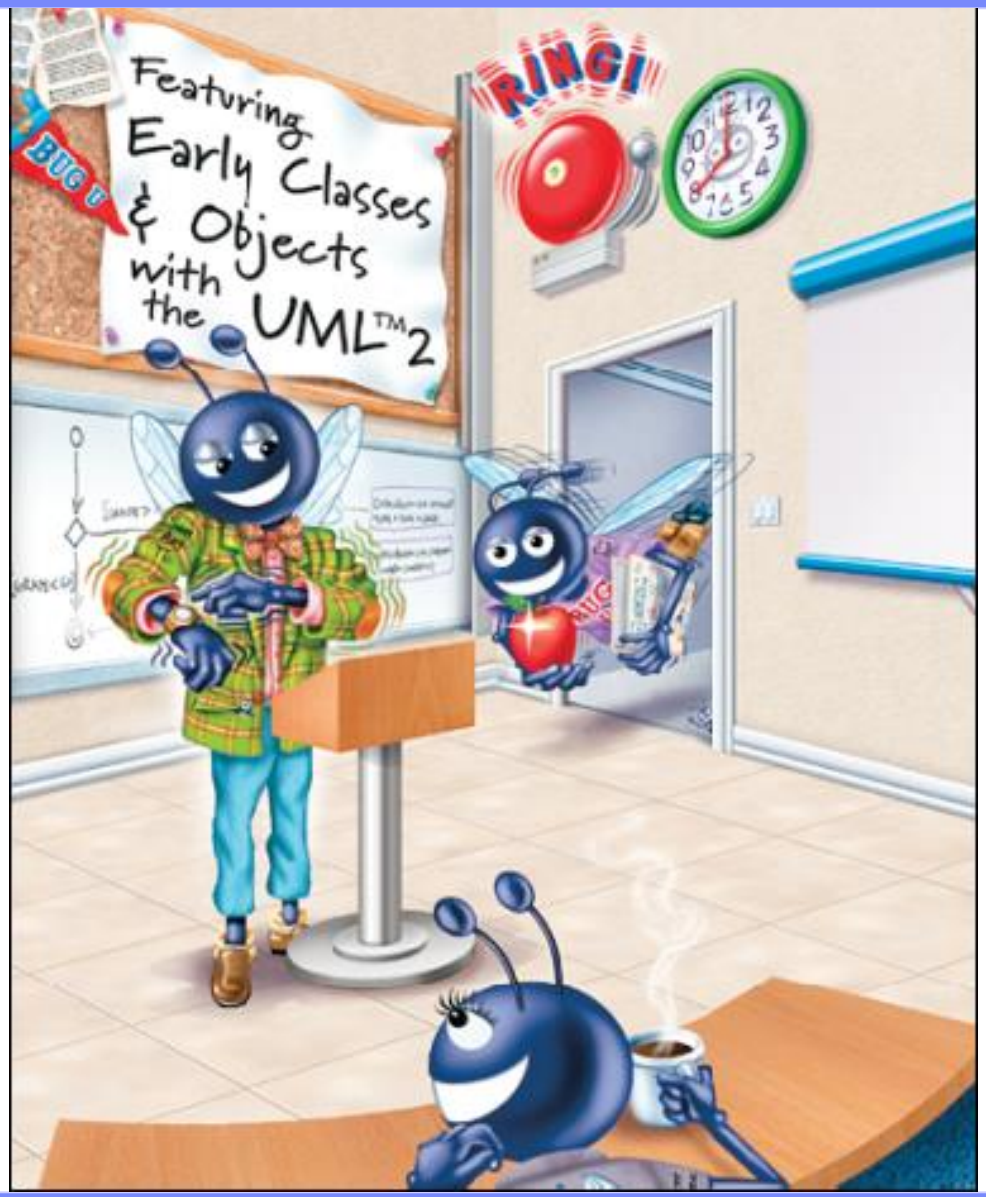
面向对象技术与编程

C++ How to Program（9th）

西安财经学院 信息学院



第3章
类与对象

学习目标:

- 如何定义类
- 如何调用成员函数
- 构造函数
- 实现与接口分离

0. 从面向过程到面向对象

- 抽象的过程
- 面向对象的程序设计的特点
- 库和类

抽象的过程

- 计算机的工作是建立在抽象的基础上。
  - 机器语言和汇编语言是对机器硬件的抽象
  - 高级语言是对汇编语言和机器语言的抽象
- 现有抽象的问题：
  - 要求程序员按计算机的结构去思考，而不是按要解决的问题的结构去思考。
  - 当程序员要解决一个问题时，必须要在机器模型和实际要解决的问题模型之间建立联系。
  - 而计算机的结构本质上还是为了支持计算，当要解决一些非计算问题时，这个联系的建立是很困难的

