

# Object Oriented Programming

Shaobai Kan

*chapter 9*



# Outline

# Outline

- Initializing objects with constructors

# Outline

- Initializing objects with constructors
- Placing a class in a separate file for reusability

# Outline

- Initializing objects with constructors
- Placing a class in a separate file for reusability
- Exercise

# *Initializing objects with constructors*

# Object Oriented Programming

**Definition.** A **constructor** is a special member function that must be defined with the same name as the class.

—— Constructors can not return values, so they can not specify a return type (not even void).

# Object Oriented Programming

**Definition.** A **constructor** is a special member function that must be defined with the same name as the class.

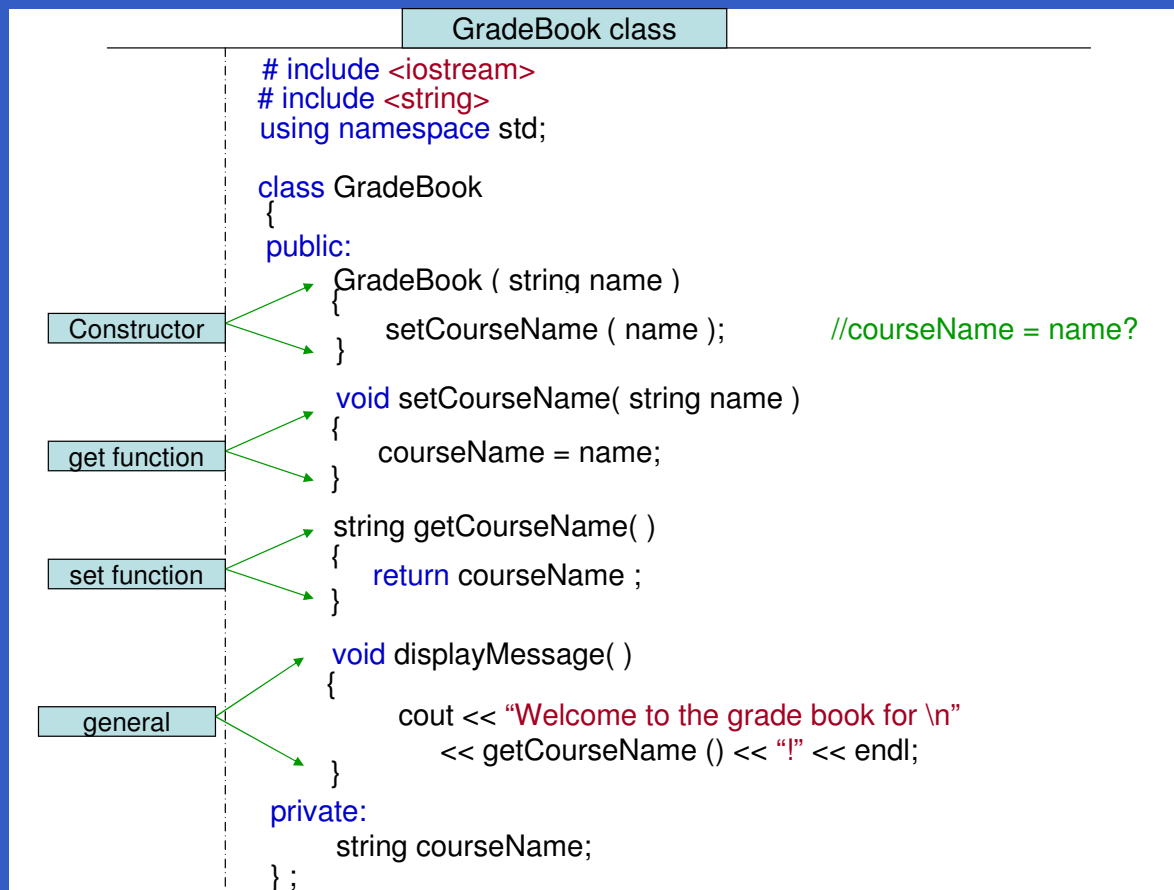
—— Constructors can not return values, so they can not specify a return type (not even void).

