Review of Lecture1 ??

To review Lecture1, you may examine the codes of Chapter 15 of the textbook. Related materials are available at http://media.pearsoncmg.com/ph/esm/deitel/C_HTP7e/CodeExamples/ch15.zip .

Lecture2

Chapter 16: Introduction to Classes and Objects

Classes and Objects

- class declarations
- information hiding in C++
- member selector operator
- class scope
- defining class methods
- using classes in a program

GradeBook example at Chapter16 of the textbook

```
1  // Fig. 16.1: fig16_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. 16.1 | Define class `GradeBook` with a member function `displayMessage`, create a `GradeBook` object and call its `displayMessage` function.

UML diagram of the previous GradeBook class

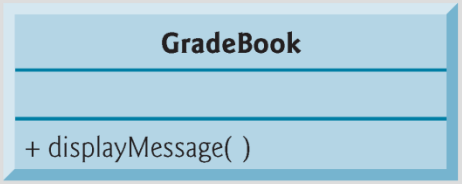

Fig. 16.2 | UML class diagram indicating that class `GradeBook` has a public `displayMessage` operation.

GradeBook example 2

```
1  // Fig. 16.3: fig16_016.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. 16.3 | Define class `GradeBook` with a member function that takes a parameter, create a `GradeBook` object and call its `displayMessage` function. (Part 1 of 2.)

GradeBook example 2 continues ...

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

UML diagram of GradeBook in example 2 ...

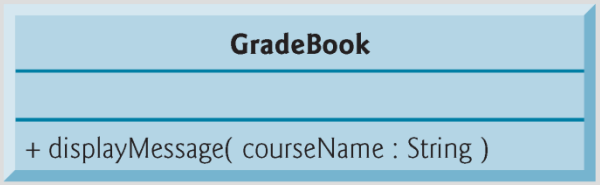

Fig. 16.4 | UML class diagram indicating that class `GradeBook` has a public `displayMessage` operation with a `courseName` parameter of UML type `String`.

GradeBook example 3 (set and get methods)

```
1  // Fig. 16.5: fig16_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. 16.5 | Defining and testing class `GradeBook` with a data member and *set* and *get* functions. (Part 1 of 3.)

GradeBook example 3 continues ...

```
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. 16.5 | Defining and testing class GradeBook with a data member and *set* and *get* functions. (Part 2 of 3.)

GradeBook example 3 continues ...

```
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. 16.5 | Defining and testing class GradeBook with a data member and *set* and *get* functions. (Part 3 of 3.)

UML diagram of GradeBook in example 3 ...

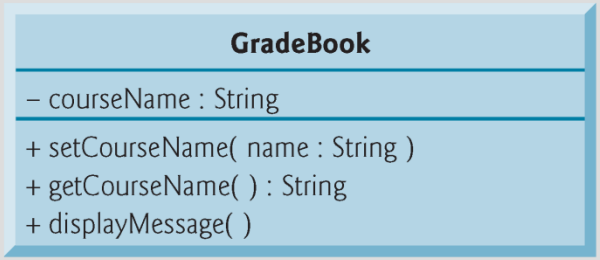

Fig. 16.6 | UML class diagram for class `GradeBook` with a private `courseName` attribute and public operations `setCourseName`, `getCourseName` and `displayMessage`.

GradeBook example 4 (constructor)

```
1  // Fig. 16.7: fig16_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    GradeBook( string name )
15    {
16       setCourseName( name ); // call set function to initialize courseName
17    } // end GradeBook constructor
18
```

Fig. 16.7 | Instantiating multiple objects of the `GradeBook` class and using the `GradeBook` constructor to specify the course name when each `GradeBook` object is created. (Part 1 of 3.)

GradeBook example 4 continues ...

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

Fig. 16.7 | Instantiating multiple objects of the `GradeBook` class and using the `GradeBook` constructor to specify the course name when each `GradeBook` object is created. (Part 2 of 3.)

GradeBook example 4 continues ...

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

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

Fig. 16.7 | Instantiating multiple objects of the GradeBook class and using the GradeBook constructor to specify the course name when each GradeBook object is created. (Part 3 of 3.)

UML diagram of GradeBook in example 4 ...

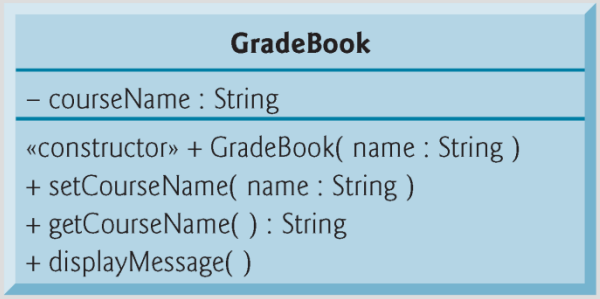

Fig. 16.8 | UML class diagram indicating that class `GradeBook` has a constructor with a `name` parameter of UML type `String`.

GradeBook example 5 (separate header file)

