```
1  // Fig. 3.5: fig03_05.cpp
2  // Define class GradeBook that contains a courseName data member
3  // and member functions to set and get its value;
4  // Create and manipulate a GradeBook object with these functions.
5  #include <iostream>
6  #include <string> // program uses C++ standard string class
7  using namespace std;
8
9  // GradeBook class definition
10 class GradeBook
11 {
12 public:
13    // function that sets the course name
14    void setCourseName( string name )
15    {
16       courseName = name; // store the course name in the object
17    } // end function setCourseName
18
19    // function that gets the course name
20    string getCourseName() const
21    {
22       return courseName; // return the object's courseName
23    } // end function getCourseName
```

Fig. 3.5 | Defining and testing class GradeBook with a data member and *set* and *get* member functions. (Part 1 of 3.)

```
24
25      // function that displays a welcome message
26      void displayMessage() const
27      {
28         // this statement calls getCourseName to get the
29         // name of the course this GradeBook represents
30         cout << "Welcome to the grade book for\n" << getCourseName() << "!"
31            << endl;
32      } // end function displayMessage
33   private:
34      string courseName; // course name for this GradeBook
35   }; // end class GradeBook
36
37   // function main begins program execution
38   int main()
39   {
40      string nameOfCourse; // string of characters to store the course name
41      GradeBook myGradeBook; // create a GradeBook object named myGradeBook
42
43      // display initial value of courseName
44      cout << "Initial course name is: " << myGradeBook.getCourseName()
45         << endl;
```

Fig. 3.5 | Defining and testing class GradeBook with a data member and *set* and *get* member functions. (Part 2 of 3.)

```
46
47      // prompt for, input and set course name
48      cout << "\nPlease enter the course name:" << endl;
49      getline( cin, nameOfCourse ); // read a course name with blanks
50      myGradeBook.setCourseName( nameOfCourse ); // set the course name
51
52      cout << endl; // outputs a blank line
53      myGradeBook.displayMessage(); // display message with new course name
54   } // end main
```

```
Initial course name is:

Please enter the course name:
CS101 Introduction to C++ Programming

Welcome to the grade book for
CS101 Introduction to C++ Programming!
```

Fig. 3.5 | Defining and testing class GradeBook with a data member and *set* and *get* member functions. (Part 3 of 3.)

3.4 Data Members, *set* Member Functions and *get* Member Functions (Cont.)

- Most data-member declarations appear after the access-specifier label `private:`
- Like `public`, keyword `private` is an access specifier.
- Variables or functions declared after access specifier `private` (and before the next access specifier) are accessible only to member functions of the class for which they're declared.
- The default access for class members is `private` so all members after the class header and before the first access specifier are `private`.
- The access specifiers `public` and `private` may be repeated, but this is unnecessary and can be confusing.

Error-Prevention Tip 3.1

Making the data members of a class `private` and the member functions of the class `public` facilitates debugging because problems with data manipulations are localized to either the class's member functions or the friends of the class.

Common Programming Error 3.2

An attempt by a function, which is not a member of a particular class (or a friend of that class) to access a `private` member of that class is a compilation error.

3.4 Data Members, *set* Member Functions and *get* Member Functions (Cont.)

- Declaring data members with access specifier `private` is known as data hiding.
- When a program creates (instantiates) an object, its data members are encapsulated (hidden) in the object and can be accessed only by member functions of the object's class.

3.4 Data Members, *set* Member Functions and *get* Member Functions (Cont.)

- In this example, `setCourseName` does not attempt to validate the course name—i.e., the function does not check that the course name adheres to any particular format or follows any other rules regarding what a "valid" course name looks like.
  - Suppose, for instance, that a university can print student transcripts containing course names of only 25 characters or fewer.
  - In this case, we might want class `GradeBook` to ensure that its data member `courseName` never contains more than 25 characters.
  - We discuss basic validation techniques in Section 3.9.
- When a function that specifies a return type other than `void` is called and completes its task, the function uses a `return` statement to return a result to its calling function.

3.4 Data Members, *set* Member Functions and *get* Member Functions (Cont.)

- Member function `displayMessage` (lines 26–32) does not return any data when it completes its task, so its return type is `void`.
- The function does not receive parameters, so its parameter list is empty.
- Line 30 calls member function `getCourseName` to obtain the value of `courseName`.
  - Member function `displayMessage` could also access data member `courseName` directly, just as member functions `setCourseName` and `getCourseName` do.
- By default, the initial value of a `string` is the so-called empty string, i.e., a string that does not contain any characters.
- Nothing appears on the screen when an empty string is displayed.

3.4 Data Members, *set* Member Functions and *get* Member Functions (Cont.)

- A client of an object—that is, any class or function that calls the object's member functions from *outside* the object—calls the class's `public` member functions to request the class's services for particular objects of the class.
  - This is why the statements in `main` call member functions `setCourseName`, `getCourseName` and `displayMessage` on a `GradeBook` object.
- Classes often provide `public` member functions to allow clients of the class to *set* (i.e., assign values to) or *get* (i.e., obtain the values of) `private` data members.
  - These member function names need not begin with `set` or `get`, but this naming convention is common.
- Set functions are also sometimes called mutators (because they mutate, or change, values), and get functions are also sometimes called accessors (because they access values).

Good Programming Practice 3.1

Always try to localize the effects of changes to a class's data members by accessing and manipulating the data members through their corresponding `get` and `set` functions.

Software Engineering Observation 3.1

Write programs that are clear and easy to maintain. Change is the rule rather than the exception. You should anticipate that your code will be modified, and possibly often.

3.4 Data Members, *set* Member Functions and *get* Member Functions (Cont.)

- Figure 3.6 contains an updated UML class diagram for the version of class `GradeBook` in Fig. 3.5.
- The UML represents data members as attributes by listing the attribute name, followed by a colon and the attribute type.

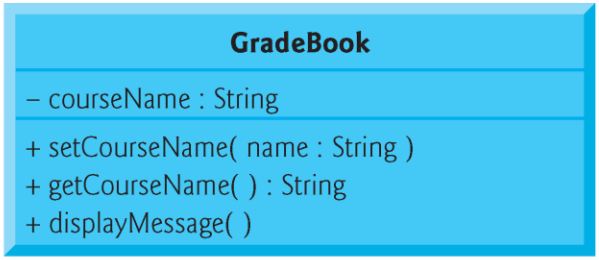

Fig. 3.6 | UML class diagram for class `GradeBook` with a private `courseName` attribute and public operations `setCourseName`, `getCourseName` and `displayMessage`.

3.5 Initializing Objects with Constructors

- Each class can provide one or more constructors that can be used to initialize an object of the class when the object is created.
- A constructor is a special member function that must be defined with the *same name as the class*, so that the compiler can distinguish it from the class's other member functions.
- An important difference between constructors and other functions is that constructors cannot return values, so they *cannot* specify a return type (not even `void`).
- Normally, constructors are declared `public`.