



## Initializing Objects with Constructors

---

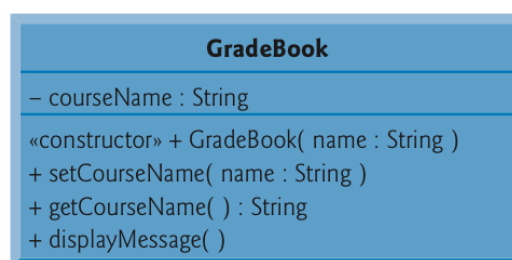
### Initializing Objects with Constructors

- A constructor is used to initialize an object of the class when the object is created.
- A constructor is a special member function that must be defined with the same name as the class, so that the compiler can distinguish it from the class's other member functions.
- An important difference between constructors and other functions is that constructors cannot return values, so they cannot specify a return type (not even void). Normally, constructors are declared public.
- C++ requires a constructor call for each object that's created, which helps ensure that each object is initialized properly before it's used in a program.
- C++ requires a constructor call for each object that's created, which helps ensure that each object is initialized properly before it's used in a program.
- If a class does not explicitly include a constructor, the compiler provides a default constructor—that is, a constructor with no parameters.

Unless no initialization of your class's data members is necessary (almost never), provide a constructor to ensure that your class's data members are initialized with meaningful values when each new object of your class is created.

### Adding the Constructor to Class GradeBook's UML Class Diagram

- Like operations, the UML models constructors in the third compartment of a class in a class diagram.
- To distinguish a constructor from a class's operations, the UML places the word "constructor" between guillemets (« and ») before the constructor's name. By convention, you list the class's constructor before other operations in the third compartment.



```

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
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
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++

```