To develop a better understanding of the rules of operator precedence, consider the evaluation of a second-degree polynomial ($y = ax^2 + bx + c$):

```
y = a * x * x + b * x + c;
```

6 1 2 4 3 5

The circled numbers indicate the order in which Java applies the operators. The multiplication operations are evaluated first in left-to-right order (i.e., they associate from left to right), because they have higher precedence than addition. The addition operations are evaluated next and are applied from left to right. There is no arithmetic operator for exponentiation in Java, so x^2 is represented as `x * x`. Section 5.4 shows an alternative for performing exponentiation in Java.

Suppose that `a`, `b`, `c` and `x` in the preceding second-degree polynomial are initialized (given values) as follows: `a = 2`, `b = 3`, `c = 7` and `x = 5`. Figure 2.13 illustrates the order in which the operators are applied.

As in algebra, it is acceptable to place unnecessary parentheses in an expression to make the expression clearer. These are called **redundant parentheses**. For example, the preceding assignment statement might be parenthesized as follows:

```
y = ( a * x * x ) + ( b * x ) + c;
```

Good Programming Practice 2.14

Using parentheses for complex arithmetic expressions, even when the parentheses are not necessary, can make the arithmetic expressions easier to read.

```
Step 1.  y = 2 * 5 * 5 + 3 * 5 + 7;      (Leftmost multiplication)
         2 * 5 is 10

Step 2.  y = 10 * 5 + 3 * 5 + 7;         (Leftmost multiplication)
         10 * 5 is 50

Step 3.  y = 50 + 3 * 5 + 7;             (Multiplication before addition)
         3 * 5 is 15

Step 4.  y = 50 + 15 + 7;                (Leftmost addition)
         50 + 15 is 65

Step 5.  y = 65 + 7;                     (Last addition)
         65 + 7 is 72

Step 6.  y = 72                          (Last operation—place 72 in y)
```

Fig. 2.13 | Order in which a second-degree polynomial is evaluated.