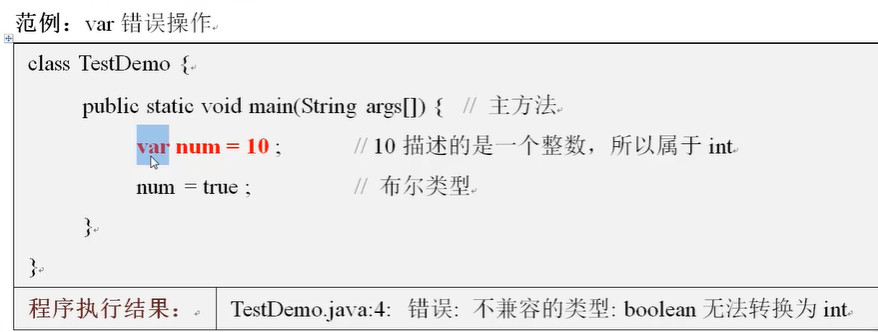
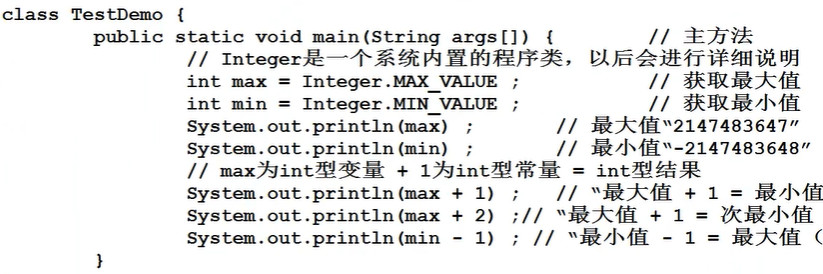


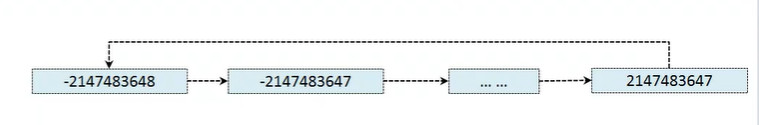
随着技术的发展，希望出现自动推测出数据类型，在Java1.10以后出现了一个var关键字，它可以根据值自动定义出该数据的类型，但是这个关键字在Java中是一个比较鸡肋的东西，因为，它在第一次给变量赋值以后，推测出数据的类型，就不能更改其数据类型：（而向JavaScript中的var是可以随意改变类型的所以其也是一门弱类型语言）



关于数据溢出：解决的方法扩大数据范围



原因：



先数据类型转化再计算，范围小到范围大得可以自动转化，范围大得到范围小得需要强制转化，最好不要强制转化。

有时会出现这样的情况：long a=12234234324;(这是一个超过int范围的数字)，Java中是不允许的，因为其默认是int型，所以会报错（数据过大），它是先将定义一个int常量，然后赋值给a。所以正确的定义要加上L进行申明。

Java中byte型值表示：



这种表示更符合人类思维，而且计算机在计算时也很方便：比如-128加1,二进制相加就能明显得到结果。也算Java与c得有无符号之分吧。

（！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！

***由于byte得特殊，Java对其进行了一些优化：***

***如果直接将一个在byte范围内的“数字常量值”赋给byte型变量，那么Java会进行自动转化，其他的情况则不会。***

***）***

浮点型：Java中所有的小数默认为double型，注意虽然同为64位，但是double要比long大的多。



字符型直接参与整型运算，并且转化为整型（相应的字符编码值，Java使用的Unicode）。

数字字符：

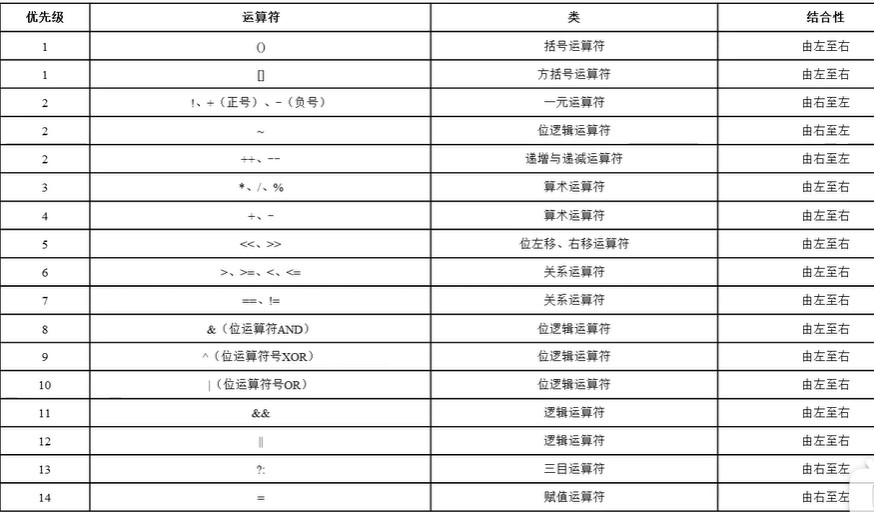
48-57（0-9）

字母字符：

65-90（A-Z）

97-122（a-z）

运算符优先级（所有的优先级最好都是通过括号来修改）：



简化运算符的功能可以节约执行的空间：\*=、-=、+=、--、++、三目运算符

逻辑运算符并不会比较类型，不同类型的数据会先转化为统一类型，如果转化失败就会报错。

布尔运算符：

&与：无论前面是真是假都会计算两边的判断条件//一假就假

|或：无论前面的条件是真是假都会计算两边的判断条件//一真就真

&&短路与：如果前面为假则直接返回假不会计算后面的条件。

||短路或：如果前面为真则直接返回真不会计算后面的条件

位运算符：<< 、>>、 &、 |、 ^也是可以用来节约内存空间。主要用于文件的加密

Switch的匹配，因为字符型能自动转换为整型所以也能用数字匹配字符。

Do while基本不会用。

明确知道循环的次数使用for，明确知道循环的条件使用while。

Continue也能实现标记跳转（类似 goto）