Ekaba Bisong
Programming in C++
University of Calabar

Lesson Note #16
May 13, 2015

Separating Interface from Implementation

Conventional software engineering wisdom says that to use an object of a class, the client code needs to know only:

- what member functions to call,
- what arguments to provide to each member function, and
- what return type to expect from each member function

The client code does not need to know how those functions are implemented.

Separating interface from implementation is a fundamental principle of good software engineering.

Interface of a Class
The interface of a class describes what services a class's clients can use and how to request those services, but not how the class carries out the services.

Separating the Interface from the Implementation
In our prior examples, each class definition contained the complete definitions of the class's public member functions and the declarations of its private data members.

However, it is better software engineering to define member functions outside the class definition, so that their implementation details can be hidden from the client code.

This practice ensures that you do not write client code that depends on the class's implementation details. If you were to do so, the client code would be more likely to "break" if the class's implementation changed. Given that one class could have many clients, such a change could cause wide-ranging problems in a software system.

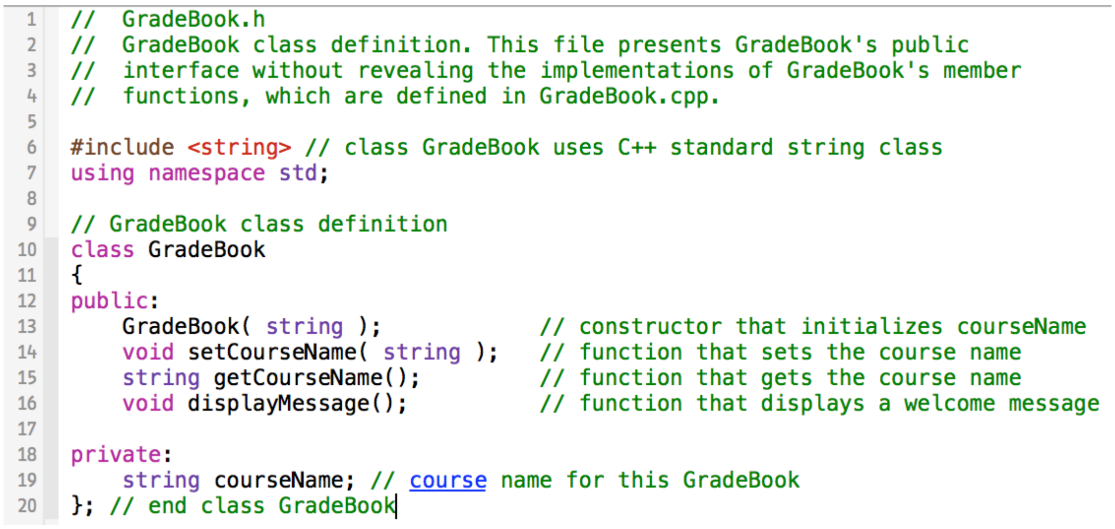
We separate a class interface from its implementation by splitting the class definition into two files:

- the header .h file, in which the class interfaces are defined, and
- a source-code file in which the class member functions are defined.

By convention, member-function definitions are placed in a source-code file of the same base name (e.g., GradeBook) as the class's header but with a .cpp filename extension.

A function prototype is a declaration of a function that tells the compiler the function's name, its return type and the types of its parameters.

