Integer types

Principle:
A primitive operation of type **int** must yield an **int.**

type **int**:

values: $-2^{31}, \ldots, 2, -1, 0, 1, 2, \ldots, 2^{31}-1$

$-2{,}147{,}483{,}648$ .. $2{,}147{,}483{,}647$

operations: – b, b + c, b – c, b * c, b / c, b % c

6 / 4 is not 1.5. It could be 1 or 2.

int division truncates toward zero, so

6 / 4 = 1 and –6 / 4 = –1 .

b % c yields the remainder when b is divided by c.

So 6 % 4 = 2 and 6 % 3 = 0.

Principle: A primitive operation of type **int** must yield an **int.**

type **int**:
values: -2^{31}, …, 2, –1, 0, 1, 2, …, $2^{31}-1$
–2,147,483,648 .. 2,147,483,647

Names for smallest and largest **int** values:

Integer.MIN_VALUE Integer.MAX_VALUE

What could be the value of this expression?

Integer.MAX_VALUE + 1

Look at the Principle at the top of this page, and list some possibilities:

ERROR? Integer.MAX_VALUE 0

The Java designers decided it should be

Integer.MIN_VALUE

"wraparound"

Use Gries/Gries, Chap. 6 pp. 216–220 as a reference!!!

type **byte**: space used: 08 bits

values: $-2^7, \ldots, 2, -1, 0, 1, 2, \ldots, 2^7-1$

Byte.MIN_VALUE –128
Byte.MAX_VALUE 127

type **short**: space used: 16 bits

values: $-2^{15}, \ldots, 2, -1, 0, 1, 2, \ldots, 2^{15}-1$

Short.MIN_VALUE –32768
Short.MAX_VALUE 32767

type **int**: space used: 32 bits

values: $-2^{31}, \ldots, 2, -1, 0, 1, 2, \ldots, 2^{31}-1$

Integer.MIN_VALUE –2,147,483,648
Integer.MAX_VALUE 2,147,483,647

type **long**: space used: 64 bits

values: $-2^{63}, \ldots, 2, -1, 0, 1, 2, \ldots, 2^{63}-1$

Long.MIN_VALUE –9,223,372,036,854,775,808
Long.MAX_VALUE 9,223,372,036,854,775,807