面向对象的程序设计

- 为程序员提供了创建工具的功能
- 解决一个问题时
  - 程序员首先考虑的是需要哪些工具
  - 创建这些工具
  - 用这些工具解决问题
- 工具就是所谓的对象
- 现有的高级语言提供的工具都是数值计算的工具

过程化 vs 面向对象

以计算圆的面积和周长的问题为例

- 过程化的设计方法：从功能和过程着手
  - 输入圆的半径或直径
  - 利用$S=\pi r^2$和$C=2\pi r$计算面积和周长
  - 输出计算结果
- 面向对象的程序设计方法：
  - 需要什么工具。如果计算机能提供给我们一个称为圆的工具，它可以以某种方式保存一个圆，告诉我们有关这个圆的一些特性，如它的半径、直径、面积和周长。
  - 定义一个圆类型的变量，以他提供的方式将一个圆保存在该变量中，然后让这个变量告诉我们这个圆的面积和周长是多少

面向对象的程序设计的特点

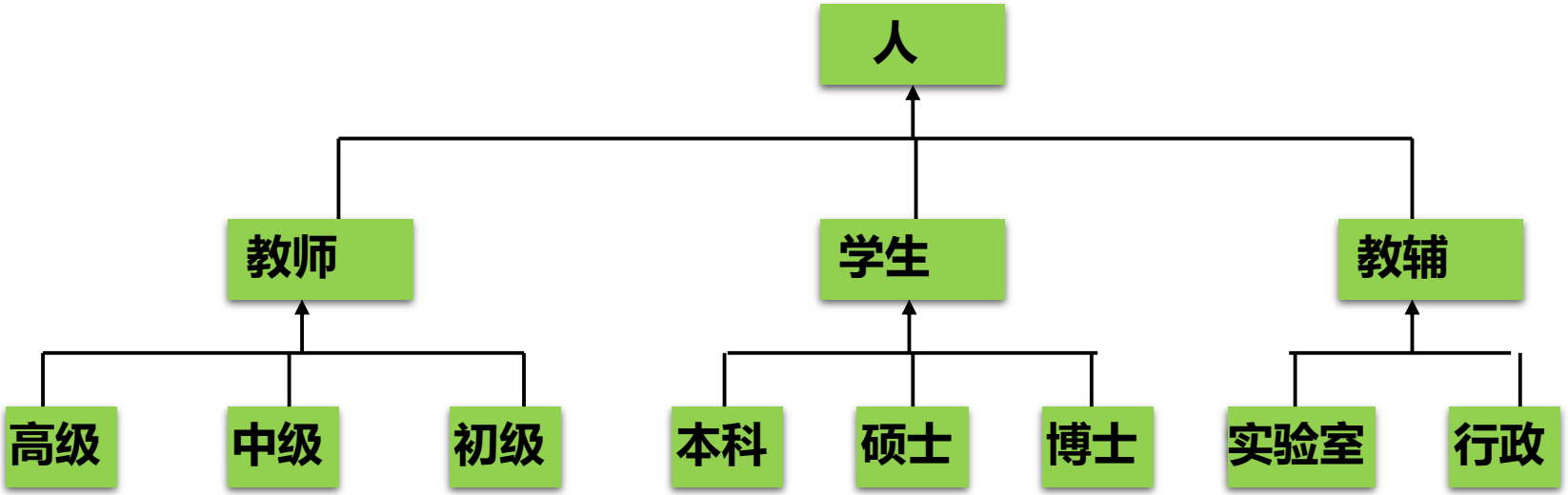
- 代码重用：

  圆类型也可以被那些也需要处理圆的其他程序员使用

- 实现隐藏：
  - 类的创建者创造新的工具
  - 类的使用者则收集已有的工具快速解决所需解决的问题
  - 这些工具是如何实现的，类的使用者不需要知道

面向对象的程序设计的特点

- **继承**：在已有工具的基础上加以扩展，形成一个功能更强的工具。如在学校管理系统中，可以形成如下的继承关系