```
1  // Fig. 16.9: GradeBook.h
2  // GradeBook class definition in a separate file from main.
3  #include <iostream>
4  #include <string> // class GradeBook uses C++ standard string class
5  using namespace std;
6
7  // GradeBook class definition
8  class GradeBook
9  {
10 public:
11    // constructor initializes courseName with string supplied as argument
12    GradeBook( string name )
13    {
14       setCourseName( name ); // call set function to initialize courseName
15    } // end GradeBook constructor
16
17    // function to set the course name
18    void setCourseName( string name )
19    {
20       courseName = name; // store the course name in the object
21    } // end function setCourseName
22
```

Fig. 16.9 | GradeBook class definition in a separate file from `main`. (Part 1 of 2.)

GradeBook example 5 continues ...

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

Fig. 16.9 | GradeBook class definition in a separate file from `main`. (Part 2 of 2.)

GradeBook example 5 continues ...

```
1  // Fig. 16.10: fig16_10.cpp
2  // Including class GradeBook from file GradeBook.h for use in main.
3  #include <iostream>
4  #include "GradeBook.h" // include definition of class GradeBook
5  using namespace std;
6
7  // function main begins program execution
8  int main()
9  {
10    // create two GradeBook objects
11    GradeBook gradeBook1( "CS101 Introduction to C++ Programming" );
12    GradeBook gradeBook2( "CS102 Data Structures in C++" );
13
14    // display initial value of courseName for each GradeBook
15    cout << "gradeBook1 created for course: " << gradeBook1.getCourseName()
16       << "\ngradeBook2 created for course: " << gradeBook2.getCourseName()
17       << endl;
18 } // end main
```

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

Fig. 16.10 | Including class `GradeBook` from file `GradeBook.h` for use in `main`.

GradeBook example 6 (separate interface from implementation)

```
1  // Fig. 16.11: GradeBook.h
2  // GradeBook class definition. This file presents GradeBook's public
3  // interface without revealing the implementations of GradeBook's member
4  // functions, which are defined in GradeBook.cpp.
5  #include <string> // class GradeBook uses C++ standard string class
6  using namespace std;
7
8  // GradeBook class definition
9  class GradeBook
10 {
11 public:
12    GradeBook( string ); // constructor that initializes courseName
13    void setCourseName( string ); // function that sets the course name
14    string getCourseName(); // function that gets the course name
15    void displayMessage(); // function that displays a welcome message
16 private:
17    string courseName; // course name for this GradeBook
18 }; // end class GradeBook
```

Fig. 16.11 | `GradeBook` class definition containing function prototypes that specify the interface of the class.

GradeBook example 6 continues ...

```
1  // Fig. 16.12: GradeBook.cpp
2  // GradeBook member-function definitions. This file contains
3  // implementations of the member functions prototyped in GradeBook.h.
4  #include <iostream>
5  #include "GradeBook.h" // include definition of class GradeBook
6  using namespace std;
7
8  // constructor initializes courseName with string supplied as argument
9  GradeBook::GradeBook( string name )
10 {
11    setCourseName( name ); // call set function to initialize courseName
12 } // end GradeBook constructor
13
14 // function to set the course name
15 void GradeBook::setCourseName( string name )
16 {
17    courseName = name; // store the course name in the object
18 } // end function setCourseName
19
```

Fig. 16.12 | GradeBook member-function definitions represent the implementation of class GradeBook. (Part 1 of 2.)

GradeBook example 6 continues ...

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

Fig. 16.12 | GradeBook member-function definitions represent the implementation of class GradeBook. (Part 2 of 2.)

GradeBook example 6 continues ...

