Lecture 15:
Introduction to Classes and Objects

Ioan Raicu
Department of Electrical Engineering & Computer Science
Northwestern University

EECS 211
Fundamentals of Computer Programming II
April 21st, 2010

3.2 Classes, Objects, Member Functions and Data Members

- Suppose you want to drive a car and make it go faster by pressing down on its accelerator pedal.
- Before you can drive a car, someone has to *design it and build it.*
- A car typically begins as engineering drawings that include the design for an accelerator pedal that makes the car go faster.
- The pedal "hides" the complex mechanisms that actually make the car go faster, just as the brake pedal "hides" the mechanisms that slow the car, the steering wheel "hides" the mechanisms that turn the car and so on.
- This enables people with little or no knowledge of how cars are engineered to drive a car easily, simply by using the accelerator pedal, the brake pedal, the steering wheel, the transmission shifting mechanism and other such simple and user-friendly "interfaces" to the car's complex internal mechanisms.

3.2 Classes, Objects, Member Functions and Data Members (cont.)

- In C++, we begin by creating a program unit called a class to house a function, just as a car's engineering drawings house the design of an accelerator pedal.
- Recall from Section 1.19 that a function belonging to a class is called a member function.
- In a class, you provide one or more member functions that are designed to perform the class's tasks.
- Just as you cannot drive an engineering drawing of a car, you cannot "drive" a class.
  - *You must create an object of a class before you can get a program to perform the tasks the class describes.*
- That is one reason C++ is known as an object-oriented programming language.
- Just as many cars can be built from the same engineering drawing, many objects can be built from the same class.

3.2 Classes, Objects, Member Functions and Data Members (cont.)

- When you drive a car, pressing its gas pedal sends a message to the car to perform a task—that is, make the car go faster.
- Similarly, you send messages to an object—each message is known as a member-function call and tells a member function of the object to perform its task.
- This is often called requesting a service from an object.

3.2 Classes, Objects, Member Functions and Data Members (cont.)

- In addition to the capabilities a car provides, it also has many attributes, such as its color, the number of doors, the amount of gas in its tank, its current speed and its total miles driven (i.e., its odometer reading).
  - Like the car's capabilities, these attributes are represented as part of a car's design in its engineering diagrams.
  - As you drive a car, these attributes are always associated with the car.
  - Every car maintains its own attributes.
- Similarly, an object has attributes that are carried with the object as it's used in a program.
  - These attributes are specified as part of the object's class.
  - Attributes are specified by the class's data members.

3.3 Defining a Class with a Member Function

- We begin with an example (Fig. 3.1) that consists of class `GradeBook` (lines 8–16), which, when it is fully developed in Chapter 7, will represent a grade book that an instructor can use to maintain student test scores, and a `main` function (lines 19–23) that creates a `GradeBook` object.
- Function `main` uses this object and its member function to display a message on the screen welcoming the instructor to the grade-book program.

3.3 Defining a Class with a Member Function

```
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()
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
```

Fig. 3.1 | Define class `GradeBook` with a member function `displayMessage`, create a `GradeBook` object and call its `displayMessage` function.

3.3 Defining a Class with a Member Function (cont.)

- Function `main` is always called automatically when you execute a program.
- Most functions do not get called automatically.
- You must call member function `displayMessage` explicitly to tell it to perform its task.
- The access-specifier label `public:` contains the keyword `public` is an access specifier.
  - Indicates 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).
  - Access specifiers are always followed by a colon (`:`).

3.3 Defining a Class with a Member Function (cont.)

- Typically, you cannot call a member function of a class until you create an object of that class.
- First, create an object of class `GradeBook` called `myGradeBook`.
  - The variable's type is `GradeBook`.
  - The compiler does not automatically know what type `GradeBook` is—it's a user-defined type.
  - Tell the compiler what `GradeBook` is by including the class definition.
  - Each class you create becomes a new type that can be used to create objects.
- Call the member function `displayMessage-` by using variable `myGradeBook` followed by the dot operator (`.`), the function name `display-Message` and an empty set of parentheses.
  - Causes the `displayMessage` function to perform its task.

3.4 Defining a Member Function with a Parameter

- Car analogy
  - Pressing a car's gas pedal sends a message to the car to perform a task—make the car go faster.
  - But how fast should the car accelerate? As you know, the farther down you press the pedal, the faster the car accelerates.
  - The message to the car includes both the task to perform and additional information that helps the car perform the task.
- Additional information that a function needs to perform its task is known as a parameter.
- A function call supplies values—called arguments—for each of the function's parameters.

3.4 Defining a Member Function with a Parameter (cont.)

- Fig. 3.3 redefines class `GradeBook` (lines 9–18) with a `display-Message` member function (lines 13–17) that displays the course name as part of the welcome message.
  - The new version of `displayMessage` requires a parameter (`courseName` in line 13) that represents the course name to output.
- A variable of type string represents a string of characters.
- A string is actually an object of the C++ Standard Library class `string`.
  - Defined in header file <string> and part of namespace `std`.
  - For now, you can think of `string` variables like variables of other types such as `int`.
  - Additional `string` capabilities in Section 3.9.

3.4 Defining a Member Function with a Parameter (cont.)

```
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 )
14    {
15       cout << "Welcome to the grade book for\n" << courseName << "!"
16          << endl;
17    } // end function displayMessage
18 }; // end class GradeBook
19
```

Fig. 3.3 | Define class `GradeBook` with a member function that takes a parameter, create a `GradeBook` object and call its `displayMessage` function. (Part 1 of 2.)

3.4 Defining a Member Function with a Parameter (cont.)

```
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  } // end main
```

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

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

Fig. 3.3 | Define class `GradeBook` with a member function that takes a parameter, create a `GradeBook` object and call its `displayMessage` function. (Part 2 of 2.)

3.5 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.

3.5 Data Members, set Functions and get Functions (cont.)

- An object has attributes that are carried with it as it's used in a program.
  - Such attributes exist throughout the life of the object.
  - 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.

3.5 Data Members, set Functions and get Functions (cont.)

- A typical instructor teaches multiple courses, each with its own course name.
- A variable that is declared in the class definition but outside the bodies of the class's member-function definitions is a data member.
- Every instance (i.e., object) of a class contains one copy of each of the class's data members.
- A benefit of making a variable a data member is that all the member functions of the class can manipulate any data members that appear in the class definition.

3.5 Data Members, set Functions and get Functions (cont.)

```
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()
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* functions. (Part 1 of 3.)

3.5 Data Members, set Functions and get Functions (cont.)

```
24
25    // function that displays a welcome message
26    void displayMessage()
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* functions. (Part 2 of 3.)

3.5 Data Members, set Functions and get Functions (cont.)

```
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* functions. (Part 3 of 3.)

3.5 Data Members, set Functions and get 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.

3.5 Data Members, set Functions and get Functions (cont.)

Software Engineering Observation 3.1

Generally, data members should be declared `private` *and member functions should be declared* `public`. *(We'll see that it's appropriate to declare certain member functions* `private`*, if they're to be accessed only by other member functions of the class.)*

3.5 Data Members, set Functions and get Functions (cont.)

Common Programming Error 3.6

An attempt by a function, which is not a member of a particular class (or a friend of that class, as we'll see in Chapter 10, Classes: A Deeper Look, Part 2), to access a `private` *member of that class is a compilation error.*

3.5 Data Members, *set* Functions and *get* 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.

3.5 Data Members, set Functions and get 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).

Questions