



## Data Members, set Functions and get Functions

---

- Variables declared in a function definition's body are known as local variables and can be used only from the line of their declaration in the function to closing right brace ( } ) of the block in which they're declared.
- A local variable must be declared before it can be used in a function.
- A local variable cannot be accessed outside the function in which it's declared.
- When a function terminates, the values of its local variables are lost. (You'll see an exception to this when we discuss static local variables.)
- A class normally consists of one or more member functions that manipulate the attributes that belong to a particular object of the class.
- Attributes are represented as variables in a class definition.
- Such variables are called data members and are declared inside a class definition but outside the bodies of the class's member-function definitions.
- Each object of a class maintains its own copy of its attributes in memory. These attributes exist throughout the life of the object.

### GradeBook Class with a Data Member, a set Function and a get Function

In the following example, class GradeBook maintains the course name as a data member so that it can be used or modified at any time during a program's execution.

- The class contains member functions *setCourseName*, *getCourseName* and *displayMessage*.
- Member function *setCourseName* stores a course name in a GradeBook data member.
- Member function *getCourseName* obtains the course name from that data member.
- Member function *displayMessage* — which now specifies no parameters—still displays a welcome message that includes the course name.

However, as you'll see, the function now obtains the course name by calling another function in the same class — *getCourseName*.

```

1 //
2 // main.cpp
3 // GradeBook
4 //
5 // Define class GradeBook with a member function displayMessage,
6 // create a GradeBook object, and call its displayMessage function.
7 //
8
9 #include <iostream>
10 #include <string> // program uses C++ standard string class
11 using namespace std;
12
13 // GradeBook class definition
14 class GradeBook
15 {
16 public:
17     // function that sets the course name
18     void setCourseName( string name )
19     {
20         courseName = name; // store the course name in the object
21     } // end function setCourseName
22
23     // function that gets the course name
24     string getCourseName ()
25     {
26         return courseName; // return the object's courseName
27     } // end function getCourseName
28
29     // function that displays a welcome message
30     void displayMessage()
31     {
32         // this statement calls getCourseName to get the name of the course this GradeBook represents
33         cout << "Welcome to the grade book for\n" << getCourseName() << "!" << endl;
34     } // end function displayMessage
35
36 private:
37     string courseName; // course name for this GradeBook
38 }; // end class GradeBook
39
40 // function main begins program execution
41 int main()
42 {
43     string nameOfCourse; // string of characters to store the course name
44     GradeBook myGradeBook; // create a GradeBook object named myGradeBook
45
46     // prompt for and input course name
47     cout << "Please enter the course name:" << endl;
48     getline(cin, nameOfCourse); // read a course name with blanks
49     myGradeBook.setCourseName( nameOfCourse ); // set the course name
50
51     cout << endl; // output a blank line
52
53     myGradeBook.displayMessage(); // display message with new course name
54 } // end main

```

Every instance (i.e., object) of class *GradeBook* contains one copy of each of the class's data members—if there are two *GradeBook* objects, each has its own copy of *courseName* (one per object).

A benefit of making *courseName* a data member is that all the member functions of the class can manipulate any data members that appear in the class definition (in this case, *courseName*).

### Access Specifiers **public** and **private**

- Most data-member declarations appear after the private access specifier. Variables or functions declared after access specifier **private** are accessible only to member functions of the class for which they're declared.

- Thus, data member `courseName` can be used only in member functions `setCourseName`, `getCourseName` and `displayMessage` of class `GradeBook`

### GradeBook's UML Class Diagram with a Data Member and set and get Functions

- This diagram models `GradeBook`'s data member `courseName` as an attribute in the middle compartment.
- The UML represents data members as attributes by listing the attribute name, followed by a colon and the attribute type.
- The UML type of attribute `courseName` is `String`, which corresponds to `string` in C++.
- Data member `courseName` is private in C++, so the class diagram lists a minus sign (-) in front of the corresponding attribute's name.
  - The minus sign in the UML is equivalent to the private access specifier in C++.
- Class `GradeBook` contains three public member functions, so the class diagram lists three operations in the third compartment.
- Operation `setCourseName` has a `String` parameter called `name`.
- The UML indicates the return type of an operation by placing a colon and the return type after the parentheses following the operation name.
- Member function `getCourseName` of class `GradeBook` has a `string` return type in C++, so the class diagram shows a `String` return type in the UML.

