Math in Code

ADDITION

How would you add to a variable? You might come up with a something like this:

```
int x = 4;

x + 3; // This doesn’t work.
```

This would be like writing a line of code that just has the number 7 on it…the computer can do the math just fine…but then it just drops it on the floor, because you didn’t tell it to do anything with it. Instead you have to tell it to put it somewhere.

```
int x = 4;

int y = x + 3;

System.out.println(y); // This prints out 7.
```

You can add together numbers and variables in any combinations you would like.

```
int x = 4;

int y = 3;

x = x + 3;

int z = x + y;

y = 5;
```

```
System.out.println(x); // This outputs 7.

System.out.println(y); // This outputs 5.

System.out.println(z); // This outputs 10.
```

Shortcuts

It is very common for a variable to need a number added to it (like x = x + 3;) so there is a shortcut for for this, which does the exact same thing, just with fewer characters f
Even more common is to have a time where you want to add 1 to a variable, so there is an even shorter form for that.

```
int x = 4;

x ++;

System.out.println(x); // This outputs 5.
```

This is the same as typing

SUBTRACTION

Everything that works for addition also works for subtraction so all of this works just fine:

```
int x = 40;

int y = 3;

x = x - 3; // x is now 37.

int z = x - y; // z is now 34

z -= 2; // z is now 32

y--; // y is now 2
```

MULTIPLICATION

For multiplication in code, we use the * symbol.

Almost everything works the same way for multiplication, except you don’t need a shortcut for multiplying by 1, so there is no ** operator.

```
int x = 4;

int y = 3;

x = x * 3; // x is now 12.

int z = x * y; // z is now 36

y *= 2; // y is now 6
```

DIVISION

For division in code, we use the / symbol.

Again, almost everything works like addition and subtraction except you don't need any shortcut to divide by 1. (Also // is already for comments).

```
int x = 8;

int y = 2;

x = x / 2; // x is now 4.

int z = x / y; // z is now 2

y /= 2; // y is now 1
```

Truncation

Be careful with division with ints in Java, since they can't have decimal places. For that reason, Java does what is called **truncation**. That means that it ignores everything after the decimal place. It is not the same as rounding.

```
int x = 3;

x /= 2; // x is now 1.
```

MODULUS

There is one more operator that we will deal with in Java and that is **modulus** (%). Modulus is the remainder in division. So if you had 7 % 3, you would look at the remainder if you did the problem 7 / 3. Since 3 goes evenly into 6, the remainder is 1, so 7 % 3 is 1. 8 % 3 is 2. 9 % 3 is 0.

Again, all the things you can do with addition you can do with modulus, but a %% doesn't exist.

```
int x = 4;

int y = 3;

x = x % 3; // x is now 1.

int z = x % y; // z is now 1

y %= 2; // y is now 1
```

TRY IT OUT!

- Read in two ints, add them together, print out the result
- Try some variations on this and see what happens:
    - Try putting in really large numbers
    - Try putting in negative numbers
    - Try changing your code to read more than two numbers
    - Try subtraction/multiplication/division/modulus
    - Try doubles/ints/Strings/booleans/some of each
    - Try doing all of the above mixed together

Note: I know that not all of the above will work as expected, but an important part of learning to code is learning to try things out and learning how to understand and fix things when they don't work. Don't be afraid to experiment. Some of these things work and some don't, try it out and figure out which is which.