
C Programming for Engineers

Object Oriented Programming



UNIVERSITY
AT ALBANY
State University of New York

ICEN 360– Spring 2017
Prof. Dola Saha

Evolution of computers

- Early computers were far less complex than today's computers
- Modern computers are smaller, but more complex



Objects

- Computer scientists have introduced the notion of ***objects*** and ***object-oriented programming*** to help manage the growing complexity of modern computers.
- Object is anything that can be represented by data in computer's memory



Properties

- The data that represent the object are organized into a set of **properties**.
- The values stored in an object's properties at any one time form the **state** of an object.

Name: PA 3794

Owner: US Airlines

Location: 39 52' 06" N 75 13' 52" W

Heading: 271°

Altitude: 19 m

AirSpeed: 0

Make: Boeing

Model: 737

Weight: 32,820 kg

Methods

- In object-oriented programming, the programs that manipulate the properties of an object are the object's **methods**.
- We can think of an object as *a collection of properties and the methods* that are used to manipulate those properties.

Class

- A **class** is a group of objects with the same properties and the same methods.

Class
<CAR>



Object



<7_series_BMW>

Object



<Ford_Mustang>

Object



<VW_Beetle>

Instance

- Each copy of an object from a particular class is called an *instance* of the object.
- The act of creating a new instance of an object is called **instantiation**.
- Two different instances of the same class will have the **same properties**, **but different values** stored in those properties.



Terminology

Object

Property

Method

The same terminology is used
in most object-oriented
programming languages.

Instantiation

Instance

State

Class

First OOP in C++

```
1 // Fig. 3.1: fig03_01.cpp
2 // Define class GradeBook with a member function displayMessage,
3 // create a GradeBook object, and call its displayMessage function.
4 #include <iostream>
5 using namespace std;
6
7 // GradeBook class definition
8 class GradeBook
9 {
10 public:
11     // function that displays a welcome message to the GradeBook user
12     void displayMessage() const
13     {
14         cout << "Welcome to the Grade Book!" << endl;
15     } // end function displayMessage
16 }; // end class GradeBook
17
18 // function main begins program execution
19 int main()
20 {
21     GradeBook myGradeBook; // create a GradeBook object named myGradeBook
22     myGradeBook.displayMessage(); // call object's displayMessage function
23 } // end main
```

Welcome to the Grade Book!

Access Specifier: Public & Private

- Keyword public or private is an access specifier.
- Access specifiers are always followed by a colon (:)

- Public:
 - Accessible to public—that is, it can be called by other functions in the program (such as main), and by member functions / methods of other classes (if there are any).
- Private:
 - Accessible only to member functions / methods of the class for which they are declared.

Passing value

```
1 // Fig. 3.3: fig03_03.cpp
2 // Define class GradeBook with a member function that takes a parameter,
3 // create a GradeBook object and call its displayMessage function.
4 #include <iostream>
5 #include <string> // program uses C++ standard string class
6 using namespace std;
7
8 // GradeBook class definition
9 class GradeBook
10 {
11 public:
12     // function that displays a welcome message to the GradeBook user
13     void displayMessage( string courseName ) const
14     {
15         cout << "Welcome to the grade book for\n" << courseName << "!"
16             << endl;
17     } // end function displayMessage
18 }; // end class GradeBook
19
20 // function main begins program execution
21 int main()
22 {
23     string nameOfCourse; // string of characters to store the course name
24     GradeBook myGradeBook; // create a GradeBook object named myGradeBook
25
26     // prompt for and input course name
27     cout << "Please enter the course name:" << endl;
28     getline( cin, nameOfCourse ); // read a course name with blanks
29     cout << endl; // output a blank line
30
31     // call myGradeBook's displayMessage function
32     // and pass nameOfCourse as an argument
33     myGradeBook.displayMessage( nameOfCourse );
34 }
```



Classroom Assignment

- Extend the second OOP to
 - Call the method twice with two different messages
 - Instantiate two gradebooks and display different messages

Example object with properties & methods (1)

```
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
```

Example object with properties & methods (2)

```
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;
```

Example object with properties & methods (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!

Constructor and Destructor

- A **constructor** is a special function that gets called automatically when the object of a class is created.
- A **destructor** is a special function that gets called automatically when an object is deleted.

