Operators

John Doe

February 8, 2022

Contents

- Note there is no exponent operator in vanilla java, instead use Math.pow(a, e), where a is the **base** and e is the **exponent**.

1 Operators

Basic Operations	Atithmetic
*	Multiplication
%	Modulo (Remainder)
-	Subtraction
+	Addition

1.1 Precedence

```
System.out.println(3 + 6);
System.out.println(3 + 6 - 2 * 4 + 5);
System.out.println(3 / 6);
```

1.2 Division

In java integer division results in only whole numbers being reported. Think of it kind of like rounding down to the nearest whole number. If one of the numbers is a float java will keep the answer as a float.

```
System.out.println(9 / 6);
System.out.println(9.0 / 6.0);
System.out.println(9.0 / 6);
```

1.3 Modulo

This operator will give us the remainder of a division operation. 6 goes into 9 one time with 3 leftover. When the argument on the left is smaller than the right the output is still the numerator of the mixed number. So 1 % 4 results in 0 1/4 -> 1, or 2 % 4 = 0 2/4 -> 2. The result from a modulo cannot exceed the argument on the right, but it can be any whole number below it.

```
System.out.println(9 % 6);
System.out.println(10 % 4);
System.out.println(8 % 4);
System.out.println(0 % 4);
System.out.println(1 % 4);
System.out.println(3 % 4);
```

2 Increment/Decrement

Both the ++ and the – operators take exactly one argunemt, **Urnary**. The **post-fix** operator is the same thing as saying (arg = arg + 1). The post fix operator will pull the value, and the increment it, thus the increment is **post** assignement. The **pre-fix** operator is like (arg = arg + 1) but the arg is incremented before assignment.

```
int i = 10;
int j = 20;
System.out.println(i++);
System.out.println(++j);
System.out.println(i);
```

```
int i = 10;
int j = 20;
i = j++;
System.out.println("i is set to j before incrementing");
System.out.println("i=" + i);
System.out.println("j=" + j);
i = ++j;
System.out.println("i is set to j after incrementing");
System.out.println("i=" + i);
System.out.println("j=" + j);
```