## **Fact.**

- C++ requires a constructor call (implicitly) for each object is created.
- If a class does not explicitly include a constructor, the compiler provides a **default constructor** (with no parameters).



# Example: class GradeBook



# Example: class GradeBook

## main function

```
int main ( )
{
    GradeBook gradeBook1( "CS101 Introduction to C++ Programming" );
    GradeBook gradeBook2( "CS102 Data Structures in C++" );

    cout << "gradebook1 created for course: "
          << gradeBook1.getCourseName( );

    cout << endl;

    cout << "gradebook2 created for course: "
          << gradeBook2.getCourseName( );

    return 0;
}
```

# Output: class GradeBook

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

# Default constructor

**Definition.** Any constructor that takes no arguments is called a **default constructor**.

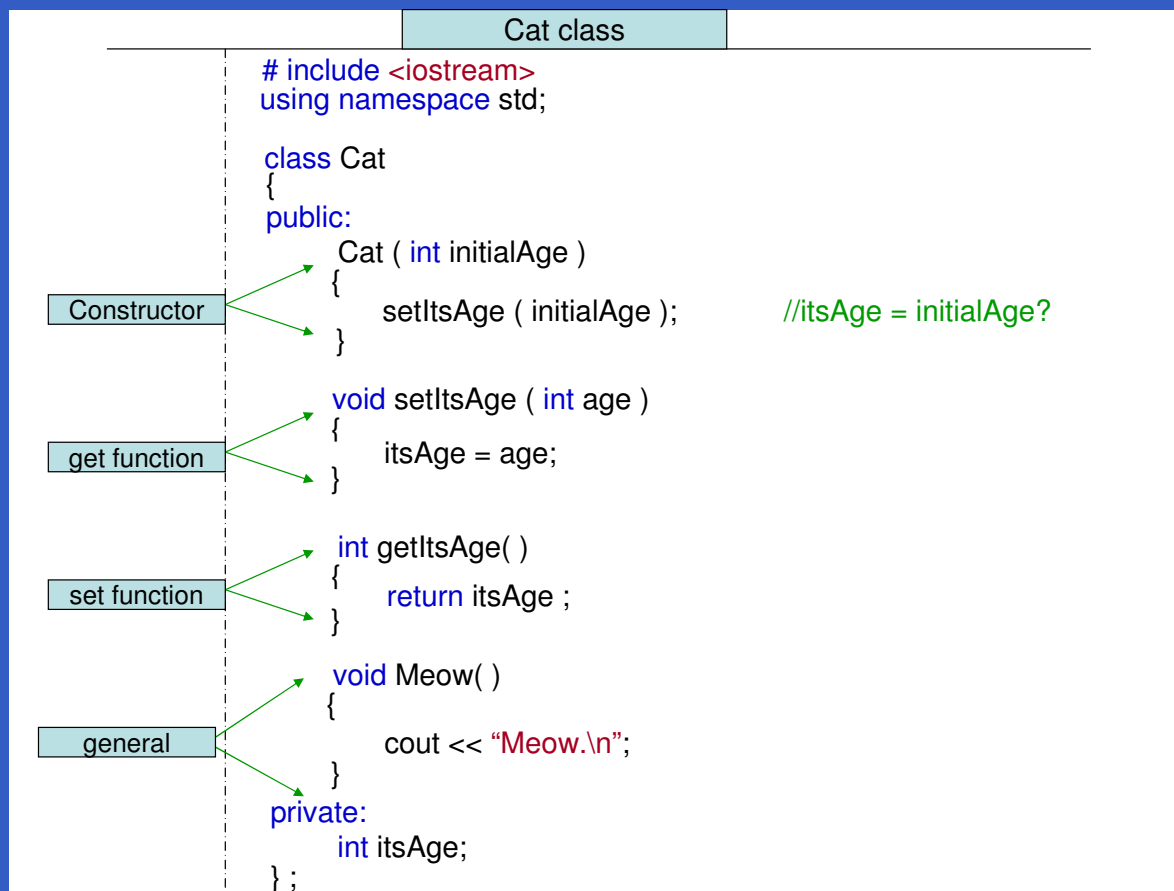
# Default constructor

**Definition.** Any constructor that takes no arguments is called a **default constructor**.

**Fact.** There are two ways to provide a **default constructor**

- The compiler implicitly creates a default constructor in a class that does not define a constructor.
- We explicitly define a constructor that takes no argument.

