

Classes and Objects



Systems Programming

Classes and Objects

- **Class Definitions and Objects**
- **Member functions**
- **Data members**
 - **Get and Set functions**
 - **Constructors**
- **Placing Classes in Separate Files**
- **Separating interface from implementation**
- **Data validation**
 - **Ensures that data in an object is in a particular format or range.**

C++ Program Structure

- Typically C++ Programs will consist of
 - A function **main**
 - One or more classes
 - Each containing **data members** and **member functions**.

19.4 Defining a Class With a Member Function

- **Class definition**
 - Tells the compiler what **member functions** and **data members** belong to the class.
 - Keyword **class** followed by the class's name.
 - Class body is enclosed in braces (**{}**)
 - Specifies data members and member functions
 - Access-specifier **public**:
 - Indicates that a member function or data member is accessible to other functions and member functions of other classes.

© 2007 Pearson Ed -All rights reserved.

C++ Gradebook Example

```
1 // Fig. 19.1: fig19_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 std::cout;
6 using std::endl;
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()
14     {
15         cout << "Welcome to the Grade Book!" << endl;
16     } // end function displayMessage
17 }; // end class GradeBook
18
19 // function main begins program execution
20 int main()
21 {
22     GradeBook myGradeBook; // create a GradeBook object named myGradeBook
23     myGradeBook.displayMessage(); // call object's displayMessage function
24     return 0; // indicate successful termination
25 } // end main
```

Beginning of class definition
for class **GradeBook**

Beginning of class body

Access specifier **public**; makes
members available to the public

Member function **displayMessage**
returns nothing

End of class body

Use dot operator to call
GradeBook's member function

© 2007 Pearson Ed -All rights reserved.

Member function takes a parameter

```
1 // Fig. 19.3: fig19_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 using std::cout;
6 using std::cin;
7 using std::endl;
8
9 #include <string> // program uses C++ standard string class
10 using std::string;
11 using std::getline;
12
13 // GradeBook class definition
14 class GradeBook
15 {
16 public:
17     // function that displays a welcome message to the GradeBook user
18     void displayMessage( string courseName )
19     {
20         cout << "Welcome to the grade book for\n" << courseName << "! "
21             << endl;
22     } // end function displayMessage
23 }; // end class GradeBook
24
25 // function main begins program execution
26 int main()
27 {
28     string nameOfCourse; // string of characters to store the course name
29     GradeBook myGradeBook; // create a GradeBook object named myGradeBook
30
```

Include string class
definition

Member function
parameter

Use the function
parameter as a
variable

© 2007 Pearson Ed -All rights reserved.

Member function takes a parameter

```
31 // prompt for and input course name
32 cout << "Please enter the course name:" << endl;
33 getline( cin, nameOfCourse ); // read a course name with blanks
34 cout << endl; // output a blank line
35
36 // call myGradeBook's displayMessage function
37 // and pass nameOfCourse as an argument
38 myGradeBook.displayMessage( nameOfCourse );
39 return 0; // indicate successful termination
40 } // end main
```

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

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

Passing an argument to
the member function

© 2007 Pearson Ed -All rights reserved.

Member function takes a parameter

- A string
 - Represents a string of characters.
 - An object of C++ Standard Library class `std::string`
 - Defined in header file `<string>`.
- Library function `getline`
 - Used to retrieve input until newline is encountered
 - Example
 - `getline(cin, nameOfCourse);`
 - Inputs a line from standard input into string object `nameOfCourse`.

© 2007 Pearson Ed -All rights reserved.

19.6 Data Members, *set* Functions and *get* Functions

- Local variables
 - Variables declared in a function definition's body cannot be used outside of that function body.
 - When a function terminates the values of its local variables are lost.
- Attributes
 - Exist throughout the life of the object.
 - Are represented as data members
 - Namely, variables in a class definition
 - Each object of class maintains **its own copy** of attributes.

19.6 Data Members, *set* Functions and *get* Functions

- Access-specifier **private**
 - Makes a data member or member function accessible only to member functions of the class.
 - **private** is the default access for class members.
 - “**information hiding**” is an object-oriented tenet.
- Returning a value from a function
 - A function that specifies a return type other than **void**
 - Returns a value to its calling function.

© 2007 Pearson Ed -All rights reserved.

19.6 Data Members, *set* Functions and *get* Functions

```
1 // Fig. 19.5: fig19_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 using std::cout;
7 using std::cin;
8 using std::endl;
9
10 #include <string> // program uses C++ standard string class
11 using std::string;
12 using std::getline;
13
14 // GradeBook class definition
15 class GradeBook
16 {
17 public:
18     // function that sets the course name
19     void setCourseName( string name )
20     {
21         courseName = name; // store the course name in the object
22     } // end function setCourseName
23
24     // function that gets the course name
25     string getCourseName()
26     {
27         return courseName; // return the object's courseName
28     } // end function getCourseName
29
```

set function modifies **private** data

get function accesses **private** data

© 2007 Pearson Ed -All rights reserved.

