Ekaba Bisong
Programming in C++
University of Calabar

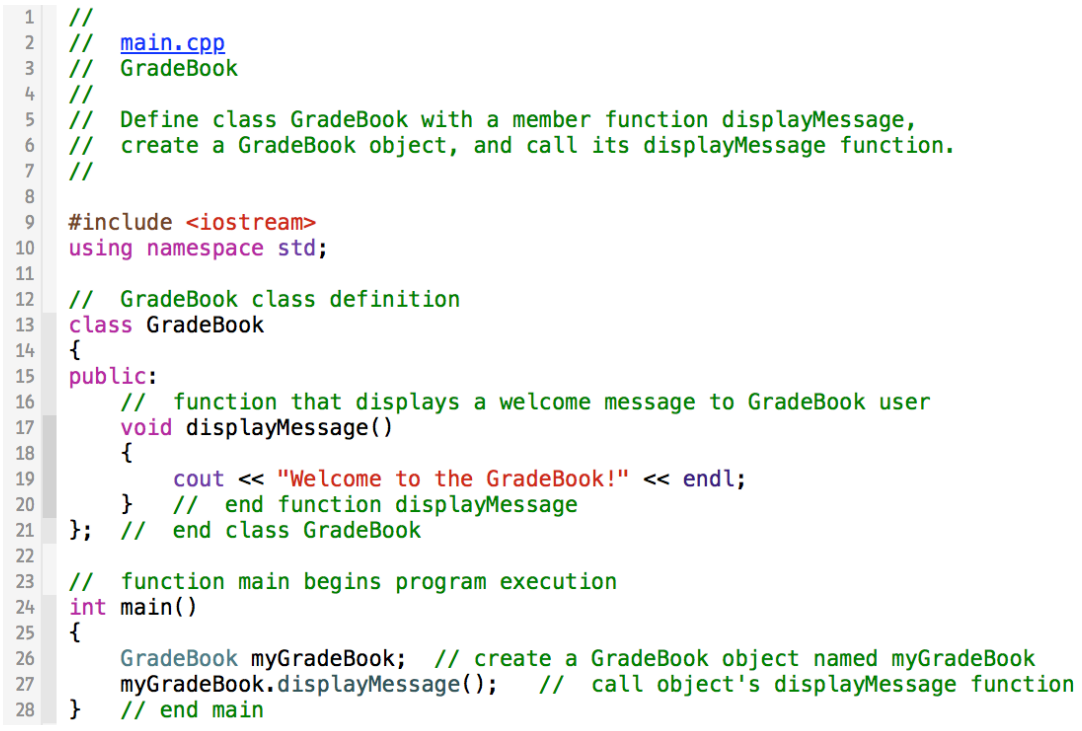
Lesson Note #11
May 11, 2015

Defining a Class with a Member Function

In previous notes, simple programs were created to display messages to the user, obtain information from the user, perform calculations and make decisions.

From here on, we'll start writing programs that employ the basic concepts of object-oriented programming. Programs developed will consist of function main and one or more classes, each containing data members and member functions.

We'll continue the pattern of illustrating concepts and terms with full working programs.



```
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 using namespace std;
11
12 //  GradeBook class definition
13 class GradeBook
14 {
15 public:
16     //  function that displays a welcome message to GradeBook user
17     void displayMessage()
18     {
19         cout << "Welcome to the GradeBook!" << endl;
20     }  //  end function displayMessage
21 };  //  end class GradeBook
22
23 //  function main begins program execution
24 int main()
25 {
26     GradeBook myGradeBook;  // create a GradeBook object named myGradeBook
27     myGradeBook.displayMessage();   //  call object's displayMessage function
28 }   // end main
```

Class GradeBook
The GradeBook class definition (lines 12–21) contains a **member function** called displayMessage (lines 17–20) that displays a message on the screen (line 19).

The class definition begins in line 13 with the **keyword class** followed by the **class name GradeBook**. By convention, the name of a user-defined class begins with a capital letter, and for readability, each subsequent word in the class name begins with a capital letter. This capitalization style is often referred to as Pascal case.

Line 15 contains the keyword public, which is an **access specifier**.

This member function appears after access specifier public: to indicate that the function is "available to the public"—that is, it can be called by other functions in the program (such as main), and by member functions of other classes (if there are any).

Please note:

- Access specifiers are always followed by a colon (:).
- Forgetting the semicolon at the end of a class definition is a syntax error.

Each function in a program performs a task and may return a value when it completes its task—for example, a function might perform a calculation, then return the result of that calculation.

When you define a function, you must specify a return type to indicate the type of the value returned by the function when it completes its task. In line 17, keyword **void** to the left of the function name displayMessage is the function's **return type**. Return type void indicates that displayMessage will not return any data to its calling function when it completes its task.

By convention, function names begin with a lowercase first letter and all subsequent words in the name begin with a capital letter. This capitalization style is often referred to as camel case and is also used for variable names.

The parentheses after the member function name indicate that this is a function. An empty set of parentheses, as shown in line 17, indicates that this member function does not require additional data to perform its task.

Line 17 is commonly referred to as a **function header**. Every function's body is delimited by left and right braces ({ and }), as in lines 18 and 20.

The body of a function contains statements that perform the function's task. In this case, member function displayMessage contains one statement (line 19) that displays the message "Welcome to the Grade Book!". After this statement executes, the function has completed its task.

Testing Class GradeBook
In using the class GradeBook, you'll remember that the function main (lines 25-28) begins the execution of every program.

You cannot call a member function of a class until you create **an object** of that class.

Line 26 creates an object of class GradeBook called myGradeBook. The variable's type is GradeBook—the class we defined in lines 13–21.

When we declare variables of type int, as we did previously, the compiler knows what int is—it's a fundamental type that's "built into" C++.
In line 26, however, the compiler does not automatically know what type GradeBook is—it's a **user-defined type**.

Line 22 calls the member function displayMessage using variable myGradeBook followed by the **dot operator (.),** the function name displayMessage and an empty set of parentheses.

The UML (Unified Modeling Language)

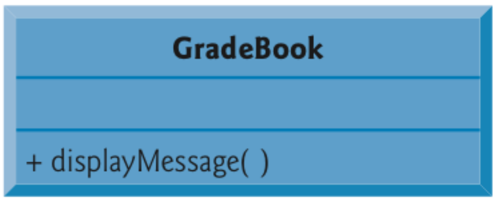
Unified Modeling Language (UML) is the most widely used graphical scheme for modeling object-oriented systems.

UML Class Diagram for Class GradeBook

In the UML, each class is modeled in a UML class diagram as a rectangle with three compartments.

- The *top compartment* contains the class's name centered horizontally and in boldface type.
- The *middle compartment* contains the class's attributes, which correspond to data members in C++. This compartment is currently empty, because class GradeBook does not have any attributes.
- The *bottom compartment* contains the class's operations, which correspond to member functions in C++.

The UML models operations by listing the operation name followed by a set of parentheses.
Class *GradeBook* has only one member function, *displayMessage*, so the bottom compartment lists one operation with this name.
Member function *displayMessage* does not require additional information to perform its tasks, so the parentheses following *displayMessage* in the class diagram are empty.
The plus sign (+) in front of the operation name indicates that *displayMessage* is a public operation in the UML (i.e., a public member function in C++).



Excerpt adapted from C++ How to Program, P.J. Deitel, H.M. Deitel. -- 8th ed