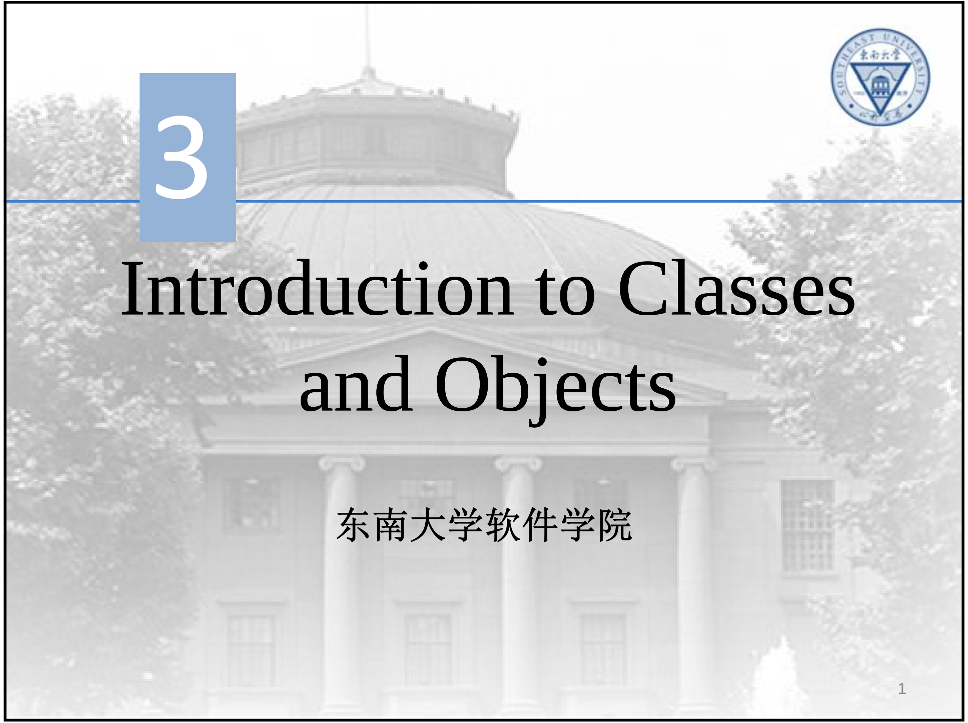


3 Introduction to Classes and Objects

东南大学软件学院

- 计算矩形、圆形、三角形的面积

```
double Cal_area_circle(double r)
{
  return 3.1416 * r * r;
}

double  Cal_Area_rect(double a, double b)
{
  return a*b;
}

double Cal_Area_tri(double a, double b, double c)
{ //海伦公式
  double s = 0.5*(a+b+c);
  double a = sqrt(s*(s-a)*(s-b)*(s-c));
  return a;
}
```

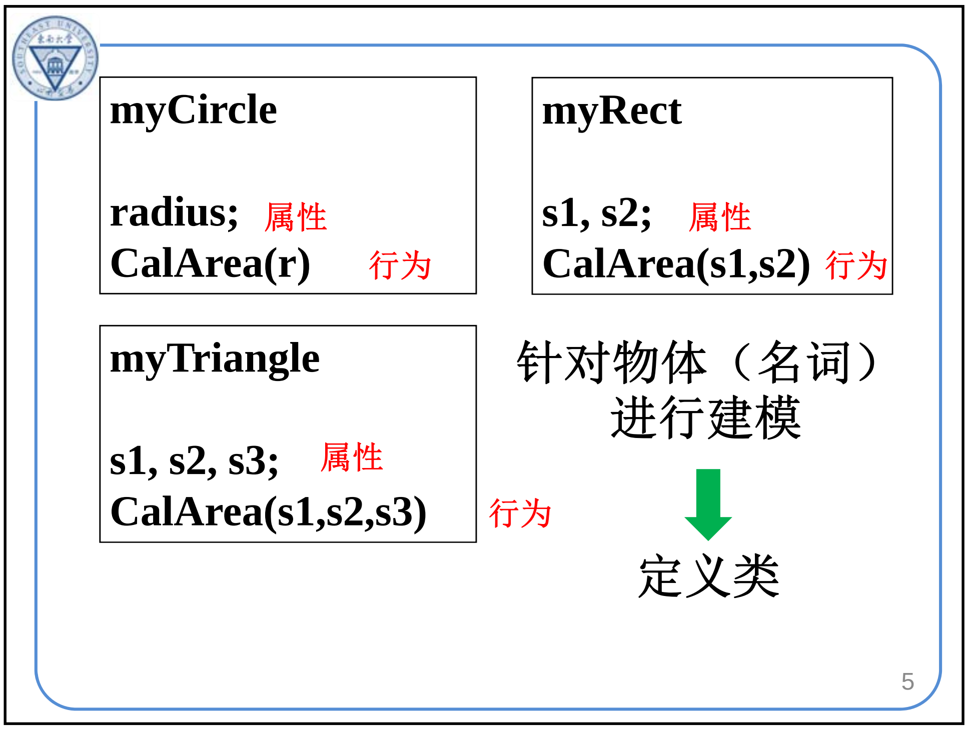
```
int main()
{
  double circle_r=3.0;
  double rect_s1 = 4.0, rect_s2 = 3.4; //sides of rectangular
  double tri_s1 = 2.0, tri_s2 = 3.0, tri_s3 = 4.0;//sides of tri
  double area_circle, area_rect, area_tri;
  area_circle = Cal_Area_circle(circle_r);
  area_rect = Cal_Area_rect(rect_s1, rect_s2);
  area_tri = Cal_Area_tri(tri_s1, tri_s2, tri_s3);
  return 0;
}
```

3

- 定义的函数针对动作的（面向过程的编程）
- 函数实参使用的是否正确很大程度上依赖程序员的细心

```
int main()
{
  double circle_r=3.0;
  double rect_s1 = 4.0, rect_s2 = 3.4; //sides of rectangular
  double tri_s1 = 2.0, tri_s2 = 3.0, tri_s3 = 4.0;//sides of tri
  double area_circle, area_rect, area_tri;
  area_circle = Cal_Area_circle(circle_r);
  area_rect = Cal_Area_rect(rect_s1, rect_s2);
  area_tri = Cal_Area_tri(tri_s1, tri_s2, tri_s3);

  double tri2_s1 = 3, tri2_s2 = 4, tri2_s3 = 5;
  double area_tri2;
  area_tri2 = Cal_Area_tri(tri2_s1, tri2_s2, tri2_s3);

  return 0;
}
```