# Example: class Cat



# Output: class Cat

## main function

```
int main ( )
{
    Cat Frisky ( 5 );
    Frisky.Meow ( );

    cout << "Frisky is a cat who is ";
    cout << Frisky.getItsAge( ) << " years old.\n" ;
    Frisky.Meow ( );
    Frisky.setItsAge ( 5 ) ;

    cout << "Now Frisky is ";
    cout << Frisky.getItsAge( ) << " years old.\n" ;
    return 0;
}
```

# ***Separating** Interface **from** Implementation*



# Separating file for reusability

**Fact.** One of the benefits of creating class definitions is that, *when packaged properly*, our classes can be reused by programmers - potentially worldwide.

# Separating file for reusability

**Fact.** One of the benefits of creating class definitions is that, *when packaged properly*, our classes can be reused by programmers - potentially worldwide.

Example. We can reuse C++ Standard Library type string in any C++ program just by including the header file `<string>`.

# Example: class GradeBook II

GradeBook.h file

```
#include <iostream>
#include <string>
using namespace std;

class GradeBook
{
public:
    GradeBook ( string name )
    {
        setCourseName ( name );
    }

    void setCourseName( string name )
    {
        courseName = name;
    }

    string getCourseName( )
    {
        return courseName ;
    }

    void displayMessage( )
    {
        cout << "Welcome to the grade book for \n"
              << getCourseName () << "!" << endl;
    }

private:
    string courseName;
};
```

# Example: class GradeBook II

## .cpp Test Program

```
#include <iostream>
#include "GradeBook.h"    //why not " #include <GradeBook.h>" ?
using namespace std;

int main ( )
{
    GradeBook gradeBook1( "CS101 Introduction to C++ Programming" );
    GradeBook gradeBook2( "CS102 Data Structures in C++" );

    cout << "gradebook1 created for course: "
          << gradeBook1.getCourseName( );

    cout << endl;

    cout << "gradebook2 created for course: "
          << gradeBook2.getCourseName( );

    return 0;
}
```