Constructor Example

```
1 // Fig. 3.7: fig03_07.cpp
2 // Instantiating multiple objects of the GradeBook class and using
3 // the GradeBook constructor to specify the course name
4 // when each GradeBook object is created.
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     // constructor initializes courseName with string supplied as argument
14     explicit GradeBook( string name )
15         : courseName( name ) // member initializer to initialize courseName
16     {
17         // empty body
18     } // end GradeBook constructor
19
```

Constructor Example

```
20 // function to set the course name
21 void setCourseName( string name )
22 {
23     courseName = name; // store the course name in the object
24 } // end function setCourseName
25
26 // function to get the course name
27 string getCourseName() const
28 {
29     return courseName; // return object's courseName
30 } // end function getCourseName
31
32 // display a welcome message to the GradeBook user
33 void displayMessage() const
34 {
35     // call getCourseName to get the courseName
36     cout << "Welcome to the grade book for\n" << getCourseName()
37         << "!" << endl;
38 } // end function displayMessage
```

Constructor Example

```
39 private:
40     string courseName; // course name for this GradeBook
41 }; // end class GradeBook
42
43 // function main begins program execution
44 int main()
45 {
46     // create two GradeBook objects
47     GradeBook gradeBook1( "CS101 Introduction to C++ Programming" );
48     GradeBook gradeBook2( "CS102 Data Structures in C++" );
49
50     // display initial value of courseName for each GradeBook
51     cout << "gradeBook1 created for course: " << gradeBook1.getCourseName()
52         << "\ngradeBook2 created for course: " << gradeBook2.getCourseName()
53         << endl;
54 } // end main
```

```
gradeBook1 created for course: CS101 Introduction to C++ Programming
gradeBook2 created for course: CS102 Data Structures in C++
```

Separate Function definition & declaration

```
#include <string> // program uses C++ standard string class
#include <iostream>
using namespace std;

// GradeBook class definition
class GradeBook
{
public:
    // constructor initializes course name and instructor name
    GradeBook( std::string, std::string );
    void setCourseName( std::string ); // function to set the course name
    std::string getCourseName(); // function to retrieve the course name
    void setInstructorName( std::string ); // function to set instructor name
    std::string getInstructorName(); // function to retrieve instructor name
    void displayMessage(); // display welcome message and instructor name
private:
    std::string courseName; // course name for this GradeBook
    std::string instructorName; // instructor name for this GradeBook
}; // end class GradeBook
```

Separate Function definition & declaration

```
// constructor initializes courseName and instructorName
// with strings supplied as arguments
GradeBook::GradeBook( string course, string instructor )
{
    setCourseName( course ); // initializes courseName
    setInstructorName( instructor ); // initializes instructorName
} // end GradeBook constructor

// function to set the course name
void GradeBook::setCourseName( string name )
{
    courseName = name; // store the course name
} // end function setCourseName

// function to retrieve the course name
string GradeBook::getCourseName()
{
    return courseName;
} // end function getCourseName

// function to set the instructor name
void GradeBook::setInstructorName( string name )
{
    instructorName = name; // store the instructor name
} // end function setInstructorName

// function to retrieve the instructor name
string GradeBook::getInstructorName()
{
    return instructorName;
} // end function getInstructorName

// display a welcome message and the instructor's name
void GradeBook::displayMessage()
{
    // display a welcome message containing the course name
    cout << "Welcome to the grade book for\n" << getCourseName() << "!"
        << endl;

    // display the instructor's name
    cout << "This course is presented by: " << getInstructorName() << endl;
} // end function displayMessage
```

Separate Function definition & declaration

```
// function main begins program execution
int main()
{
    // create a GradeBook object; pass a course name and instructor name
    GradeBook gradeBook(
        "CS101 Introduction to C++ Programming", "Professor Smith" );

    // display initial value of instructorName of GradeBook object
    cout << "gradeBook instructor name is: "
        << gradeBook.getInstructorName() << "\n\n";

    // modify the instructorName using set function
    gradeBook.setInstructorName( "Assistant Professor Bates" );

    // display new value of instructorName
    cout << "new gradeBook instructor name is: "
        << gradeBook.getInstructorName() << "\n\n";

    // display welcome message and instructor's name
    gradeBook.displayMessage();
} // end main
```