Increment and Decrement Operators

Introduction to Programming

Increment and Decrement Operators

- These are special operators that *increment* and *decrement* a variable by one
- Can only be applied to variables
- Increment operator `++`

  `a = a + 1;` *becomes* `a++;`
- Decrement operator `--`

  `count = count – 1;` *becomes* `count--;`

Increment and Decrement Operators

- These are *Unary* operators – take one operand
- The operand is ***always*** a variable
- Eg

```
value++
--myNum
++a
price--
```

Pre-increment and *Post*-increment

- Increment and decrement operators can be used in both *prefix* and *postfix* form
- Increment and decrement operators can *follow* or *precede* the variable

```
variable++;        or    ++variable;
```

- The effect on the variable is the *same*
- The *position* of the operator affects when the operation is performed

Post-fix and pre-fix

- The expressions `++i` and `i++` have a value
- Each causes the stored value of `i` in memory to be incremented by 1
- The expression `++i` causes the stored value of `i` to be incremented first
- The expression `i++` has as its value the current value of `i`; then the expression causes the stored value of `i` to be incremented (`i` incremented last)

Sample Program

```
public static void main(String[] args)
{
     int i, j;
     i = 10;
     j = i++;

     System.out.println("value of i is " +i);
     System.out.println("value of j is " +j);
}

   // j = i++   the current value of i is assigned to j
   // i is then incremented
   // Hence, j has the value 10 not 11
```

Post –fix and *Pre* - fix

Pre-fix

- If the ++ or −− appears *before* a variable the variable is incremented/decremented *before* it is used

Post-fix

- If the ++ or −− appears *after* a variable the variable is incremented/decremented *after* it is used

- This is only relevant when the result of the operation is assigned to a variable.

`age++ ;`	*same as*	`++age;`
`newAge = age++;`	*not the same as*	`newAge = ++age;`

Examples

Consider

```
max = 2;
count = 10 * max++;
```

OR

```
max = 2;
count = 10 * ++max;
```

Pre increment

```
public static void main(String[] args)
{
     int i, j;

     i=10;
     j= ++i ;
     System.out.println("value of i is " +i);
     System.out.println("value of j is " +j);
}
```

What is the output?

```
   public static void main(String[] args)
   {
         int a, b, c = 0;
         a = ++c;
         b = c++;

         System.out.println(a + " "  + b + " " + c);

   }
```

What is the output?

```
public class TestIncrement
{
    public static void main(String [] args)
    {
        int a, b, c = 0;
        a = ++c;
        b = c++;

        System.out.println(a + " "  + b + " " + ++c);

}
```

- Increment ++ and decrement -- operators cause the value of a variable in memory to be changed.
- Other operators do not do this
- Eg
  `a + b` leaves values of variables `a` and `b` unchanged
- The operators ++ and -- are said to have a *side-effect*, not only do they yield a value, they also change the stored value of a variable in memory

Example

```
int a = 0, b = 0, c = 0;

a = ++b + ++c;
System.out.println(a +" "  +b + " " +c);

a = b++ + c++;
System.out.println(a +" "  +b + " " +c);

a = ++b + c++;
System.out.println(a +" "  +b + " " +c);

a = b-- + --c;
System.out.println(a +" "  +b + " " +c);
```

Example Solution

```
int a = 0, b = 0, c = 0;

a = ++b + ++c;
System.out.println(a +" "  +b + " " +c); // 2 1 1

a = b++ + c++;
System.out.println(a +" "  +b + " " +c); // 2 2 2

a = ++b + c++;
System.out.println(a +" "  +b + " " +c); // 5 3 3

a = b-- + --c;
System.out.println(a +" "  +b + " " +c); // 5 2 2
```