The Decrement/Increment operators

The decrement and increment operators ++ and -- have the side effect of changing a variable. Consider int variable k. Evaluation of expression:

++k	increments k and has as its value the new value of k.
k++	increments k and has as its value the value of k *before* the increment.
--k	decrements k and has as its value the new value of k.
k--	decrements k and has as its value the value of k *before* the increment.

The type of k can be any number type (byte, short, int, long, char, float, double) or its wrapper class (Byte, Short, Integer, Long, Character, Float, Double). Consider this code, remembering that type byte has values -128..127:

```
Byte b1= 127; System.out.println(++b1);      // Change b1 to a new object that contains -128 and print -128
```

To evaluate ++b1, Byte variable b1 is unboxed, giving the value 127; 1 is added, giving 128; 128 is cast to type byte, giving -128; -128 is boxed and a pointer to the new object of class Byte is stored in b1; -128 is returned.

Thus, unboxing, casting, and boxing will occur as necessary.

For are two more examples; the one involving char c is interesting:

```
byte b2= 127; System.out.println(++b2);      // Change b2 to a new object that contains -128 and print -128
char c= 'a'; System.out.println(++c);        // Change c to a new object that contains 'b' and print 'b'
```

Uses of increment/decrement operations in loops

Increment and decrement operators are often used as shown in the following two loops. The first loop prints the values of array b in reverse order. The second prints the lowercase characters 'a'.. 'z'. In these loops, the values of the expressions i-- and c++ are not used but are simply discarded.

```
for (int i= b.length-1; i >= 0; i--) System.out.println(b[i]);

for (char c= 'a'; c <= 'z'; c++)  System.out.println(c);
```

Stay away from real side effects

Some people use statements like this:

```
b[i++]= 5;
```

The main job of this statement is to store 5 in b[i], but at the same time, it increments i. The assignment to b[i] has the *side effect* of incrementing i.

We rarely use such side effects, for it can be confusing. The theory back this up. Any system for proving program correctness that allows side effects is far more complicated than one that doesn't allow side effects.