4



myCircle

radius; 属性
CalArea(r) 行为

myRect

s1, s2; 属性
CalArea(s1,s2) 行为

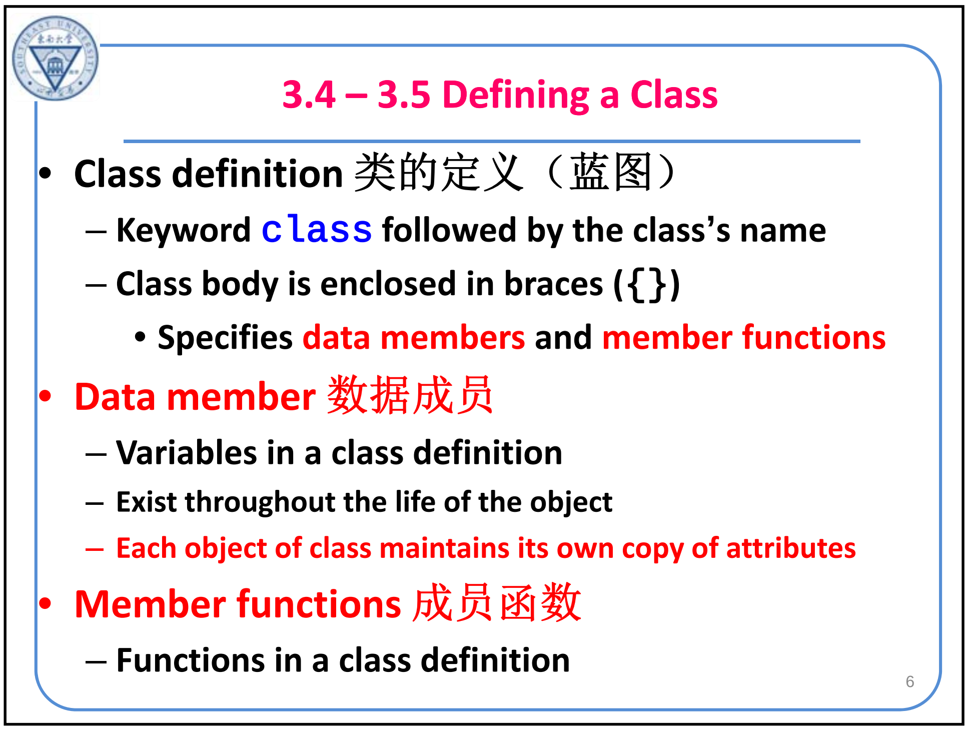
myTriangle

s1, s2, s3; 属性
CalArea(s1,s2,s3) 行为

针对物体（名词）
进行建模

定义类

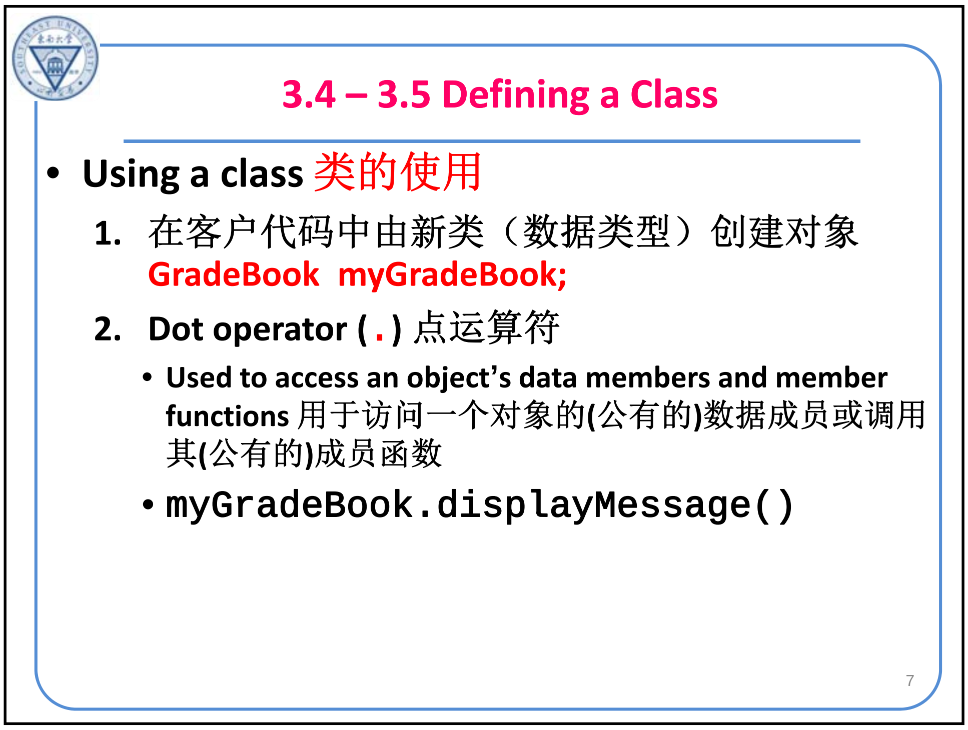
5



3.4 – 3.5 Defining a Class

- Class definition 类的定义（蓝图）
  - Keyword `class` followed by the class's name
  - Class body is enclosed in braces ({})
    - Specifies data members and member functions
- Data member 数据成员
  - Variables in a class definition
  - Exist throughout the life of the object
  - Each object of class maintains its own copy of attributes
- Member functions 成员函数
  - Functions in a class definition

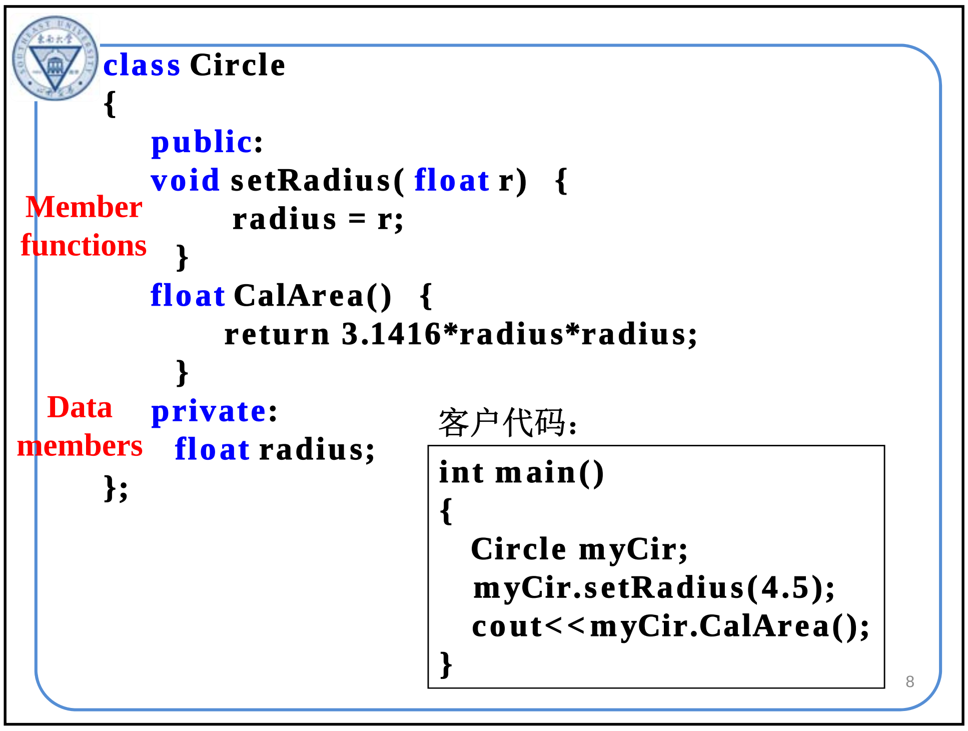
6



3.4 – 3.5 Defining a Class

- Using a class 类的使用
  1. 在客户代码中由新类（数据类型）创建对象
     GradeBook myGradeBook;
  2. Dot operator (.) 点运算符
     - Used to access an object's data members and member functions 用于访问一个对象的(公有的)数据成员或调用其(公有的)成员函数
     - `myGradeBook.displayMessage()`

7



```
class Circle
{
    public:
    void setRadius( float r)  {
            radius = r;
     }
    float CalArea()  {
          return 3.1416*radius*radius;
     }
    private:
      float radius;
};
```

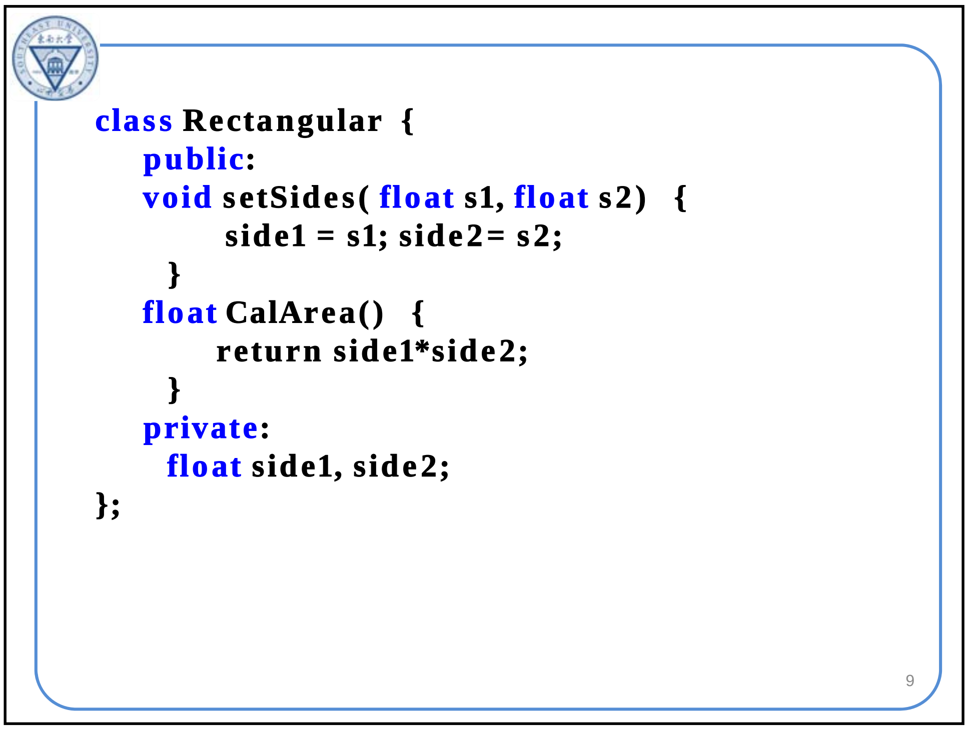
Member functions

Data members

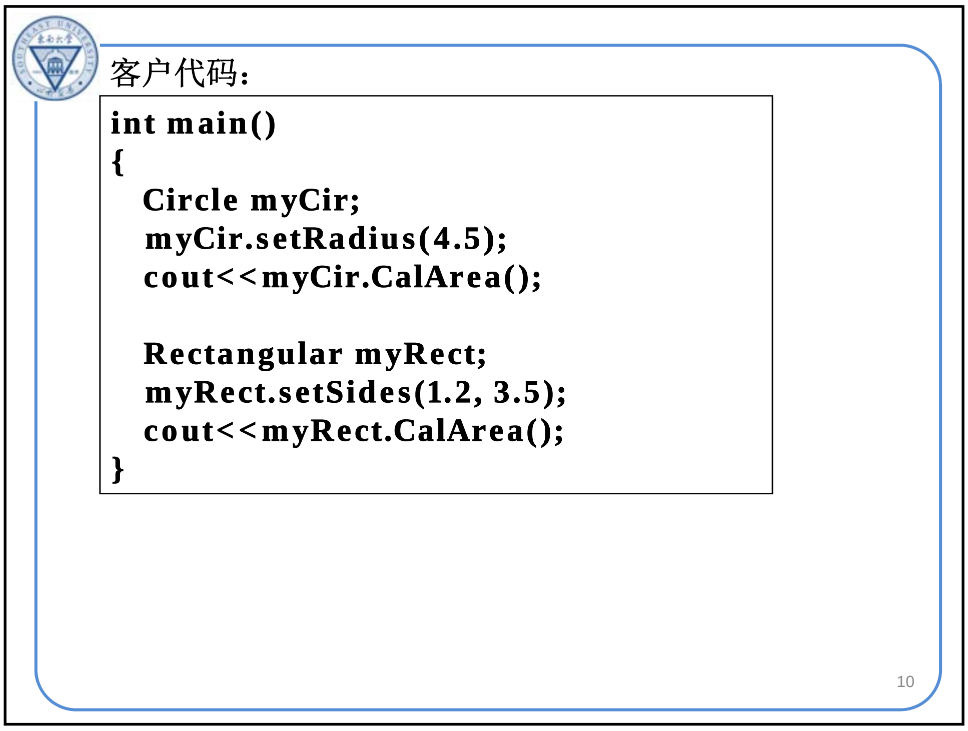
客户代码:

```
int main()
{
   Circle myCir;
   myCir.setRadius(4.5);
   cout<<myCir.CalArea();
}
```