19.6 Data Members, *set* Functions and *get* Functions

```
30 // function that displays a welcome message
31 void displayMessage()
32 {
33     // this statement calls getCourseName to get the
34     // name of the course this GradeBook represents
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
41
42 // function main begins program execution
43 int main()
44 {
45     string nameOfCourse; // string of characters to store the course name
46     GradeBook myGradeBook; // create a GradeBook object named myGradeBook
47
48     // display initial value of courseName
49     cout << "Initial course name is: " << myGradeBook.getCourseName()
50         << endl;
51
```

Use *set* and *get* functions, even within the class

private members accessible only to member functions of the class

default constructor

Accessing **private** data outside class definition

© 2007 Pearson Ed -All rights reserved.

19.6 Data Members, *set* Functions and *get* Functions

```
52 // prompt for, input and set course name
53 cout << "\nPlease enter the course name:" << endl;
54 getline( cin, nameOfCourse ); // read a course name with blanks
55 myGradeBook.setCourseName( nameOfCourse ); // set the course name
56
57 cout << endl; // outputs a blank line
58 myGradeBook.displayMessage(); // display message with new course name
59 return 0; // indicate successful termination
60 } // end main
```

Modifying **private** data outside class definition

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!

© 2007 Pearson Ed -All rights reserved.

Software Engineering Observation 19.1

- As a rule of thumb, data members should be declared `private` and member functions should be declared `public`. (We will see that it is appropriate to declare certain member functions `private`, if they are to be accessed only by other member functions of the class.)

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

- Software engineering with *set* and *get* functions
 - public member functions that allow clients of a class to set or get the values of private data members.
 - *set* functions are sometimes called **mutators** and *get* functions are sometimes called **accessors**.
 - Allows the creator of the class to control how clients access private data.
 - Should also be used by other member functions of the same class.

© 2007 Pearson Ed -All rights reserved.

Initializing Objects with Constructors

- Constructors
 - Functions used to initialize an object's data when it is created.
 - The call is made **implicitly** by the compiler when the object is created.
 - Must be defined with the same name as the class.
 - Cannot return values.
 - Not even **void !!**
 - A **default constructor** has no parameters.
 - The compiler will provide one when a class does not explicitly include a constructor.
 - A compiler's default constructor only calls constructors of data members that are objects of classes.

© 2007 Pearson Ed -All rights reserved.

Constructor Example

```
1 // Fig. 19.7: fig19_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 using std::cout;
7 using std::endl;
8
9 #include <string> // program uses C++ standard string class
10 using std::string;
11
12 // GradeBook class definition
13 class GradeBook
14 {
15 public:
16     // constructor initializes courseName with string supplied as argument
17     GradeBook( string name )
18     {
19         setCourseName( name ); // call set function to initialize courseName
20     } // end GradeBook constructor
21
22     // function to set the course name
23     void setCourseName( string name )
24     {
25         courseName = name; // store the course name in the object
26     } // end function setCourseName
27
```

Constructor has same name as class and no return type

Initialize data member

© 2007 Pearson Ed -All rights reserved.

Constructor Example

```
28 // function to get the course name
29 string getCourseName()
30 {
31     return courseName; // return object's courseName
32 } // end function getCourseName
33
34 // display a welcome message to the GradeBook user
35 void displayMessage()
36 {
37     // call getCourseName to get the courseName
38     cout << "Welcome to the grade book for\n" << getCourseName()
39         << "!" << endl;
40 } // end function displayMessage
41 private:
42     string courseName; // course name for this GradeBook
43 }; // end class GradeBook
44
```

© 2007 Pearson Ed -All rights reserved.

Constructor Example

```
45 // function main begins program execution
46 int main()
47 {
48     // create two GradeBook objects
49     GradeBook gradeBook1( "CS101 Introduction to C++ Programming" );
50     GradeBook gradeBook2( "CS102 Data Structures in C++" );
51
52     // display initial value of courseName for each GradeBook
53     cout << "gradeBook1 created for course: " << gradeBook1.getCourseName()
54         << "\ngradeBook2 created for course: " << gradeBook2.getCourseName()
55         << endl;
56     return 0; // indicate successful termination
57 } // end main
```

Creating objects implicitly calls the constructor

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

© 2007 Pearson Ed -All rights reserved.

Placing a Class in a Separate File for Reusability

- `.cpp` file is known as a source-code file.
- Header files
 - Separate files in which class definitions are placed.
 - Allow compiler to recognize the classes when used elsewhere.
 - Generally have `.h` filename extensions
- Driver files
 - A program used to test software (such as classes).
 - Contains a `main` function so it can be executed.

