```
1  // Fig. 16.11: GradeBook.h
2  // GradeBook class definition. This file presents GradeBook's public
3  // interface without revealing the implementations of GradeBook's member
4  // functions, which are defined in GradeBook.cpp.
5  #include <string> // class GradeBook uses C++ standard string class
6  using namespace std;
7
8  // GradeBook class definition
9  class GradeBook
10 {
11 public:
12    GradeBook( string ); // constructor that initializes courseName
13    void setCourseName( string ); // function that sets the course name
14    string getCourseName(); // function that gets the course name
15    void displayMessage(); // function that displays a welcome message
16 private:
17    string courseName; // course name for this GradeBook
18 }; // end class GradeBook
```

Fig. 16.11 | GradeBook class definition containing function prototypes that specify the interface of the class.

function `displayMessage`'s function prototype (line 15) specifies that `displayMessage` does not require parameters and does not return a value. These function prototypes are the same as the corresponding function headers in Fig. 16.9, except that the parameter names (which are *optional* in prototypes) are not included and each function prototype *must* end with a semicolon.

Good Programming Practice 16.2

Although parameter names in function prototypes are optional (they're ignored by the compiler), many programmers use these names for documentation purposes.

Error-Prevention Tip 16.4

Parameter names in a function prototype (which, again, are ignored by the compiler) can be misleading if the names used do not match those used in the function definition. For this reason, many programmers create function prototypes by copying the first line of the corresponding function definitions (when the source code for the functions is available), then appending a semicolon to the end of each prototype.

GradeBook.cpp: Defining Member Functions in a Separate Source-Code File

Source-code file `GradeBook.cpp` (Fig. 16.12) *defines* class `GradeBook`'s member functions, which were *declared* in lines 12–15 of Fig. 16.11. The definitions appear in lines 9–32 and are nearly identical to the member-function definitions in lines 12–35 of Fig. 16.9.

Each member-function name in the function headers (lines 9, 15, 21 and 27) is preceded by the class name and `::`, which is known as the **binary scope resolution operator**. This "ties" each member function to the (now separate) `GradeBook` class definition (Fig. 16.11), which declares the class's member functions and data members. Without "`GradeBook::`" preceding each function name, these functions would *not* be recognized by the compiler as member functions of class `GradeBook`—the compiler would consider them


```
1  // Fig. 16.12: GradeBook.cpp
2  // GradeBook member-function definitions. This file contains
3  // implementations of the member functions prototyped in GradeBook.h.
4  #include <iostream>
5  #include "GradeBook.h" // include definition of class GradeBook
6  using namespace std;
7
8  // constructor initializes courseName with string supplied as argument
9  GradeBook::GradeBook( string name )
10 {
11    setCourseName( name ); // call set function to initialize courseName
12 } // end GradeBook constructor
13
14 // function to set the course name
15 void GradeBook::setCourseName( string name )
16 {
17    courseName = name; // store the course name in the object
18 } // end function setCourseName
19
20 // function to get the course name
21 string GradeBook::getCourseName()
22 {
23    return courseName; // return object's courseName
24 } // end function getCourseName
25
26 // display a welcome message to the GradeBook user
27 void GradeBook::displayMessage()
28 {
29    // call getCourseName to get the courseName
30    cout << "Welcome to the grade book for\n" << getCourseName()
31       << "!" << endl;
32 } // end function displayMessage
```

Fig. 16.12 | GradeBook member-function definitions represent the implementation of class GradeBook.

"free" or "loose" functions, like `main`. These are also called *global functions*. Such functions cannot access `GradeBook`'s `private` data or call the class's member functions, without specifying an object. So, the compiler would *not* be able to compile these functions. For example, lines 17 and 23 that access variable `courseName` would cause compilation errors because `courseName` is not declared as a local variable in each function—the compiler would not know that `courseName` is already declared as a data member of class `GradeBook`.

Common Programming Error 16.3

When defining a class's member functions outside that class, omitting the class name and binary scope resolution operator (::) preceding the function names causes errors.

To indicate that the member functions in `GradeBook.cpp` are part of class `GradeBook`, we must first include the `GradeBook.h` header (line 5 of Fig. 16.12). This allows us to access the class name `GradeBook` in the `GradeBook.cpp` file. When compiling `GradeBook.cpp`, the compiler uses the information in `GradeBook.h` to ensure that