8



```
class Rectangular  {
    public:
    void setSides( float s1, float s2)   {
           side1 = s1; side2= s2;
      }
    float CalArea()   {
          return side1*side2;
      }
    private:
      float side1, side2;
};
```



客户代码:

```
int main()
{
   Circle myCir;
   myCir.setRadius(4.5);
   cout<<myCir.CalArea();

   Rectangular myRect;
   myRect.setSides(1.2, 3.5);
   cout<<myRect.CalArea();
}
```

客户代码：

```
int main()
{
   Circle myCir;
   myCir.setRadius(4.5);
   cout<<myCir.CalArea();

   Rectangular myRect;
   myRect.setSides(1.0, 3.0);
   cout<<myRect.CalArea();

   Rectangular myRect2;
   myRect2.setSides(4.0, 7.5);
   cout<<myRect2.CalArea();

}
```

Each object of class maintains its own copy of attributes
类的每个对象都对数据成员保留一份自己的拷贝。

myRect 和myRect2具有不同的边长值，但都是Rectangular类的对象。

Class GradeBook

属性：

-课程名称

-学生名单

-学生学号

-成绩

行为：

-输出（打印）成绩单

-计算平均分

-统计成绩分布情况

-按分数高低排序

利用GradeBook类生成不同的成绩册（对象）

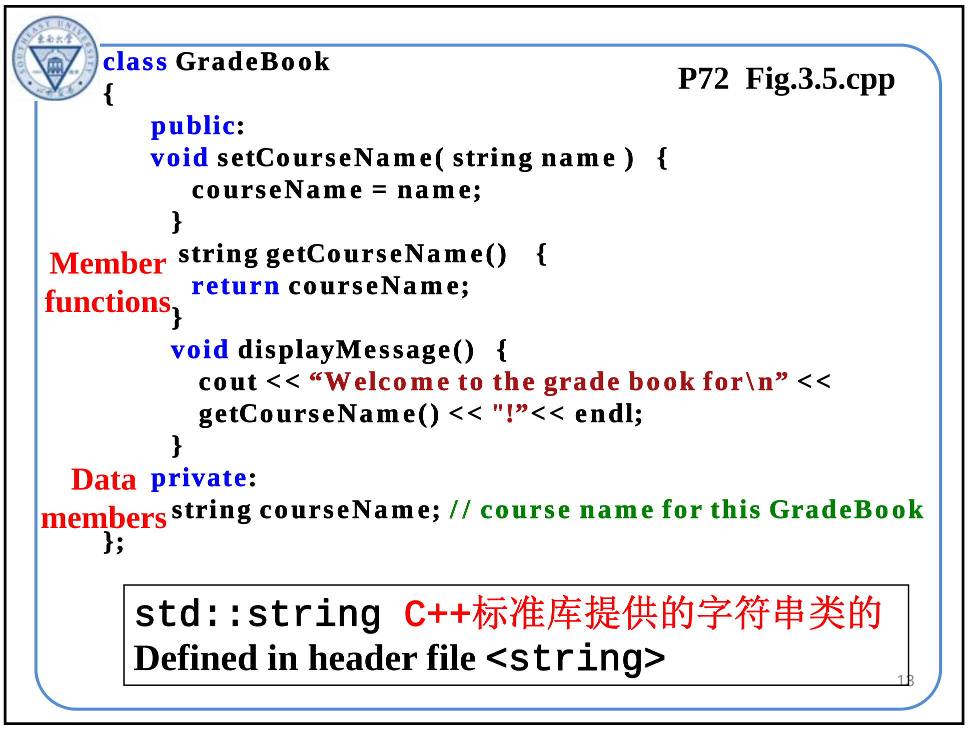
高等物理

高等数学

程序设计

数据结构

……



P72 Fig.3.5.cpp

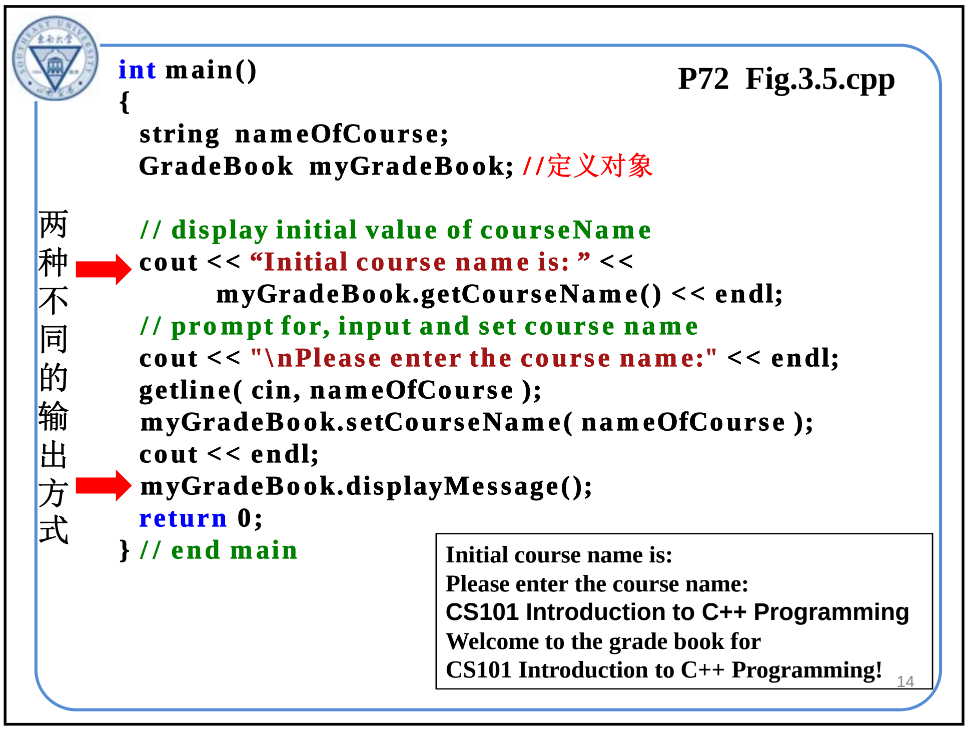
Member functions

Data members

```
class GradeBook
{
    public:
    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; // course name for this GradeBook
};
```

std::string C++标准库提供的字符串类的
Defined in header file <string>

13



P72 Fig.3.5.cpp

两种不同的输出方式

```
int main()
{
   string  nameOfCourse;
   GradeBook  myGradeBook; //定义对象

   // display initial value of courseName
   cout << "Initial course name is: " <<
          myGradeBook.getCourseName() << endl;
   // prompt for, input and set course name
   cout << "\nPlease enter the course name:" << endl;
   getline( cin, nameOfCourse );
   myGradeBook.setCourseName( nameOfCourse );
   cout << endl;
   myGradeBook.displayMessage();
   return 0;
} // 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!
```

14

3.4 – 3.5 Defining a Class

- 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`
      - 可以读入白字符
    - `cin << nameOfCourse`
      - 不可以读入白字符

15

```
class GradeBook
{
    public:
    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; // course name for this GradeBook
};
```

成员函数

数据成员

16

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

- Access-specifier `public`:访问限定符
  - Indicates that a member function or data member is accessible to other functions and member functions of other classes可以被其他函数或其他类的成员函数访问
- Access-specifier `private`访问限定符
  - Makes a data member or member function accessible only to member functions of the class 仅能被当前类的成员函数访问
  - `private` is the default access for class members
  - Data hiding 数据隐藏

17

客户代码

P72 Fig.3.5.cpp

```
int main()
{
  string  nameOfCourse;
  GradeBook  myGradeBook;

  // display initial value of courseName
  cout << “Initial course name is: ” <<
       myGradeBook.getCourseName()<< endl;
  // prompt for, input and set course name
  cout << "\nPlease enter the course name:" << endl;
  getline( cin, nameOfCourse );
  myGradeBook.setCourseName( nameOfCourse );
  cout << endl;
  myGradeBook.displayMessage();
  return 0;
} // end main
```