```
1  // Fig. 16.13: fig16_13.cpp
2  // GradeBook class demonstration after separating
3  // its interface from its implementation.
4  #include <iostream>
5  #include "GradeBook.h" // include definition of class GradeBook
6  using namespace std;
7
8  // function main begins program execution
9  int main()
10 {
11    // create two GradeBook objects
12    GradeBook gradeBook1( "CS101 Introduction to C++ Programming" );
13    GradeBook gradeBook2( "CS102 Data Structures in C++" );
14
15    // display initial value of courseName for each GradeBook
16    cout << "gradeBook1 created for course: " << gradeBook1.getCourseName()
17       << "\ngradeBook2 created for course: " << gradeBook2.getCourseName()
18       << endl;
19 } // end main
```

Fig. 16.13 | GradeBook class demonstration after separating its interface from its implementation. (Part 1 of 2.)

GradeBook example 6 continues ...

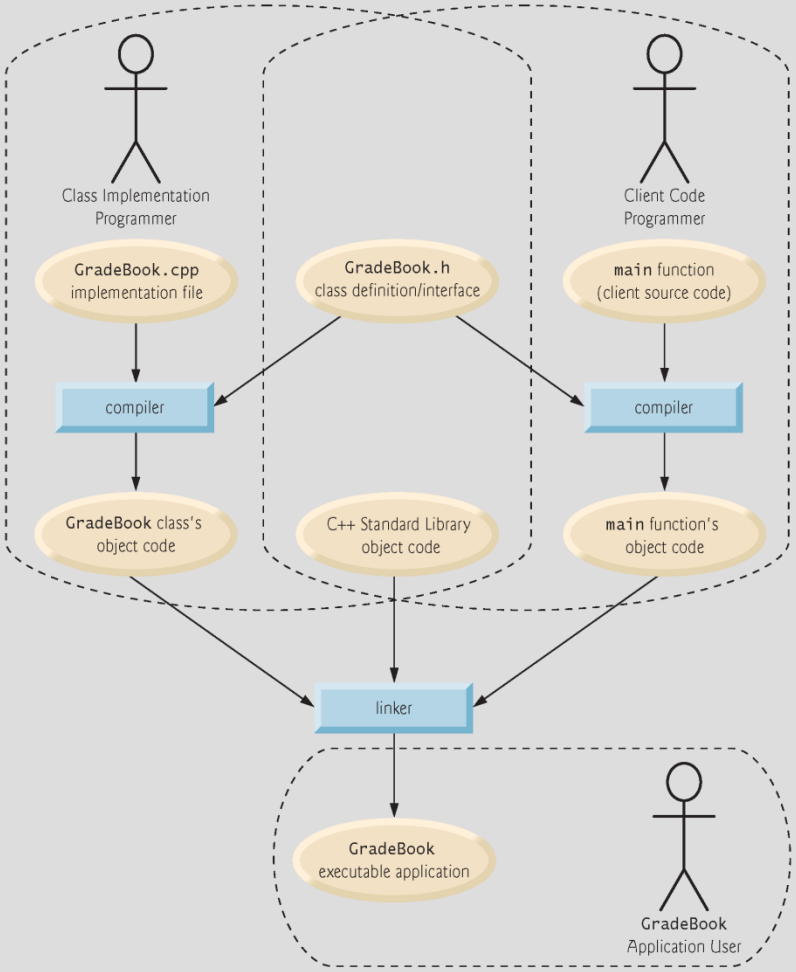

Fig. 16.14 | Compilation and linking process that produces an executable

GradeBook example 7
(validating data)

```
1  // Fig. 16.11: GradeBook.h
2  // GradeBook class definition. This file presents GradeBook's public
3  // interface without revealing the implementations of GradeBook's member
4  // functions, which are defined in GradeBook.cpp.
5  #include <string> // class GradeBook uses C++ standard string class
6  using namespace std;
7
8  // GradeBook class definition
9  class GradeBook
10 {
11 public:
12    GradeBook( string ); // constructor that initializes courseName
13    void setCourseName( string ); // function that sets the course name
14    string getCourseName(); // function that gets the course name
15    void displayMessage(); // function that displays a welcome message
16 private:
17    string courseName; // course name for this GradeBook
18 }; // end class GradeBook
```

Fig. 16.11 | `GradeBook` class definition containing function prototypes that specify the interface of the class.

GradeBook example 7 continues ...

