create two Stock objects called sally and solly. The sally object, for example, could represent Sally's stock holdings in a particular company.

Next, notice that the information we decided to store appears in the form of class data members, such as company and shares. The company member of sally, for example, holds the name of the company, the share member holds the number of shares Sally owns, the share_val member holds the value of each share, and the total_val member holds the total value of all the shares. Similarly, the operations we want appear as class function members, such as sell() and update(). Class member functions also are termed class methods. A member function can be defined in place, like set_tot(), or it can be represented by a prototype, like the other member functions in this class. The full definitions for the other member functions come later, but the prototypes suffice to describe the function interfaces. The binding of data and methods into a single unit is the most striking feature of the class. Because of this design, creating a Stock object automatically establishes the rules governing how that object can be used.

You've already seen how the istream and ostream classes have member functions, such as get() and getline(). The function prototypes in the Stock class declaration demonstrate how member functions are established. The iostream header file, for example, has a getline() prototype in the istream class declaration.

Also new are the keywords private and public. These labels describe access control for class members. Any program that uses an object of a particular class can access the public portions directly. A program can access the private members of an object only by using the public member functions (or, as you see in Chapter 11, "Working with Classes," via a friend function). For example, the only way to alter the shares member of the Stock class is to use one of the Stock member functions. Thus, the public member functions act as go-betweens between a program and an object's private members; they provide the interface between object and program. This insulation of data from direct access by a program is called data hiding. (C++ provides a third access-control keyword, protected, which we discuss when we cover class inheritance in Chapter 13, "Class Inheritance.") (See Figure 10.1.) Whereas data hiding may be an unscrupulous act in, say, a stock fund prospectus, it's a good practice in computing because it preserves the integrity of the data.