不能使用
myGradeBook.courseName

18

Software Engineering Observation 3.1

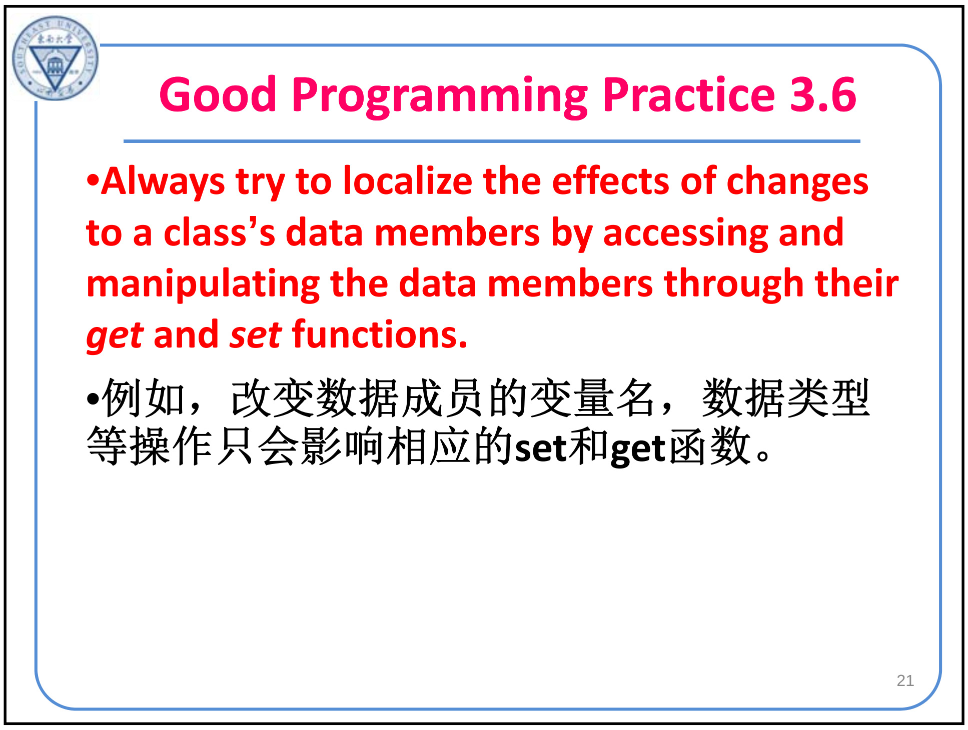
•As a rule, data members should be declared `private` and member functions should be declared `public`. (We will see that it is appropriate to declare certain member functions `private`, if they are to be accessed only by other member functions of the class.)

•数据成员的访问控制声明成私有成员，

•成员函数的访问控制声明成公有成员。

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

- 如何设置和获取私有的数据成员的值呢？
  - 为数据成员添加一对公有的**set** 和 **get** 函数
  - `public` member functions that allow clients of a class to set or get the values of `private` data members
  - Should also be used by other member functions of the same class 当前类的其他成员函数也应当使用
  - set/get函数名称是约定俗成的定义方式，不是C++标准

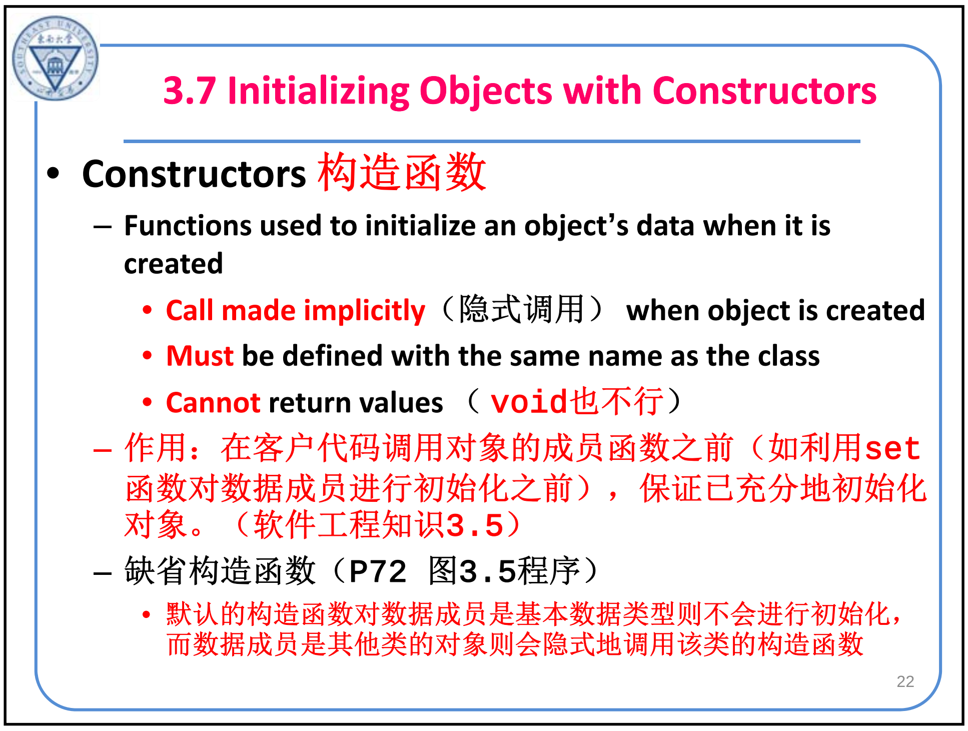


Good Programming Practice 3.6

•**Always try to localize the effects of changes to a class's data members by accessing and manipulating the data members through their *get* and *set* functions.**

•例如，改变数据成员的变量名，数据类型等操作只会影响相应的**set**和**get**函数。

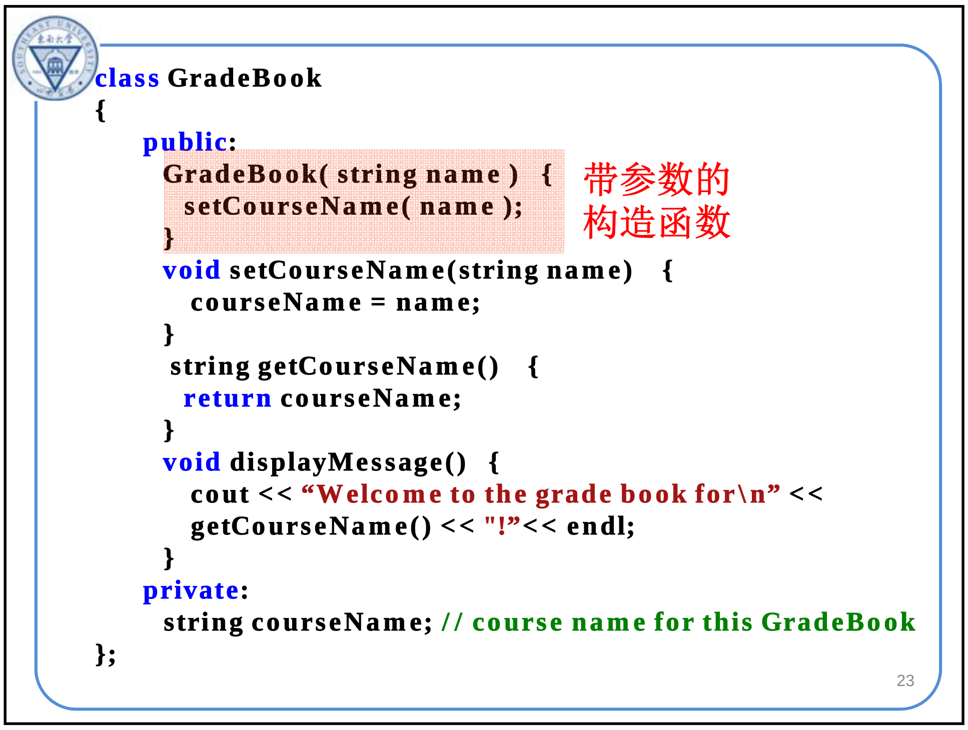
21



3.7 Initializing Objects with Constructors

- **Constructors** 构造函数
  - **Functions used to initialize an object's data when it is created**
    - **Call made implicitly**（隐式调用） **when object is created**
    - **Must be defined with the same name as the class**
    - **Cannot return values** （ void也不行）
  - 作用：在客户代码调用对象的成员函数之前（如利用set函数对数据成员进行初始化之前），保证已充分地初始化对象。（软件工程知识3.5）
  - 缺省构造函数（P72 图3.5程序）
    - 默认的构造函数对数据成员是基本数据类型则不会进行初始化，而数据成员是其他类的对象则会隐式地调用该类的构造函数

22



```
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; // course name for this GradeBook
};
```

