In the assignment

a = 123;

we are using, to the right of the equals sign, the decimal representation of an integer. As you are well aware, the adjective "decimal" carries the information that the integer is represented in base 10. To determine which integer has this decimal name, we make use of our understanding of the standard positional system in which each digit position has a value 10 times that of the neighboring position to the right and the rightmost position has the value 1 (that is, 100). So we interpret 123 as

(1 × 102) + (2 × 101) + (3 × 100),

that is, one hundred plus twenty plus three, or one hundred twenty-three.

Integers may be represented in other bases too. In general, if b is an integer greater than 1, then to represent integers in base b we need a collection of digits, one for each of the numbers from 0 through b−1. (For base 10, we use the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.) And we use a positional system in which each digit position has a value b times that of the neighboring position to the right and the rightmost position has the value 1 (that is, b0). Thus, if w, x, y, z are base b digits, then wxyz represents the number

(w × b3) + (x × b2) + (y × b1) + (z × b0).

When used in a Java expression, a sequence of digits such as we have been discussing is called a numerical literal. (We introduce other kinds of numerical literals in a little while.) Java recognizes three different number bases, namely, 8, 10, and 16. Base 8 number representations are said to be octal, base 10 representations are decimal, and base 16 representations are hexadecimal. To let Java know when we are using a non-decimal base, we abide by the following conventions:

* An octal literal starts with 0 (that's a zero, not a letter O) and uses the octal digits 0, 1, 2, 3, 4, 5, 6, 7.
* A hexadecimal literal starts with 0x (again, that's a zero … and a lowercase letter x) and uses the hexadecimal digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f (where the last six digits stand for ten, eleven, twelve, thirteen, fourteen, and fifteen, respectively).

Experiment with the three standard Java numerical bases by entering the following numerical literals one at a time into the text box below and pressing the **Run** button each time. Take the time to make sure that you understand why each report says what it does. (The report is always expressed in base 10.)

A very simple number base that is used behind the scenes in just about every computer application is base 2. Base 2 number representations are said to be *binary*; they use only the digits 0 and 1. Java does not provide a direct way to write numerical literals in base 2. So we have to fall back on descriptive phrases. For example, "the binary representation of the integer 123 is 1111011". This is because

123 = (1 × 26) + (1 × 25) + (1 × 24) + (1 × 23)   
                + (0 × 22) + (1 × 21) + (1 × 20)

 A simple technique for converting binary to octal literal is to group the binary digits in threes starting from the right and to replace each group of three by its decimal equivalent.

A simple technique for converting binary to hexadecimal literal is to to group the binary digits in fours starting from the right and to replace each group of four by its decimal equivalent.