Shortcut Operators

- Something we often need to do in Java is to modify the current value of a variable. To change a variable’s value, we first need to determine its current value on the right side of the assignment and then do the computation to change how we want.

*numPizzasSold = numPizzasSold + 1;*

- Then we finally store the new value back into the variable.

- The most common shortcuts in Java are increment (++) and decrement (--) operators. There are actually two versions of these shortcuts. There is a pre-increment/decrement, where the operator is written before the variable name and is done before anything else in the statement.

*++numPizzasSold;*

*--numPizzasSold;*

- Then we have the post-increment/decrement where the operator is written after the variable name and is done after anything else in the statement.

*numPizzasSold++;*

*numPizzasSold--;*

- Arithmetic assignment operators do both arithmetic and assignment.

+ =

- =

\* =

/ =

% =

Suppose you have an int variable named num1 with the value 10. What will the value of num1 be after the following statement? num1 \*= 2;

A. 20

B. 10

C. 100

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

package SpecOps.src;

public class SpecOps {

public static void main(String[] args) {

int num1 = 2, num2 = 3;

num1 += 2;

//num1 should now equal 4

num2++;

// num2 should now equal 4

num1++;

// num 1 should now equal 5

num1 = ++num2 + 2;

/\*

num1 did equal 5.

now num1 equals 7

and num2 equals 5 because the pre-increment added one to it's original

value of 4 prior to the calculation.

\*/

System.out.println("num1: " + num1);

System.out.println("num2: " + num2);

/\*

Now we'll set num1 back to 5.

And num2 back to 4.

So we can observe how pre and post increment operators must intentionally

be placed.

There is a certain sequence that will compute different values.

A pre-increment++ will compute +1 before the rest of the statement.

A post-increment++ will compute +1 after the rest of the statement.

\*/

num1 = 5;

num2 = 4;

num1 = num2++ + 2;

// num2 should equal 4 now because of the post-decrement

System.out.println("num1: " + num1);

System.out.println("num2: " + num2);

/\*

num1 result = 6

num2 result equals 5 because it's post-increment added one to

its value after it calculated num1 + num2.

If we look at the results we get a different calculation depending

on if we use a pre or a post-increment.

num1: 7

num2: 5

num1: 6

num2: 5

\*/

}

}

- The pre/post-increment operators will always *assign* +1 to the variable we attach it too (num1++). So if we print the value of a pre or post-decrement, it will be the same. But if we use a pre-decrement, the value of the calculation is 1 greater than if we used a post-decrement because the value of the variable will not yet have added +1.

- These operators should probably not be used within a statement unless it’s necessary to do so.

package SpecOps.src;

public class SpecOps {

public static void main(String[] args) {

int num1 = 2, num2 = 3, num3 = 5;

num3 \*= num2 + num1;

/\*

The addition of num1 + num2 will happen first, before the multiplication

assignment operator.

In this case, the multiplication is part of the assignment, so it will

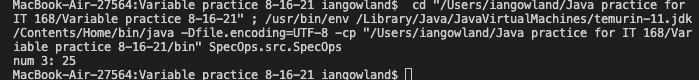
not be multiplied until everything on the right side is calculated.

The value should be 5 \* (2+3) = 25

\*/

System.out.println("num 3: " + num3);

}



YES!

Which of the following statements has a different meaning than the others?

A. ++num2;

B. num2 += 1;

C. num2 = 1;

D. num2++;

E. num2 = num2 + 1;

Text

Description automatically generated