带参数的构造函数

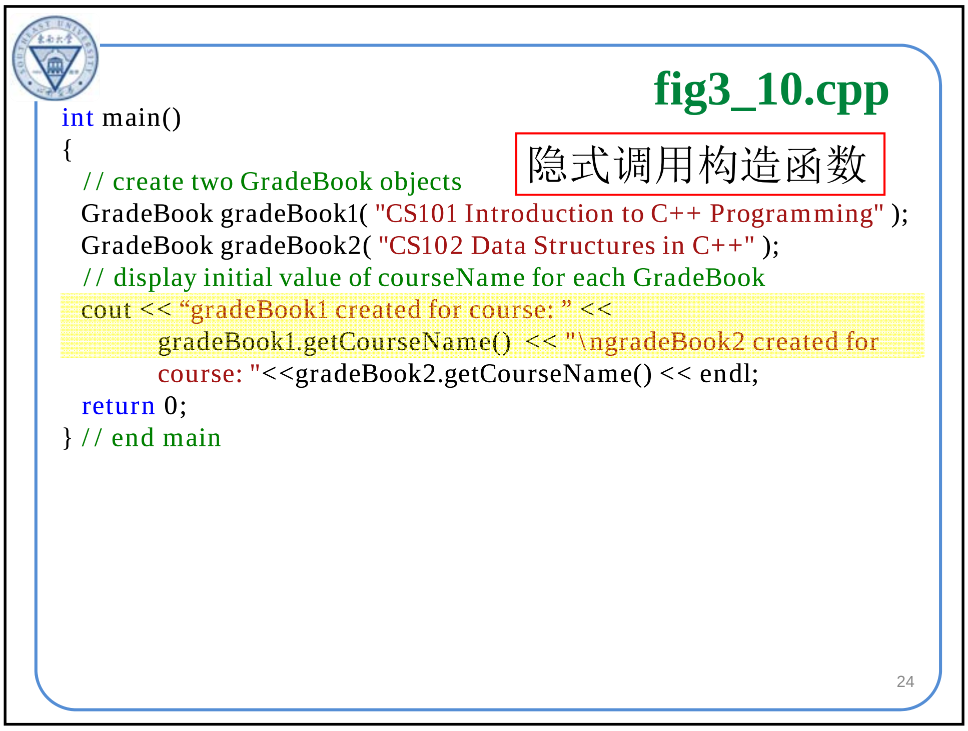
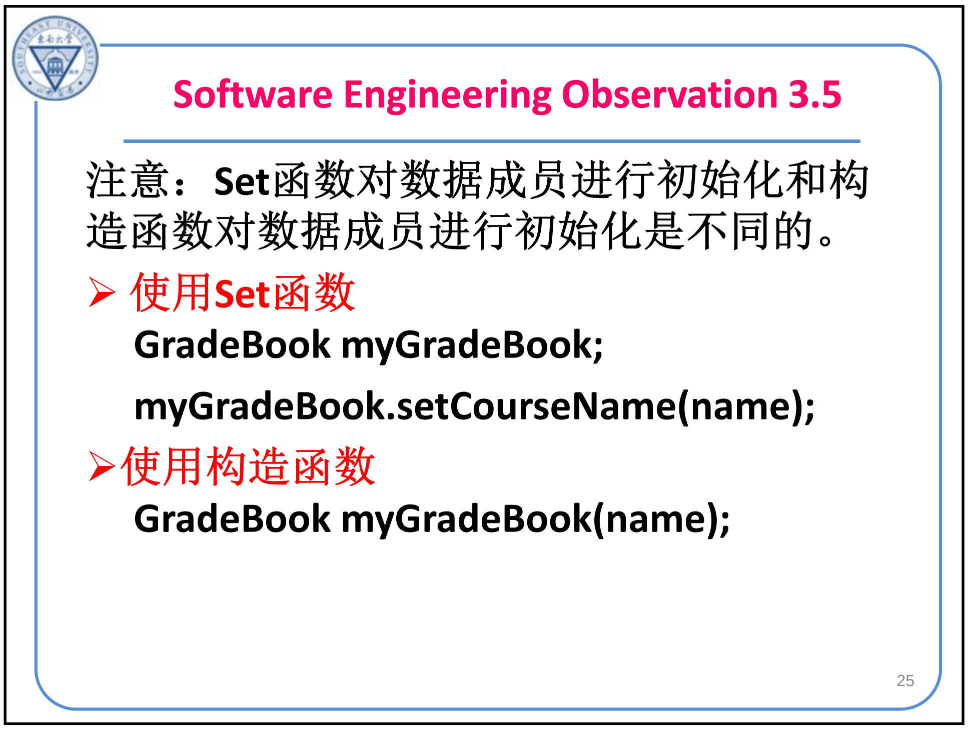


fig3_10.cpp

隐式调用构造函数

```
int main()
{
  // create two GradeBook objects
  GradeBook gradeBook1( "CS101 Introduction to C++ Programming" );
  GradeBook gradeBook2( "CS102 Data Structures in C++" );
  // display initial value of courseName for each GradeBook
  cout << “gradeBook1 created for course: ” <<
         gradeBook1.getCourseName()  << "\ngradeBook2 created for
         course: "<<gradeBook2.getCourseName() << endl;
  return 0;
} // end main
```



Software Engineering Observation 3.5

注意：**Set**函数对数据成员进行初始化和构造函数对数据成员进行初始化是不同的。

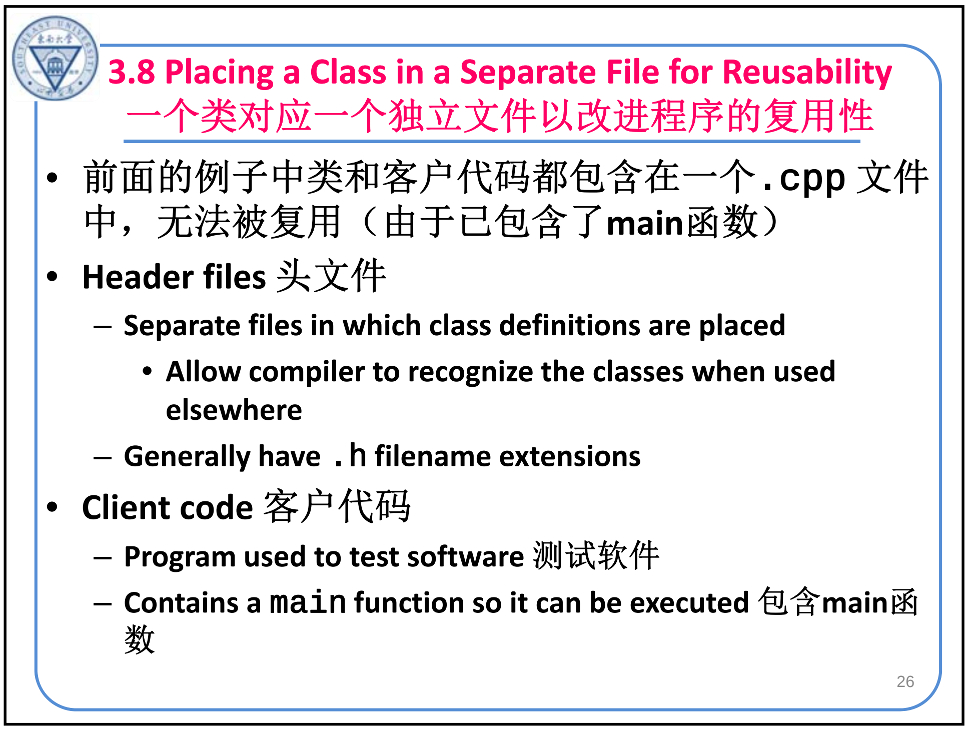
- 使用**Set**函数

```
GradeBook myGradeBook;
myGradeBook.setCourseName(name);
```

- 使用构造函数

```
GradeBook myGradeBook(name);
```

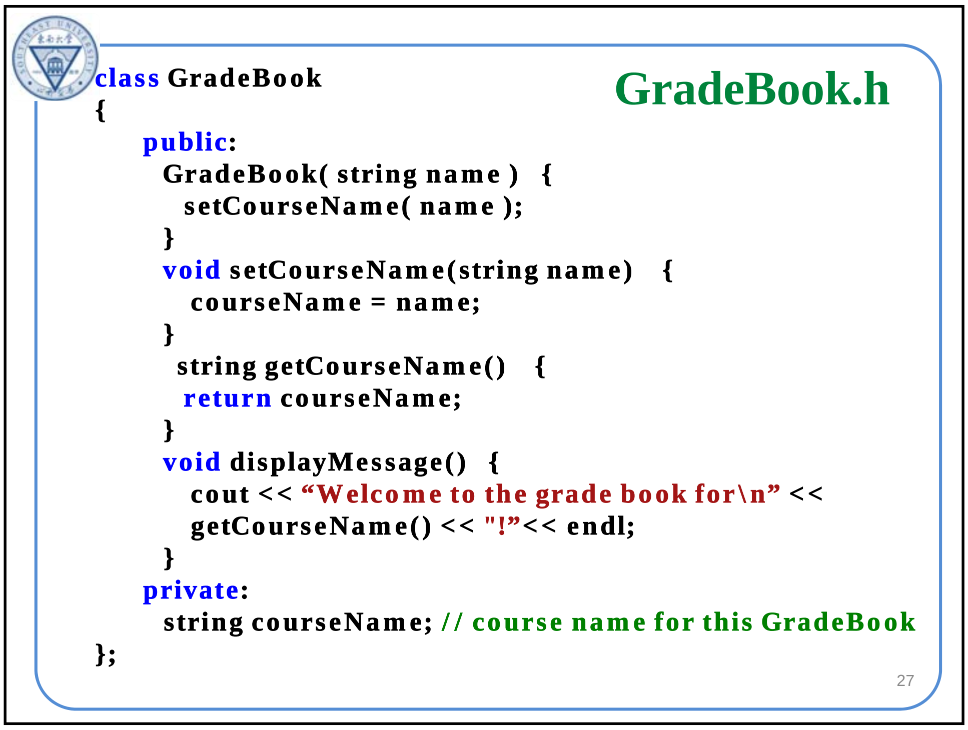
25



3.8 Placing a Class in a Separate File for Reusability 一个类对应一个独立文件以改进程序的复用性

- 前面的例子中类和客户代码都包含在一个`.cpp` 文件中，无法被复用（由于已包含了**main**函数）
- **Header files** 头文件
  - **Separate files in which class definitions are placed**
    - **Allow compiler to recognize the classes when used elsewhere**
  - **Generally have** `.h` **filename extensions**
- **Client code** 客户代码
  - **Program used to test software** 测试软件
  - **Contains a** `main` **function so it can be executed** 包含**main**函数

