By adding this function to the class declaration, you can let a program investigate a series of stocks to find the one with the greatest value. However, let's take a different approach, mainly so you can learn about the this pointer. The approach is to define a member function that looks at two Stock objects and returns a reference to the larger of the two. Attempting to implement this approach raises some interesting questions, and we look into them now.

First, how do you provide the member function with two objects to compare? Suppose, for example, you decide to name the method topval(). Then, the function call stock1.topval() accesses the data of the stock1 object, whereas the message stock2.topval() accesses the data of the stock2 object. If you want the method to compare two objects, you have to pass the second object as an argument. For efficiency, pass the argument by reference. That is, have the topval() method use a type const Stock & argument.

Second, how do you communicate the method's answer back to the calling program? The most direct way is to have the method return a reference to the object that has the larger total value. Thus, the comparison method should have the following prototype:

const Stock & topval(const Stock & s) const;

This function accesses one object implicitly and one object explicitly, and it returns a reference to one of those two objects. The const in the parentheses states that the function won't modify the explicitly accessed object, and the const that follows the parentheses states that the function won't modify the implicitly accessed object. Because the function returns a reference to one of the two const objects, the return type also has to be a const reference.

Suppose, then, that you want to compare Stock objects stock1 and stock2 and assign the one with the greater total value to the object top. You can use either of the following statements:

top = stock1.topval(stock2);
top = stock2.topval(stock1);

The first form accesses stock1 implicitly and stock2 explicitly, whereas the second accesses stock1 explicitly and stock2 implicitly. (See Figure 10.3.) Either way, the method compares the two objects and returns a reference to the one with the higher total value.