```
1  // Fig. 16.16: GradeBook.cpp
2  // Implementations of the GradeBook member-function definitions.
3  // The setCourseName function performs validation.
4  #include <iostream>
5  #include "GradeBook.h" // include definition of class GradeBook
6  using namespace std;
7
8  // constructor initializes courseName with string supplied as argument
9  GradeBook::GradeBook( string name )
10 {
11    setCourseName( name ); // validate and store courseName
12 } // end GradeBook constructor
13
14 // function that sets the course name;
15 // ensures that the course name has at most 25 characters
16 void GradeBook::setCourseName( string name )
17 {
18    if ( name.length() <= 25 ) // if name has 25 or fewer characters
19       courseName = name; // store the course name in the object
20
```

Fig. 16.16 | Member-function definitions for class `GradeBook` with a *set* function that validates the length of data member `courseName`. (Part 1 of 2.)

GradeBook example 7 continues ...

```
21    if ( name.length() > 25 ) // if name has more than 25 characters
22    {
23       // set courseName to first 25 characters of parameter name
24       courseName = name.substr( 0, 25 ); // start at 0, length of 25
25
26       cout << "Name \"" << name << "\" exceeds maximum length (25).\n"
27          << "Limiting courseName to first 25 characters.\n" << endl;
28    } // end if
29 } // end function setCourseName
30
31 // function to get the course name
32 string GradeBook::getCourseName()
33 {
34    return courseName; // return object's courseName
35 } // end function getCourseName
36
37 // display a welcome message to the GradeBook user
38 void GradeBook::displayMessage()
39 {
40    // call getCourseName to get the courseName
41    cout << "Welcome to the grade book for\n" << getCourseName()
42       << "!" << endl;
43 } // end function displayMessage
```

Fig. 16.16 | Member-function definitions for class `GradeBook` with a *set* function that validates the length of data member `courseName`. (Part 2 of 2.)

GradeBook example 7 continues ...

```
1  // Fig. 16.17: fig16_17.cpp
2  // Create and manipulate a GradeBook object; illustrate validation.
3  #include <iostream>
4  #include "GradeBook.h" // include definition of class GradeBook
5  using namespace std;
6
7  // function main begins program execution
8  int main()
9  {
10    // create two GradeBook objects;
11    // initial course name of gradeBook1 is too long
12    GradeBook gradeBook1( "CS101 Introduction to Programming in C++" );
13    GradeBook gradeBook2( "CS102 C++ Data Structures" );
14
15    // display each GradeBook's courseName
16    cout << "gradeBook1's initial course name is: "
17       << gradeBook1.getCourseName()
18       << "\ngradeBook2's initial course name is: "
19       << gradeBook2.getCourseName() << endl;
20
21    // modify myGradeBook's courseName (with a valid-length string)
22    gradeBook1.setCourseName( "CS101 C++ Programming" );
23
```

Fig. 16.17 | Creating and manipulating a `GradeBook` object in which the course name is limited to 25 characters in length. (Part 1 of 2.)

GradeBook example 7 continues ...

```
24      // display each GradeBook's courseName
25      cout << "\ngradeBook1's course name is: "
26         << gradeBook1.getCourseName()
27         << "\ngradeBook2's course name is: "
28         << gradeBook2.getCourseName() << endl;
29   } // end main
```

```
Name "CS101 Introduction to Programming in C++" exceeds maximum length (25).
Limiting courseName to first 25 characters.

gradeBook1's initial course name is: CS101 Introduction to Pro
gradeBook2's initial course name is: CS102 C++ Data Structures

gradeBook1's course name is: CS101 C++ Programming
gradeBook2's course name is: CS102 C++ Data Structures
```

Fig. 16.17 | Creating and manipulating a `GradeBook` object in which the course name is limited to 25 characters in length. (Part 2 of 2.)

TO DO @ HOME
(Sample Applications)

- Think about the "***abstraction***" concept and declare the following class infrastructures:
  - Car class
  - Student class

- Implement the Car and the Student classes above. Test those class implementations through an appropiate main function.

some other slides??

- You shall study the slides of the textbook.
- You may also study the slides offered by Dr. Ufuk Çelikkan.