26



GradeBook.h

```
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; // course name for this GradeBook
};
```

27

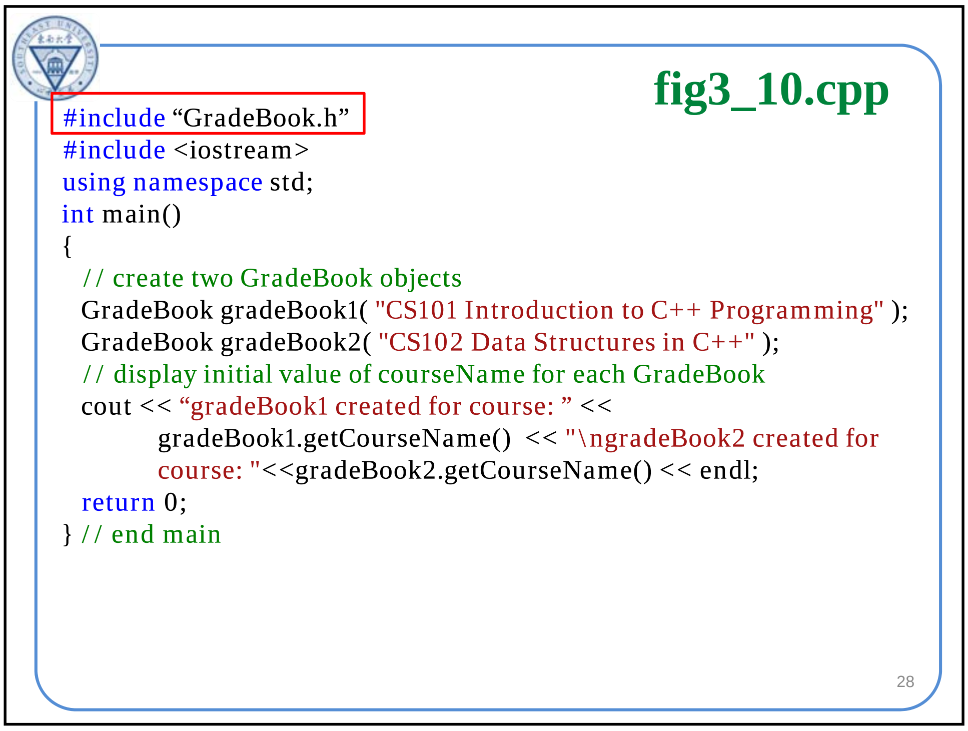
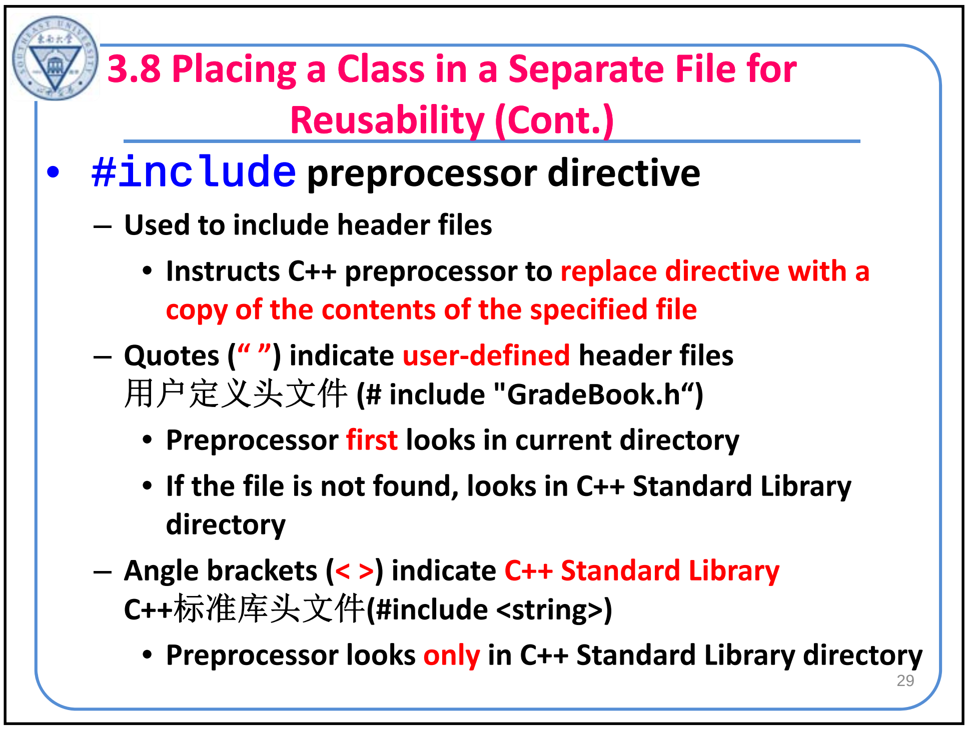


fig3_10.cpp

```
#include “GradeBook.h”
#include <iostream>
using namespace std;
int main()
{
  // create two GradeBook objects
  GradeBook gradeBook1( "CS101 Introduction to C++ Programming" );
  GradeBook gradeBook2( "CS102 Data Structures in C++" );
  // display initial value of courseName for each GradeBook
  cout << “gradeBook1 created for course: ” <<
          gradeBook1.getCourseName()  << "\ngradeBook2 created for
          course: "<<gradeBook2.getCourseName() << endl;
  return 0;
} // end main
```

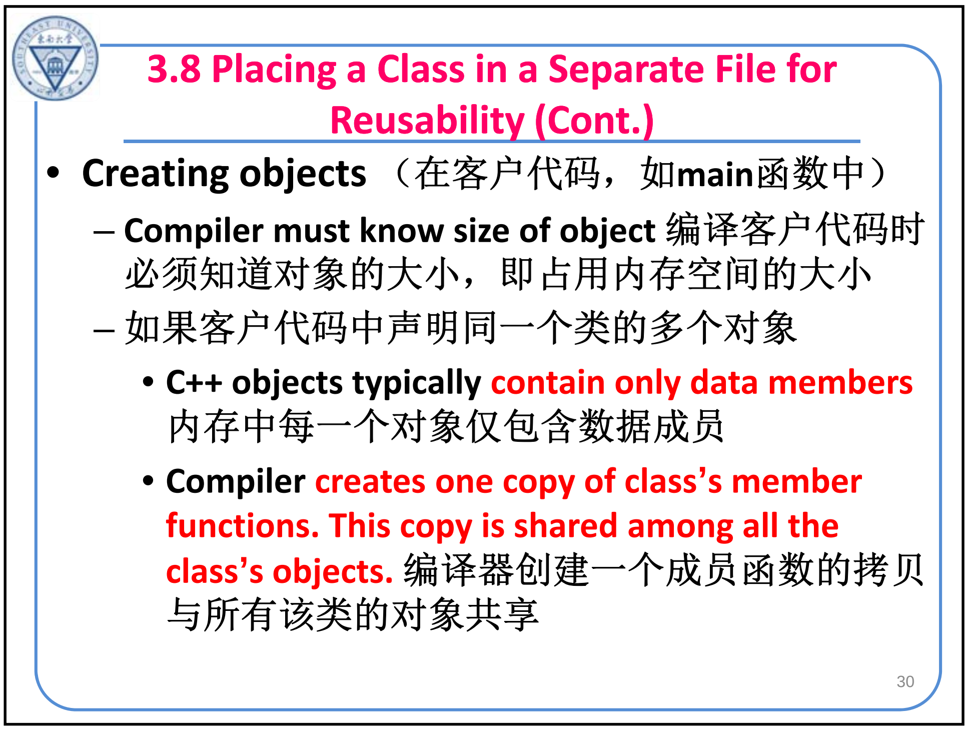
28



3.8 Placing a Class in a Separate File for Reusability (Cont.)

- `#include` preprocessor directive
  - Used to include header files
    - Instructs C++ preprocessor to replace directive with a copy of the contents of the specified file
  - Quotes (“ ”) indicate user-defined header files
    用户定义头文件 (# include "GradeBook.h“)
    - Preprocessor first looks in current directory
    - If the file is not found, looks in C++ Standard Library directory
  - Angle brackets (< >) indicate C++ Standard Library
    C++标准库头文件(#include <string>)
    - Preprocessor looks only in C++ Standard Library directory

