Java Primitive Numeric Types

- byte
- short
- int
- long
- float
- double

Note: Not to be confused with boxed types - Byte, Short, Integer, Long, Float & Double.
Overflow

An overflow is a phenomenon that occurs when an operation causes a number to get over it's bounds.

Ex.

```java
byte b = 60;
b *= 3; // Should be 180, but byte can only store up to +127
System.out.println("b is " + b); // shows -76
```

As you can see, overflow is not desirable, so choose your data types wisely, and make sure that it can fit all the possible ranges in your application.
Underflow

Underflow is also similar to overflow, only that it goes below it's lower bounds in floating point numbers. In this case, it will be approximated as 0.

Ex.

```java
float f = 123;
f /= 1e50; // should be 1.23e-48, but the smallest value
          // float can store is 1.4e-45
System.out.println("f is " + f); // shows 0.0
```
Two's complement is the common binary representation of a negative integral number.

The leftmost bit (most significant bit or MSB) serves as a sign bit, where 1 means negative and 0 means positive. Example of 8 bit numbers:

11111111 = -1
00001100 = 12
2's complement

- To get two's complement of a positive number, simply subtract it by the number of values for that bit representation. Example on 8 bit:

\[
\begin{align*}
100000000 & \quad // \quad 2^8 = 256 \\
-00001100 & \quad // \quad -) \quad 12 \\
11110100 & \quad // \quad -12 \; (2's \; complement)
\end{align*}
\]

- Another way of getting a two's complement is to complement/negate/invert the number then add 1.

\[
\begin{align*}
00001100 & \quad // \quad +12 \\
11110011 & \quad // \quad \text{negate}(12) \\
11110100 & \quad // \quad \text{then plus 1}
\end{align*}
\]
Doing subtraction by addition

Subtraction is actually performed by adding the minuend with the 2's complement of the subtrahend:

\[
64 - 10 = 01000000 - 00001010 \\
= 01000000 + 11110110 \\
= 00110110 = 54
\]
Explanation behind subtraction by addition
Explanation behind subtraction by addition

This is an example of 64 minus 10.

256 - 10 -> formula for 2's complement!
Why use 2's complement?

- To save CPU space by reusing existing components (adder and inverter)
- To be able to express negative and positive numbers
What is stored in floating point?

- sign (positive/negative)
- significand
  
  \[-3.24 \times 10^{23}\]
- exponent bias
  
  \[-3.24 \times 10^{23}\]