CMSC131

Additional readings on expressions and operations.

Expressions

We have seen several examples of expressions in Java.

– Some of these returned numbers, other strings.

The result of some expressions were assigned to variables, others were passed to methods (such as when printing) and others were themselves used as part of larger expressions.

"x=1" is actually an expression. After it assigns 1 to x, it returns the value 1 as well.

"Side Effects"

Consider the following code…

```
public static void main(String[] args) {
   int x,y;
   x=y=1;
   System.out.println(x+" "+y);
}
```

It will compile and print **1 1** to the screen.

Increment and Decrement

There are a few more math operators we've seen in Java.

We need to be careful with how we use these.

```
      x++;  //post increment
      ++x;  //pre increment
      x+=val; //"add" that val to x
```

There are also others like

```
     x--; --x; x-=val; x*=val; x/=val;
```

What do you think the output is?

```
int x,y;

x=2; y=5;
System.out.println(x++ * y++);

x=2; y=5;
System.out.println(++x * ++y);

x=2; y=5;
System.out.println(++x * y++);

x=2; y=5;
System.out.println(x++ * ++y);
```

Precedence / Order of Operations

In Java they are (top being higher precedence)

- parentheses
- unary operations like
  `-x, !x, ++x, --x, x++, x--`
- multiplication and division and modulus
- addition and subtraction
- inequality comparisons (greater than, less than, etc)
- equality comparisons (equal to, not equal to)
- logical and
- logical or
- assignment operations like
  `=, +=, *=, /=, %=`

In the case of a tie…

If two operators have the same precedence, then they are generally evaluated from left to right on the line.

HOWEVER, assignments are actually done from right to left!

```
x = y = z = 4;
```

Order of Operations

What do you think **`x`** will end up holding in

```
int x=8/4*2/2;
```

Short-circuiting

We briefly discussed in lab how once the left-hand operand of an "and" is false, there's no logical need to consider the right-hand operand and once the left-hand operand of an "or" is true, there's no logical need to consider the right-hand operand.

So, what is the output of the following?

```
int x=1, y=1, z;
if (x++ > 5 && y-- < 5) {
     z = 10;
} else {
     z = 20;
}
System.out.println(x + "   " + y + "   " + z);
```