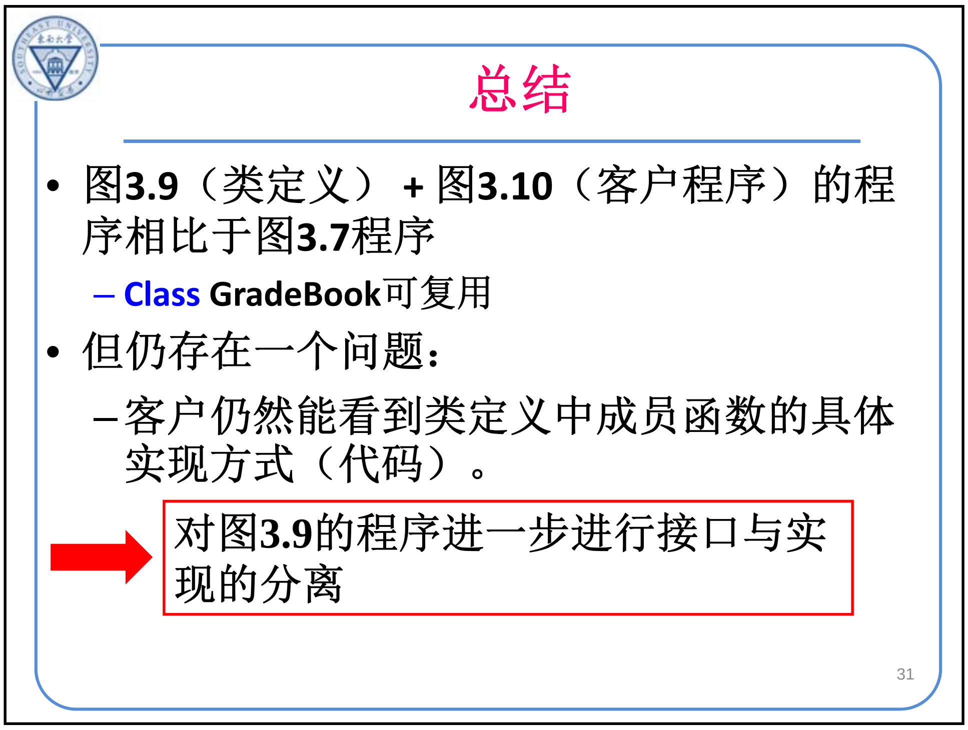
29



3.8 Placing a Class in a Separate File for Reusability (Cont.)

- Creating objects （在客户代码，如main函数中）
  - Compiler must know size of object 编译客户代码时必须知道对象的大小，即占用内存空间的大小
  - 如果客户代码中声明同一个类的多个对象
    - C++ objects typically contain only data members
      内存中每一个对象仅包含数据成员
    - Compiler creates one copy of class’s member functions. This copy is shared among all the class’s objects. 编译器创建一个成员函数的拷贝与所有该类的对象共享

30

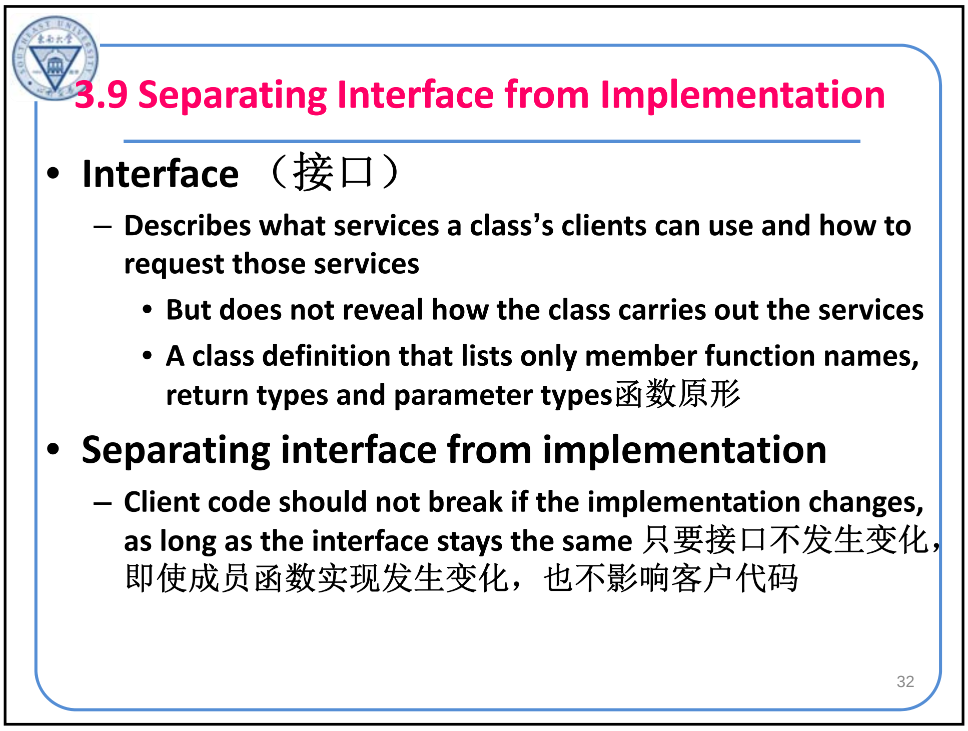


总结

- 图3.9（类定义） + 图3.10（客户程序）的程序相比于图3.7程序
  - Class GradeBook可复用
- 但仍存在一个问题：
  - 客户仍然能看到类定义中成员函数的具体实现方式（代码）。

对图3.9的程序进一步进行接口与实现的分离

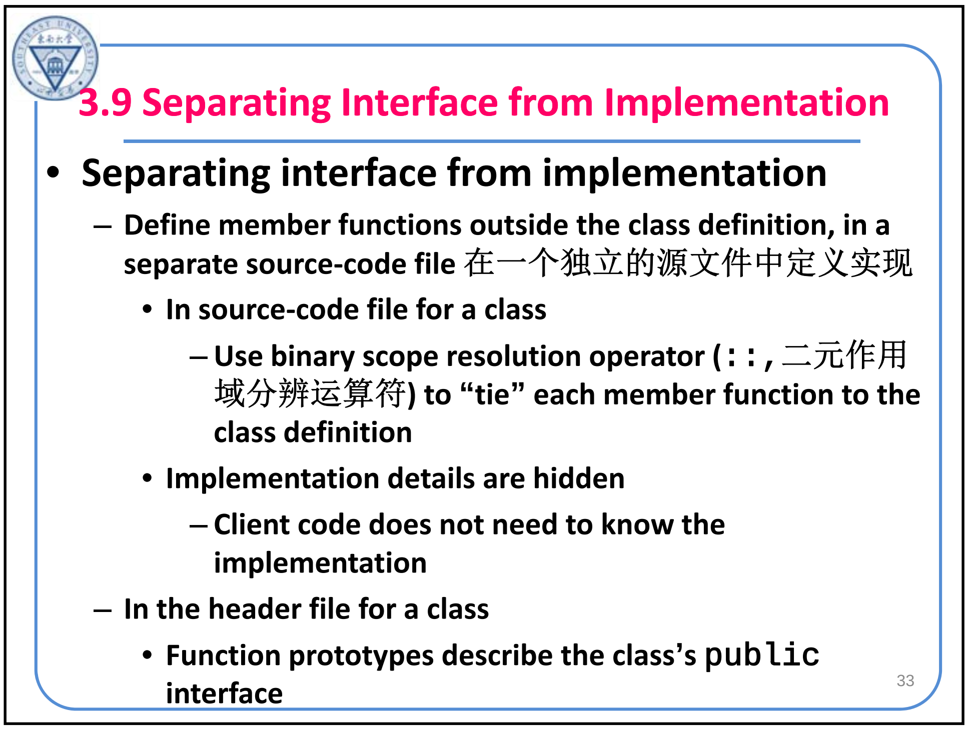
31



3.9 Separating Interface from Implementation

- **Interface （接口）**
  - **Describes what services a class's clients can use and how to request those services**
    - **But does not reveal how the class carries out the services**
    - **A class definition that lists only member function names, return types and parameter types**函数原形
- **Separating interface from implementation**
  - **Client code should not break if the implementation changes, as long as the interface stays the same** 只要接口不发生变化，即使成员函数实现发生变化，也不影响客户代码

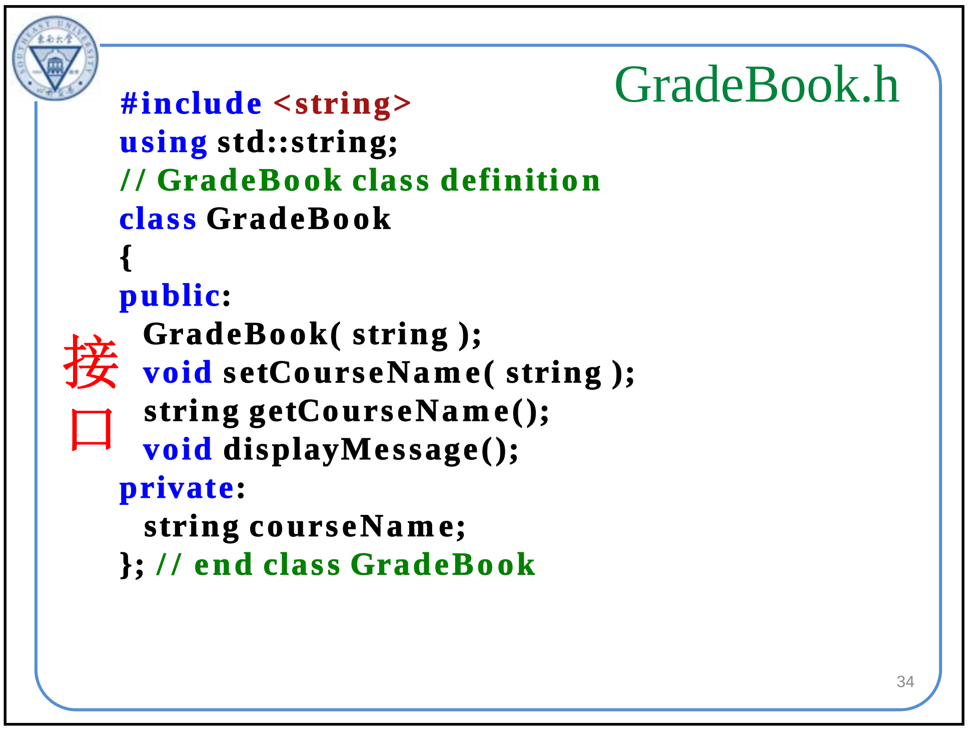
32



3.9 Separating Interface from Implementation

- Separating interface from implementation
  - Define member functions outside the class definition, in a separate source-code file 在一个独立的源文件中定义实现
    - In source-code file for a class
      - Use binary scope resolution operator (`::`, 二元作用域分辨运算符) to “tie” each member function to the class definition
    - Implementation details are hidden
      - Client code does not need to know the implementation
  - In the header file for a class
    - Function prototypes describe the class’s `public` interface

