The example treated in this document illustrates that care must be taken in calculating the number of basic steps.

In the box to the right, with x a variable of type **int**, we know that the assignment to x is a basic step. The assignment to String variable s sure looks similar to the assignment to x, and the first thought is that it must also be a basic step. But it is not. In fact, we will show that the number of basic steps is proportional to the length of s.

x= x + 1;

s= s + '$';

First, we must understand how strings are implemented. Variable s contains a pointer to a String object. This object contains, among other things, a pointer to an object that is an array of chars, which contains the characters in the string. We call this the *backing array* for the string. In the example shown to the right, the array contains three chars: 'd', 'e', and '4'.

String@00

String

?

char[]

s

char[]@02

char[]

0

1

2

'd'

'e'

'4'

We now show how the catenation s + '$' is evaluated in three steps:

(1) Create a second String object and a second char[] object, the latter with space for 1 more character, and create a pointer v to the new String object;

String@01

String

?

char[]

v

char[]@04

char[]

0

1

2

3

'd'

'e'

'4'

'$'

(2) Copy the 3 chars 'd', 'e', and '4' from object char[]@02 to object char[]@04; and

(3) Place the catenated character '$' into the array object char[]@04, producing the new objects shown to the right.

The assignment s= s + '$'; is then completed by assigning v to s, so s finally points to string object String@01.

**Figuring out the basic steps in evaluating s= s + '$';**

The first step in evaluating s + '$' is to create the new String object *and* the char[] object to which it points. We can consider this to be *one* basic step. Of course, it takes a lot of time, perhaps 1000 times more than just evaluating x+y, but the time *is* independent of all values, including the char array in char[]@0. Remember that the compiler figures out where each variable and method goes in the String and char[] objects, so space allocation costs just contant time when the objects are being created. So we consider it to be *one* big basic step.

The second step is to copy the characters in the original char array (in object char[]@02) to the new char array (in object char[]@04). This takes, s.length() basic steps, because s.length() chars have to be copied into the new array.

Then, the catenated character '$', has to be placed in the new array. This is one basic step.

Finally, the assignment s= v; has to be executed. This is one basic step.

Therefore the number of basic steps is s.length() + 3.

Therefore, the number of basic steps taken in executing s= s + '$'; is proportional to the number of characters in string s.