Learning Objectives - Arithmetic Operators

- **Recognize the symbols for arithmetic operators**
- **Understand that strings can use the + and * operators even though math isn't involved**
- **Know how to increment or decrement a variable**
- **Understand the order of operations**

Addition

The Addition (+) Operator

The addition operator works as you would expect with numbers.

```
System.out.print(7 + 3);
```

You can also add two variables together.

```
int a = 7;
int b = 3;
System.out.print(a + b);
```

challenge

What happens if you:

- Make `a` of type double(e.g. `double a = 7.0;`)?
- Make `b` a negative number (e.g. `int b = -3;`)?
- Make `b` an explicitly positive number (e.g. `int b = +3;`)

Incrementing Variables

Incrementing Variables

Incrementing a variable means to change the value of a variable by a set amount. You will most often have a counting variable, which means you will increment by 1.

```
int a = 0;
a = a + 1;
System.out.print(a);
```

How to Read `a = a + 1`

The variable `a` appears twice on the same line of code. But each instance of `a` refers to something different.

How to Read a = a + 1

The ++ and += Operators

Incrementing is a common task for programmers. Many programming languages have developed a shorthand for `a = a + 1` because of this, `a++` does the same thing as `a = a + 1`.

```
int a = 0;
int b = 0;
a = a + 1;
b++;
System.out.println(a);
System.out.println(b);
```

In the cases you need to increment by a different number, you can specify it using the `+=` operator. You can replace `b++;` with `b+=1;` in the above code and get the same result.

challenge

What happens if you:

- Change `b` such that `b+=2`?
- Change `b` such that `b+=-1`?
- Change `b` such that `b-=1`?

String Concatenation

String Concatenation

String concatenation is the act of combining two strings together. This is done with the `+` operator.

```
String a = "This is an ";
String b = "example string";
String c = a + b;
System.out.println(c);
```

challenge

What happens if you:

- Concatenate two strings without an extra space (i.e. `a = "This is an"`)?
- Use the `+=` operator instead of the `+` operator (i.e. `a+=b;`)?
- Add `3` to a string?
- Add `"3"` to a string?

Subtraction

Subtraction

```
int a = 10;
int b = 3;
int c = a - b;
System.out.println(c);
```

challenge

What happens if you:

- Change `b` to `-3`?
- Change `c` to `c = a - -b`?
- Change `b` to `3.0`?

The `--` and `-=` Operators

Decrementing is the opposite of incrementing. Just like you can increment with `++`, you can decrement using `--`.

```
int a = 10;
a--;
System.out.println(a);
```

Like `+=`, there is a shorthand for decrementing a variable - `-=`.

Subtraction and Strings

You might be able to concatenate strings with the `+` operator, but you cannot use the `-` operator with them.

```java
String a = "one two three";
String b = "one";
String c = a - b;
System.out.println(c);
```

Division

Division

Division in Python is done with the `/` operator

```
double a = 25;
double b = 4;
System.out.println(a / b);
```

challenge

What happens if you:

- Change `b` to `0`?
- Change `b` to `0.5`?
- Change the code to

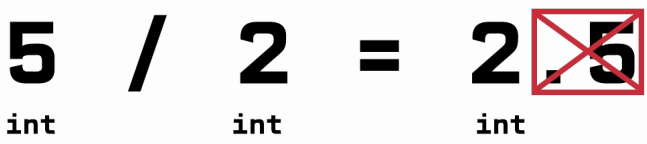
```
double a = 25;
double b = 4;
a /= b;
System.out.println(a);
```

▼ **Hint**

`/=` works similar to `+=` and `-=`

Integer Division

Normally, you use `double` in Java division since the result usually involves decimals. If you use integers, the division operator returns an `int`. This "integer division" does not round up, nor round down. It removes the decimal value from the answer.



.guides/img/intDivision

```
int a = 5;
int b = 2;
System.out.println(a / b);
```

Type Casting

Type Casting

Type casting (or type conversion) is when you change the data type of a variable.

```
int numerator = 40;
int denominator = 25;
System.out.println( numerator / denominator);
System.out.println( (double) numerator / denominator);
```

`numerator` and `denominator` are integers, but `(double)` converts `numerator` into a double.

challenge

What happens if you:

- Cast only `denominator` to a double?
- Cast both `numertor` and `denominator` to a double?
- Cast the result to a double (i.e. `(double)(numerator / denominator)`)?

▼ **More Info**

If either or both numbers in Java division are a `double`, then `double` division will occur. In the last example, numerator and denominator are both `int` when the division takes place - then the integer division result is converted to a double.

Parsing Strings

Do you know why the code below will not work?

```
int a = 5;
String b = "3";
System.out.println(a + b);
```

You cannot add a string to an integer. You can convert `b` to an integer to fix the problem.

```
int a = 5;
String b = "3";
System.out.println(a + Integer.parseInt(b));
```

Data read from the keyboard or a file is always stored as a string. If you want to use this data, you will need to know how to convert it to the proper data type.

challenge

What happens if you:

- Parse a String to a double using `Double.parseDouble()`
- Parse a String to a boolean using `Boolean.parseBoolean()`
- Convert a different type to a string with `String.valueOf()`

Modulo

Modulo

Modulo is the mathematical operation that performs division but returns the remainder. The modulo operator is `%`.

Modulo

```
int modulo = 5 % 2;
System.out.println(modulo);
```

challenge

What happens if you:

- Change `modulo` to `5 % -2`?
- Change `modulo` to `5 % 0`?
- Change `modulo` to `5 % 2.0`?

Multiplication

Multiplication

Java uses the `*` operator for multiplication.

```java
int a = 5;
int b = 10;
System.out.println(a * b);
```

challenge

What happens if you:

- Change `b` to `0.1`?
- Change `b` to `-3`?

▼ **Hint**

`*=` works similar to `+=` and `-=`

Order of Operations

Order of Operations

Java uses the PEMDAS method for determining order of operations.

P Parentheses
E Exponents - powers & square roots
MD Multiplication & Division - left to right
AS Addition & Subtraction - left to right

PEMDAS

The code below should output `10.0`.

```
int a = 2;
int b = 3;
int c = 4;
double result = 3 * a - 2 / (b + 5) + c;
System.out.println(result);
```

▼ **Explanation**

- The first step is to compute `b + 5` (which is `8`) because it is surrounded by parentheses.
- Next, do the multiplication and division going from left to right. `3 * a` is `6`.
- `2` divided by `8` is `0` (remember, the `/` operator returns an `int` when you use two `int`s so `0.25` becomes `0`).
- Next, addition and subtraction from left to right - `6 - 0` to get `6`.
- Finally, add `6` and `4` together to get `10.0`.

challenge

Mental Math

```
5 + 7 - 10 * 3 /0.5
```

▼ Solution

-48.0

```
(5 * 8) - 7 % 2 - (-1 * 18)
```

▼ Solution

57.0

```
9 / 3 + (100 % 0.5) - 3
```

▼ Solution

0.0