33

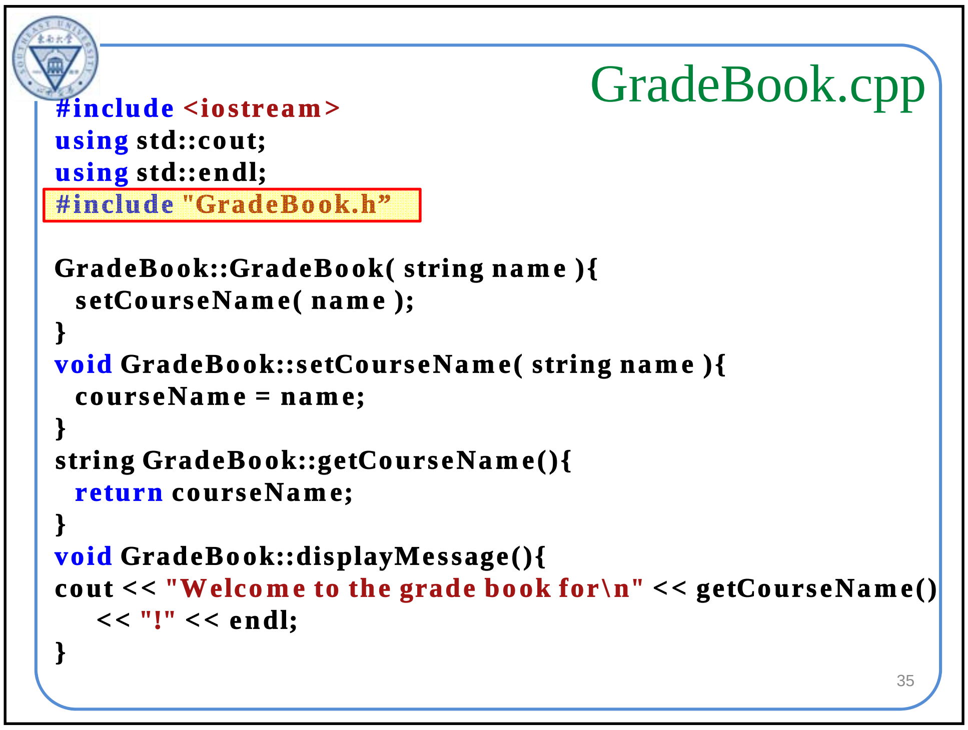


GradeBook.h

```
#include <string>
using std::string;
// GradeBook class definition
class GradeBook
{
public:
  GradeBook( string );
  void setCourseName( string );
  string getCourseName();
  void displayMessage();
private:
  string courseName;
}; // end class GradeBook
```

接口

34



GradeBook.cpp

```
#include <iostream>
using std::cout;
using std::endl;
#include "GradeBook.h”

GradeBook::GradeBook( string name ){
  setCourseName( name );
}
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;
}
```

35

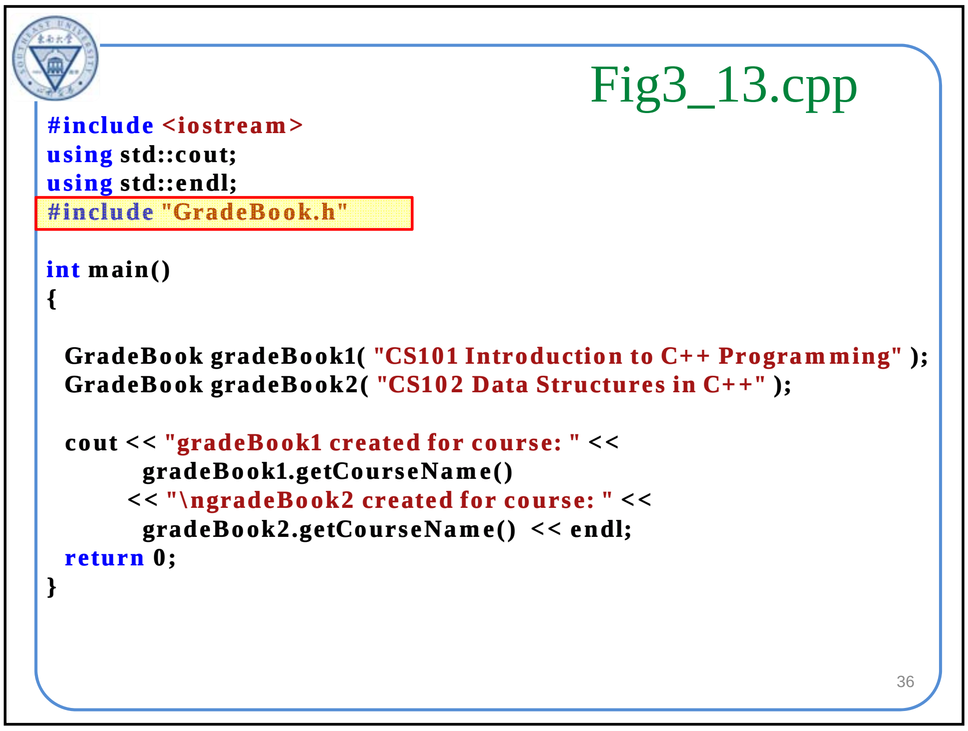


Fig3_13.cpp

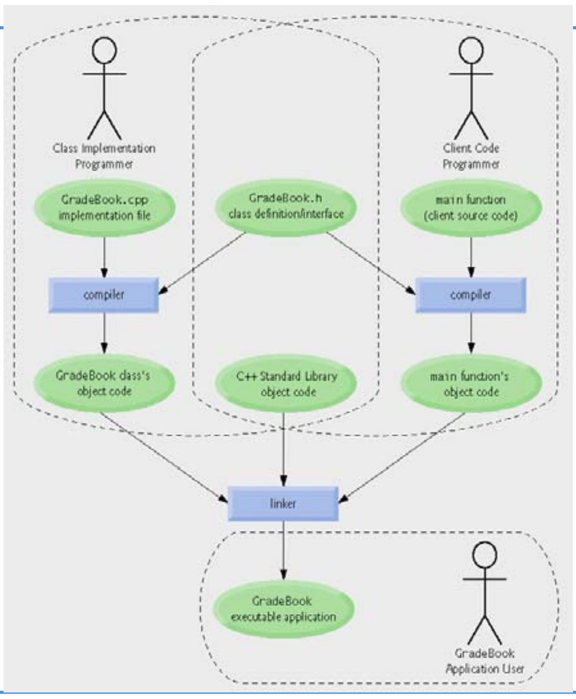
```
#include <iostream>
using std::cout;
using std::endl;
#include "GradeBook.h"

int main()
{

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

  cout << "gradeBook1 created for course: " <<
         gradeBook1.getCourseName()
        << "\ngradeBook2 created for course: " <<
         gradeBook2.getCourseName()  << endl;
  return 0;
}
```

36

总结

- 实现与接口分离（共包括3个文件）
  - GradeBook.h 类定义及接口
  - GradeBook.cpp 类中成员函数的实现代码
    - 必须#include "GradeBook.h”
  - fig3_13.cpp 客户代码源文件
    - 必须#include "GradeBook.h”
  - 客户代码使用GradeBook类时，只需要GradeBook.h文件以及GradeBook类的可执行代码(.lib或.dll文件)

37


Class Implementation Programmer
Client Code Programmer
GradeBook.cpp implementation file
GradeBook.h class definition/interface
main function (client source code)
compiler
compiler
GradeBook class's object code
C++ Standard Library object code
main function's object code
linker
GradeBook executable application
GradeBook Application User

38

3.10 Validating Data with *set* Functions

- *set* functions can validate data
  - Known as validity checking 有效性检验
  - The data member contains a valid value
  - Can return values indicating that attempts were made to assign invalid data
- 例：利用set函数限制课程名称长度最长为25个字符

39

GradeBook.cpp

```
void GradeBook::setCourseName( string name )
{
  if ( name.length() <= 25 )
    courseName = name;
  if ( name.length() > 25 )
  {
    // set courseName to first 25 characters of parameter name
    courseName = name.substr( 0, 25 ); // start at 0, length of 25
    cout << "Name \"" << name << "\" exceeds maximum length (25).\n"
      << "Limiting courseName to first 25 characters.\n" << endl;
  }
} // end function setCourseName
```

```
GradeBook gradeBook1( "CS101 Introduction to
Programming in C++" );
```

courseName实际存储的是CS101 Introduction to Pro

40