节；而在32位JVM中，对象头可能占用8字节（近似值，具体大小还和JVM的实现细节有关）。

**在什么情况下应该优先使用原始类型而不是包装类？**

不允许null值时，使用原始类型节省内存；如果允许null值，则必然要用包装类，用到泛型，就需要用包装类。

**为什么Java集合不能直接存储原始类型？**

集合中的所有元素都必须是对象类型，这样可以保证集合操作的一致性和通用性。如果允许存储基本数据类型，会破坏一致性。存放包装类也添加了许多可以使用的方法。

**特殊值处理**

**Double.NaN表示什么？如何检测一个值是否为NaN？**

Double.NaN 表示“不是一个数”（Not a Number），它是 double 类型的一个特殊值，用于表示没有意义的浮点数运算结果。例如，0.0 / 0.0 或 Math.sqrt(-1) 都会得到 NaN。

检测一个值是否为 NaN，可以使用 Double.isNaN() 方法

**Double.POSITIVE\_INFINITY和Double.NEGATIVE\_INFINITY是什么？**

正无穷大和负无穷大

**Integer可以表示null，而int不能，这有什么影响？**

影响其默认值，integer为null int为0。

**比较与相等**

**Integer a = 1000, b = 1000; System.out.println(a == b); 输出什么？**

False，因为超过-128~127新建了两个不同对象存储a，b

**如何正确比较两个包装类对象的值是否相等？**

使用equals方法

**new Integer(1).equals(new Long(1))的结果是什么？**

false因为他们类型不同euqals方法会比较类。

**位运算与特殊方法**

**Integer.bitCount()方法的作用是什么？**

计算一个整数转换成二进制数之后1的个数

**Integer.reverse()方法的作用是什么？**

计算一个整数转换成二进制数之后将其逐位反转后变成的整数

**Double.compare(0.0, -0.0)的结果是什么？**

返回1,compare会优先比较正负。

**进制转换**

**如何将整数转换为二进制字符串？**

Integer.toBinaryString(int i)方法（除二取余）

**如何将整数转换为十六进制字符串？**

Integer.toHexString(int i)方法（除十六取余）

**如何将二进制字符串"1010"转换为整数？**

Integer.parseInt(String s, int radix)方法 int radix 可以指定转成的进制这里设置为2

**其他实用方法**

**Integer.rotateLeft()方法的作用是什么？**

Integer.rotateLeft(num, 2)将整数的二进制数位数左移低位用高位移出的数补移几位就是乘以2^几次方

**Character.isJavaIdentifierStart()方法的作用是什么？**

该方法用于判断指定的字符是否可以作为 Java 标识符的起始字符。例如

Character.isJavaIdentifierStart(“3”)输出false

**Boolean.logicalAnd()方法的作用是什么？**

与门操作

Boolean.logicalAnd(true, false)输出0

Boolean.logicalAnd(true, true)输出1

**常量与极值**

**Integer.MAX\_VALUE的值是多少？**

Integer.MAX\_VALUE 是最大的 int 类型正整数，等于 2 ^31 −1。

**Integer.MIN\_VALUE的值是多少？**

Integer.MIN\_VALUE 是最小的 int 类型负整数，等于 −2 ^31

**Double.MAX\_VALUE和Double.MIN\_VALUE有什么区别？**

Double.MAX\_VALUE：它表示的是 double 类型能存储的最大正数。

Double.MIN\_VALUE：是 double 类型能表示的最小正非零值。它表示的是 double 类型能存储的最小正数，但不是最负的负数。

**字符串转换**

**Integer.toString(123, 8)的结果是什么？**

173将123转换成8进制的字符串

**如何将Integer对象格式化为前面补零的4位字符串？**

使用String a=String.format("%04d", 123)

**哈希与相等**

**new Integer(1).hashCode()的结果是什么？**

new Integer().hashCode()方法是根据该 Integer 对象的值来生成哈希码。实际上返回的就是integer的值1。返回类型为int。

**？为什么重写equals()方法时通常也要重写hashCode()方法？**

**如果不重写hashCode（）如果存在相同的值那么不同对象的equals方法输出的结果还是false因为还是调用了object对比地址。**

在 Java 中，hashCode() 方法用于生成对象的哈希码，而 equals() 方法用于判断两个对象是否相等。如果重写了 equals() 方法但没有重写 hashCode() 方法，可能会导致两个逻辑上相等的对象（根据 equals() 判断）具有不同的哈希码。这会违反 hashCode() 和 equals() 的一致性原则，从而在使用哈希表（如 HashMap、HashSet）时出现问题。例如，两个相等的对象应该具有相同的哈希码，否则它们可能无法正确地存储和检索。

**序列化**

**？包装类是否实现了Serializable（这个接口是？）序列化跟io操作有关接口？**

是的，所有的包装类（如 Integer、Double、Boolean 等）都实现了 Serializable 接口。这意味着它们的对象可以被序列化，即将对象的状态转换为字节流以便存储或传输。

**类型系统**

**包装类的继承结构是怎样的？**

Integer、Double、Long 等都继承自 Number。Character 类继承自 Object

**Number类有哪些抽象方法？**

byte byteValue()：将此 Number 对象转换为 byte 类型。

double doubleValue()：将此 Number 对象转换为 double 类型。

float floatValue()：将此 Number 对象转换为 float 类型。

int intValue()：将此 Number 对象转换为 int 类型。

long longValue()：将此 Number 对象转换为 long 类型。

short shortValue()：将此 Number 对象转换为 short 类型

**Java 8新增**

**Java 8为包装类添加了哪些新方法？**

Integer.compare(int x, int y)：比较两个 int 值。

Long.compare(long x, long y)：比较两个 long 值。

Long.sum(long a, long b)：返回两个 long 值的和。

**Long.sum()方法的作用是什么？**

返回两个 long 值的和。