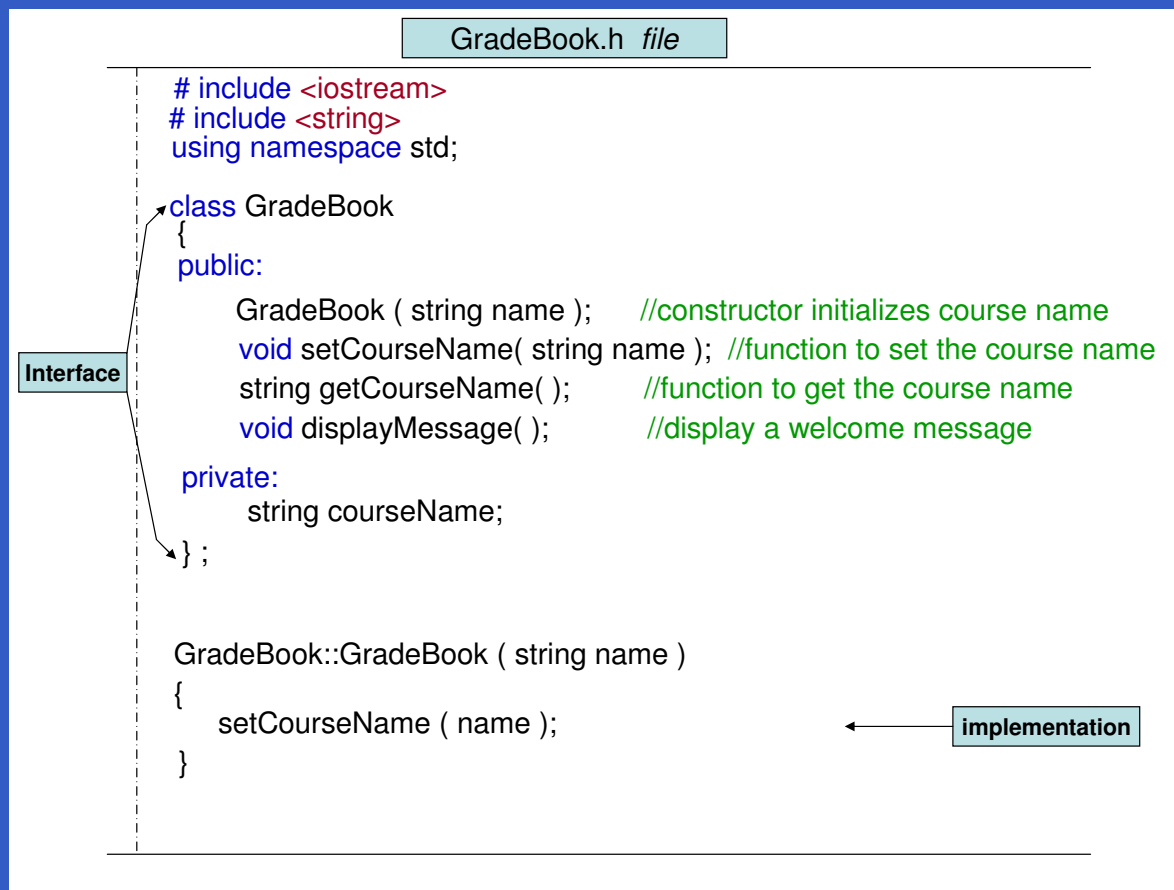
# How header files are located?

- When the preprocessor encounters a header file name in **quotes** (" "), it attempts to locate the header file in the same directory as the file in which the **#include** directive appears. If the preprocessor cannot find the header file in that directory, it searches for it in the same location(s) as the C++ Standard Library header files.

# How header files are located?

- When the preprocessor encounters a header file name in **quotes** (" "), it attempts to locate the header file in the same directory as the file in which the **#include** directive appears. If the preprocessor cannot find the header file in that directory, it searches for it in the same location(s) as the C++ Standard Library header files.
- When the preprocessor encounters a header file name in **angle brackets**(< >), it assumes that the header file is part of the C++ Standard Library and does not look in the directory of the program that is being processed.

# Example: class GradeBook III



# Example: class GradeBook III

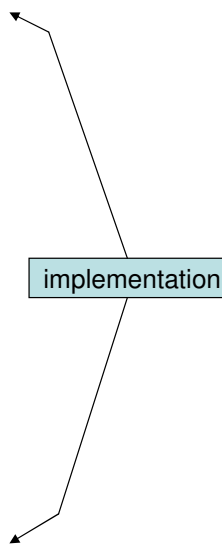
GradeBook.h file

```
void GradeBook::setCourseName( string name )  
{  
    courseName = name;  
}
```

```
string GradeBook::getCourseName( )  
{  
    return courseName ;  
}
```

```
void GradeBook::displayMessage( )  
{  
    cout << "Welcome to the grade book for \n"  
    << getCourseName ( ) << "!" << endl;  
}
```

implementation





# *Exercise*

# Exercise

**Exercise 1.** Create a class called *Employee* that includes three pieces of information as data members — a first name (type *string*), a last name (type *string*) and monthly salary (type *int*). Your class should have a constructor that initializes the three data members. Provide a *set* function and *get* function for each data member. If the monthly salary is not positive, set it to 0. Write a test program that demonstrates class *Employee*'s capabilities. Create two *Employee* objects and display each object's *yearly* salary. Then give each *Employee* a 10 percent raise and display each *Employee*'s yearly salary again.

## ***Homework:***

- Read Sec. 9.5 - 9.6
- Exercise 1 in this slide