Bit shifting

Monday, June 7, 2021 5:44 PM

Java has 3 bit-shift operators: <<, >>, >>>

<< left shift

take some number 37

in binary: 32 4 1
10 0101

37 << 3 when you left shift, insert 0's

256 32 8
10 0101 000 == 296

Note that <<3 is same as $\times 2^3$
37 × 8 = 296

if you wanted to store the result in a byte, Java will chop off any bits past 8

~~1~~ 0010 1000

32 8
0010 1000 == 40

but if you store the same rusult in an int, Java chops off bits past 32 (in this case no effect)

0000 0000 0000 0000 0000 0001 0010 1000

>>> right shift logical

- this same as << but it shifts to the right
- always inserts 0's on the left

Ex1 37 >>> 4

32 4 1
10 0101

00 0010 ~~0101~~

2
00 0010 == 2

Ex2 165 >>> 4

128 32 4 1
1010 0101

0000 1010 ~~0101~~

8 2
0000 1010 == 10

>> right shift arithmetic

- preserves the sign of the number
- if the left-most bit is a [1, inserts 1's; 0, inserts 0's]

Ex1

37 >> 4

(Assume 37 is stored in a byte)

32 4 1
0010 0101
↑left-most is a 0

0000 0010 | 0101

0000 0010 == 2

for positive numbers, >> is the same as >>>

Ex2 165 >> 4

(Assume 165 is stored in a byte)

128 32 4 1
1010 0101 → this is actually -91 in two's complement
↑left-most is 1

1111 1010 | 0101

128 64 32 16 8 4 2 1
1111 1010 == 250

but in two's complement == -6

1111 1010
flip 0000 0101
+1
0000 0110
-6

1010 0101
flip 0101 1010
64 16 8 2 +1
0101 1011
-91

Note that -6 is $-91/2^4$ but rounded UP instead of DOWN. (in this case UP = more negative)

>> 4 is same as dividing by 2^4