© 2007 Pearson Ed -All rights reserved.

19.9 Separating Interface from Implementation

- **Interface**
 - Describes what services a class's clients can use and how to request those services.
 - without revealing how the class carries out the services.
 - A class definition that lists only member function names, return types and parameter types
 - Function prototypes
 - A class's interface consists of the class's **public** member functions (services).
- **Separating interface from implementation**
 - Client code should not break if implementation changes, as long as the interface stays the same.

© 2007 Pearson Ed -All rights reserved.

Separating Interface from Implementation

- Define the member functions outside the class definition, in a separate source-code file.
 - In a source-code file for a class
 - Use **binary scope resolution operator (::)** to tie each member function to the class definition.
 - Implementation details are hidden.
 - Client code does not need to know the implementation.
- In a header file for a class
 - The function prototypes describe the class's public interface.

© 2007 Pearson Ed -All rights reserved.

Separating Interface from Implementation

```
1 // Fig. 19.11: 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 std::string;
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
```

Interface contains data members
and member function prototypes



© 2007 Pearson Ed -All rights reserved.

Separating Interface from Implementation

```
1 // Fig. 19.12: GradeBook.cpp
2 // GradeBook member-function definitions. This file contains
3 // implementations of the member functions prototyped in GradeBook.h.
4 #include <iostream>
5 using std::cout;
6 using std::endl;
7
8 #include "GradeBook.h" // include definition of class GradeBook
9
10 // constructor initializes courseName with string supplied as argument
11 GradeBook::GradeBook( string name )
12 {
13     setCourseName( name ); // call set function to initialize courseName
14 } // end GradeBook constructor
15
16 // function to set the course name
17 void GradeBook::setCourseName( string name )
18 {
19     courseName = name; // store the course name in the object
20 } // end function setCourseName
21
```

GradeBook implementation is placed in a separate source-code file

Include the header file to access the class name **GradeBook**

Binary scope resolution operator ties a function to its class

© 2007 Pearson Ed -All rights reserved.

Separating Interface from Implementation

```
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
```

© 2007 Pearson Ed -All rights reserved.

Separating Interface from Implementation

```
1 // Fig. 19.13: flg19_13.cpp
2 // GradeBook class demonstration after separating
3 // its interface from its implementation.
4 #include <iostream>
5 using std::cout;
6 using std::endl;
7
8 #include "GradeBook.h" // include definition of class GradeBook
9
10 // function main begins program execution
11 int main()
12 {
13     // create two GradeBook objects
14     GradeBook gradeBook1( "CS101 Introduction to C++ Programming" );
15     GradeBook gradeBook2( "CS102 Data Structures in C++" );
16
17     // display initial value of courseName for each GradeBook
18     cout << "gradeBook1 created for course: " << gradeBook1.getCourseName()
19         << "\ngradeBook2 created for course: " << gradeBook2.getCourseName()
20         << endl;
21     return 0; // indicate successful termination
22 } // end main
```

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

© 2007 Pearson Ed -All rights reserved.

19.10 Validating Data with *set* Functions

- *set* functions can validate data.
 - Known as validity checking.
 - Keeps object in a consistent state.
 - The data member contains a valid value.
 - Can return values indicating that attempts were made to assign invalid data.
- *string* member functions
 - *length* returns the number of characters in the *string*.
 - *Substr* returns specified substring within the *string*.

© 2007 Pearson Ed -All rights reserved.

Validating Data with *set* Functions

Outline

```
1 // Fig. 19.16: GradeBook.cpp
2 // Implementations of the GradeBook member-function definitions.
3 // The setCourseName function performs validation.
4 #include <iostream>
5 using std::cout;
6 using std::endl;
7
8 #include "GradeBook.h" // include definition of class GradeBook
9
10 // constructor initializes courseName with string supplied as argument
11 GradeBook::GradeBook( string name )
12 {
13     setCourseName( name ); // validate and store courseName
14 } // end GradeBook constructor
15
16 // function that sets the course name;
17 // ensures that the course name has at most 25 characters
18 void GradeBook::setCourseName( string name )
19 {
20     if ( name.length() <= 25 ) // if name has 25 or fewer characters
21         courseName = name; // store the course name in the object
22
```

Constructor calls *set* function to perform validity checking

set functions perform validity checking to keep **courseName** in a consistent state

© 2007 Pearson Ed -All rights reserved.

Validating Data with *set* Functions

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

© 2007 Pearson Ed -All rights reserved.

Summary

- Introduced class definitions and objects
 - Public versus private access into class.
- Syntax for member functions
- Syntax data members
 - Get and Set functions
 - Constructors
- Placing classes in separate files
- Separating interface from implementation
- Data validation in set functions.