GradeBook.h



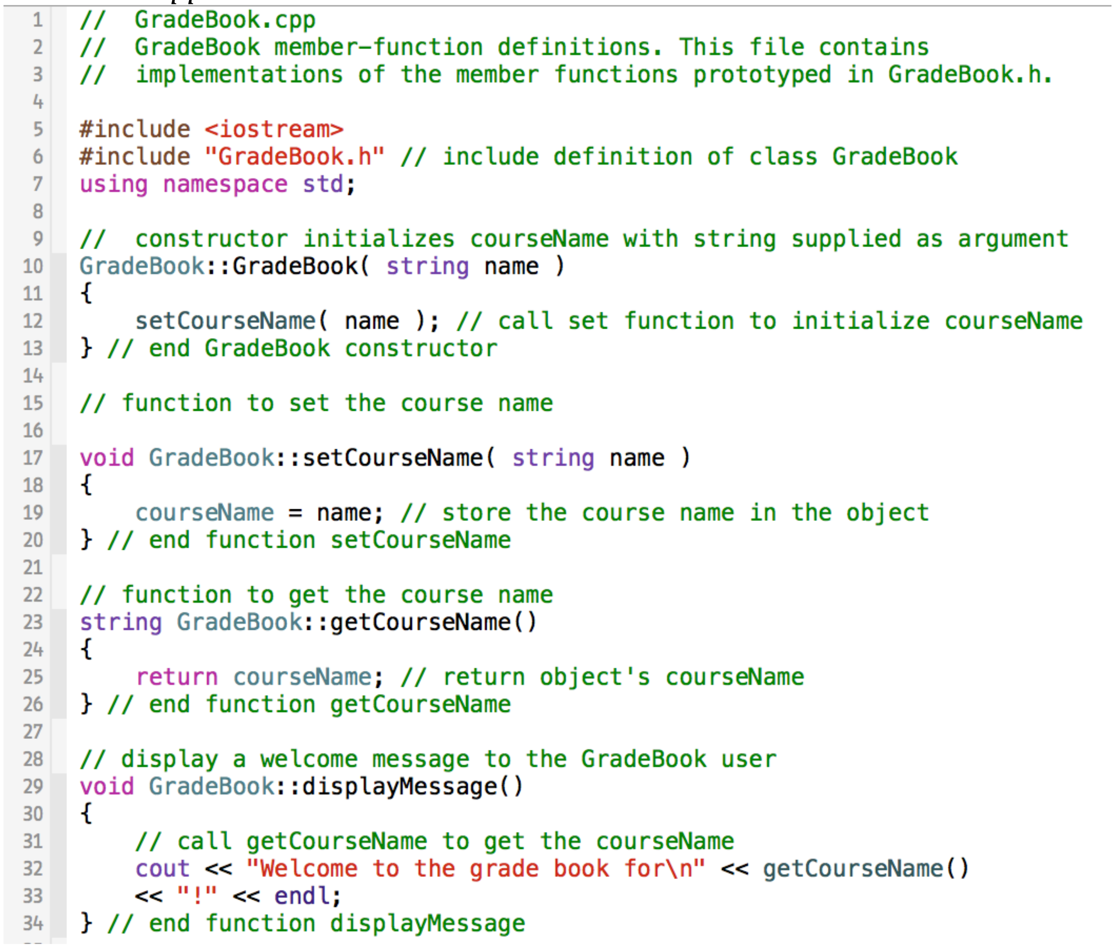
```cpp
//  GradeBook.h
//  GradeBook class definition. This file presents GradeBook's public
//  interface without revealing the implementations of GradeBook's member
//  functions, which are defined in GradeBook.cpp.

#include <string> // class GradeBook uses C++ standard string class
using namespace std;

// GradeBook class definition
class GradeBook
{
public:
    GradeBook( string );            // constructor that initializes courseName
    void setCourseName( string );   // function that sets the course name
    string getCourseName();         // function that gets the course name
    void displayMessage();          // function that displays a welcome message

private:
    string courseName; // course name for this GradeBook
}; // end class GradeBook
```

GradeBook.cpp

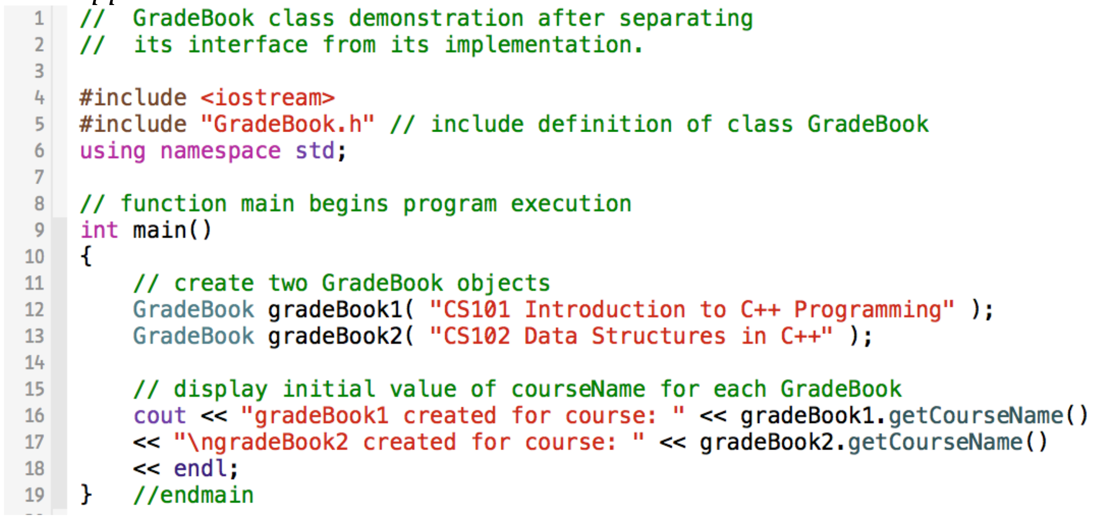


```cpp
1  //  GradeBook.cpp
2  //  GradeBook member-function definitions. This file contains
3  //  implementations of the member functions prototyped in GradeBook.h.
4
5  #include <iostream>
6  #include "GradeBook.h" // include definition of class GradeBook
7  using namespace std;
8
9  //  constructor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

Each member-function name in the function headers (lines 10, 17, 23 and 29) 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, 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.

Main.cpp



```cpp
//  GradeBook class demonstration after separating
//  its interface from its implementation.

#include <iostream>
#include "GradeBook.h" // include definition of class GradeBook
using namespace std;

// function main begins program execution
int main()
{
    // create two GradeBook objects
    GradeBook gradeBook1( "CS101 Introduction to C++ Programming" );
    GradeBook gradeBook2( "CS102 Data Structures in C++" );

    // display initial value of courseName for each GradeBook
    cout << "gradeBook1 created for course: " << gradeBook1.getCourseName()
    << "\ngradeBook2 created for course: " << gradeBook2.getCourseName()
    << endl;
}   //endmain
```

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