

Why integer overflow "wraps around"

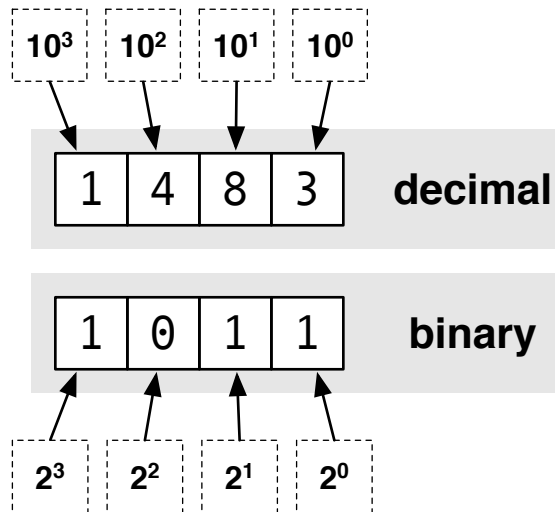
handout for CS 302 by Will Benton (willb@cs)

The whole-number types in Java (e.g. `int`, `long`, etc.) have limited ranges. Of course, any type representable in a finite computer has a limited range, but you're likely to actually run into the limits of the Java primitive types before you have to deal with a number that you can't represent on a computer.

Manipulating a variable so that its value would exceed the range of its type results in *integer overflow*. When this happens, the value of the variable "*wraps around*" to the opposite end of the range, just as a classic video game character might wrap around to the other side of the screen upon crossing an edge.

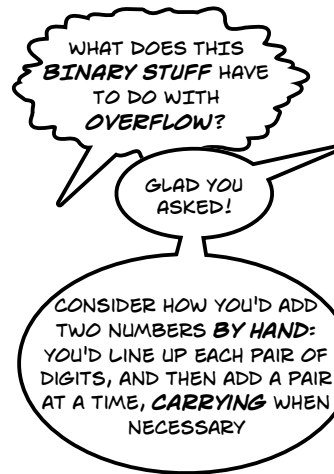
This handout will explain why this happens. To do so, we'll first need to consider how computers represent numbers. (If you find this interesting, you'll enjoy a computer organization class!)

Computers represent numbers in *binary*, or base-2. Humans typically deal with numbers in *decimal*, or base-10. Both kinds of numbers have "places," in which a digit denotes some quantity of a power of the base. Let's examine two different four-digit numbers to see the difference:



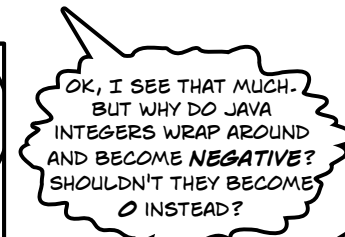
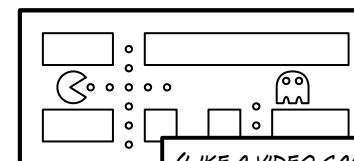
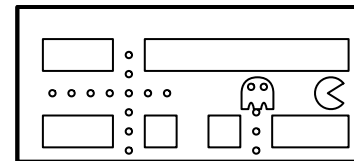
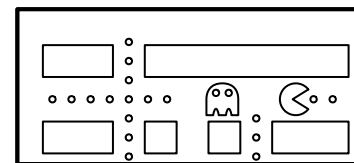
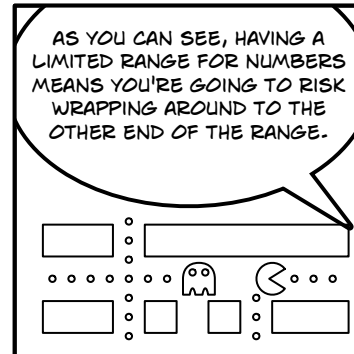
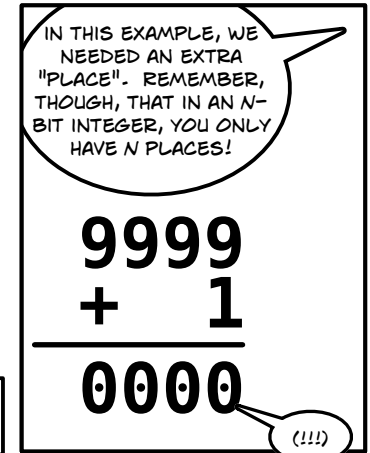
The decimal number has the value 1483: $1 * 10^3 + 4 * 10^2 + 8 * 10^1 + 3 * 10^0$. The binary number has the value 11: $1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0$.

Note that with n bits, you can represent a number up to $b^n - 1$, where b is the base.



$$\begin{array}{r} 9999 \\ + 1 \\ \hline 10000 \end{array}$$

(*ADDITION BASICALLY WORKS THE SAME WAY IN BINARY.)



THE WAY THAT JAVA REPRESENTS NEGATIVE NUMBERS IS CALLED TWO'S COMPLEMENT

IN TWO'S COMPLEMENT, YOU NEGATE A NUMBER BY COMPLEMENTING EACH DIGIT AND ADDING ONE....

SURE!

$$\begin{array}{l} 7_{10} == 0111_2 \\ -7_{10} == 1001_2 \\ -8_{10} == 1000_2 \end{array}$$

(YEAH, SURE, BUT I STILL DON'T GET IT.)

WELL, LET'S TALK ABOUT HOW JAVA REPRESENTS NEGATIVE NUMBERS IN BINARY; THAT SHOULD CLEAR THINGS UP!

(*NOTE THAT THIS ALLOWS AN N-BIT NUMBER TO ASSUME A RANGE OF VALUES FROM -2^{N-1} TO $+2^{N-1}$.)

-9999

SO IF WE'RE USING 4-BIT NUMBERS, WHAT HAPPENS WHEN WE ADD 1 TO 7?

WE REPRESENT NEGATIVE NUMBERS IN DECIMAL BY PREFIXING THEM WITH A - SIGN. HOWEVER, THERE'S NO PLACE FOR SUCH A SIGN IN A BINARY NUMBER!

$$\begin{array}{r} 0111 \\ + 0001 \\ \hline 1000 \text{ OVERFLOW!} \end{array}$$