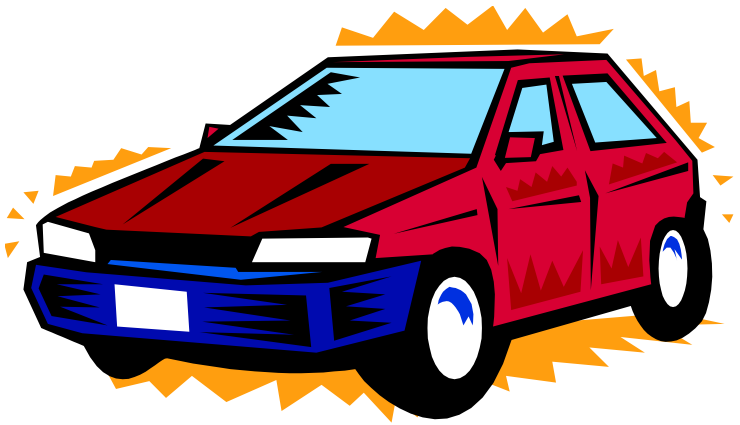
面向对象的程序设计的特点

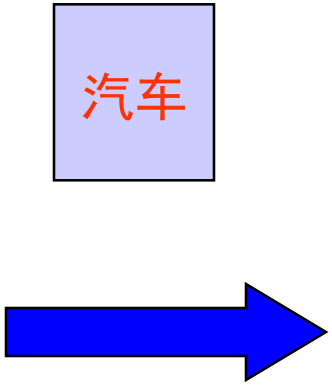
- 多态性：
  - 当处理层次结构的类型时，程序员往往想把各个层次的对象都看成是基类成员。
  - 如需要对教师进行考核，不必管他是什么职称，只要向所有教师发一个考核指令。每位教师自会按照自己的类型作出相应的处理。如高级职称的教师会按高级职称的标准进行考核，初级职称的教师会按初级职称的标准进行考核。
- 好处：程序代码就可以不受新增类型的影响。如增加一个院士的类型，它也是教师类的一个子类，整个程序不用修改，但功能得到了扩展。

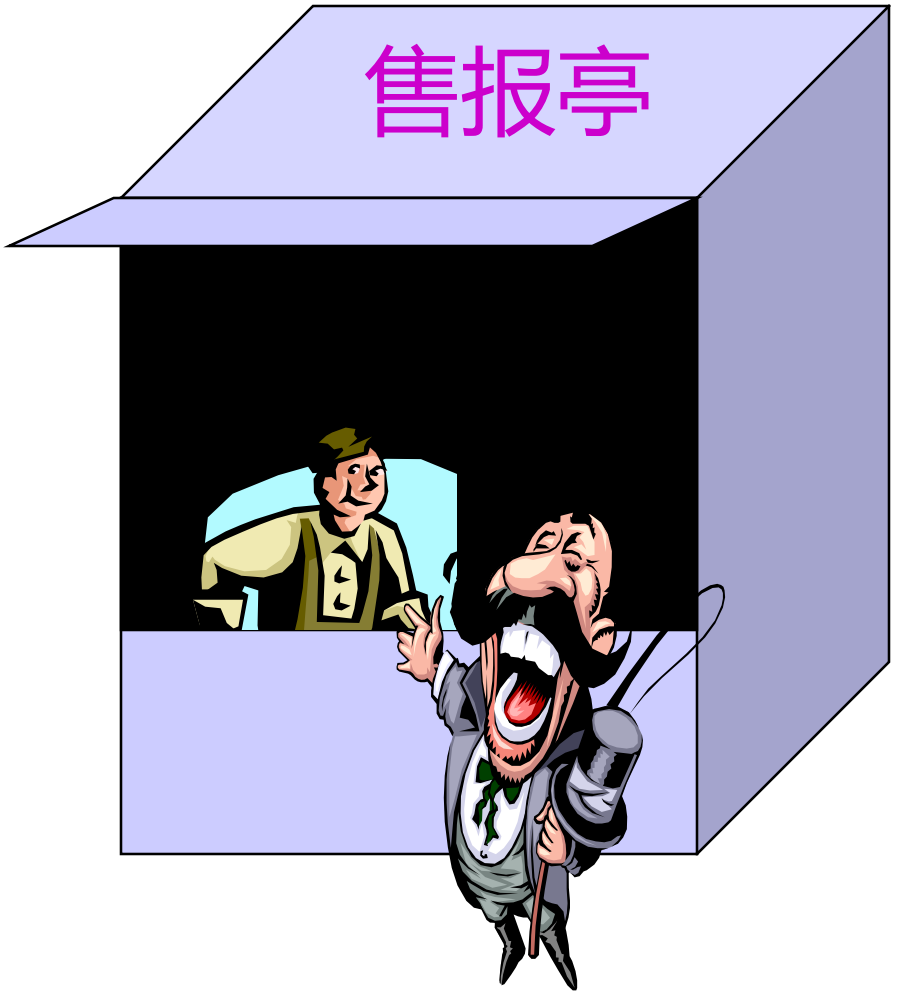
1 Classes, Objects, Member Functions and Data Members

- 类(Class)：具有相同属性和行为的一组对象的集合。其内部包括属性和行为两个主要部分。
- 对象(Object)：集合中的每一个个体称为一个对象。
- 数据成员(Data Member)：标志每个个体都具有的某方面的特性（有时也称为属性）。
- 函数成员(Function Member)：用于对数据成员进行操作的动作。

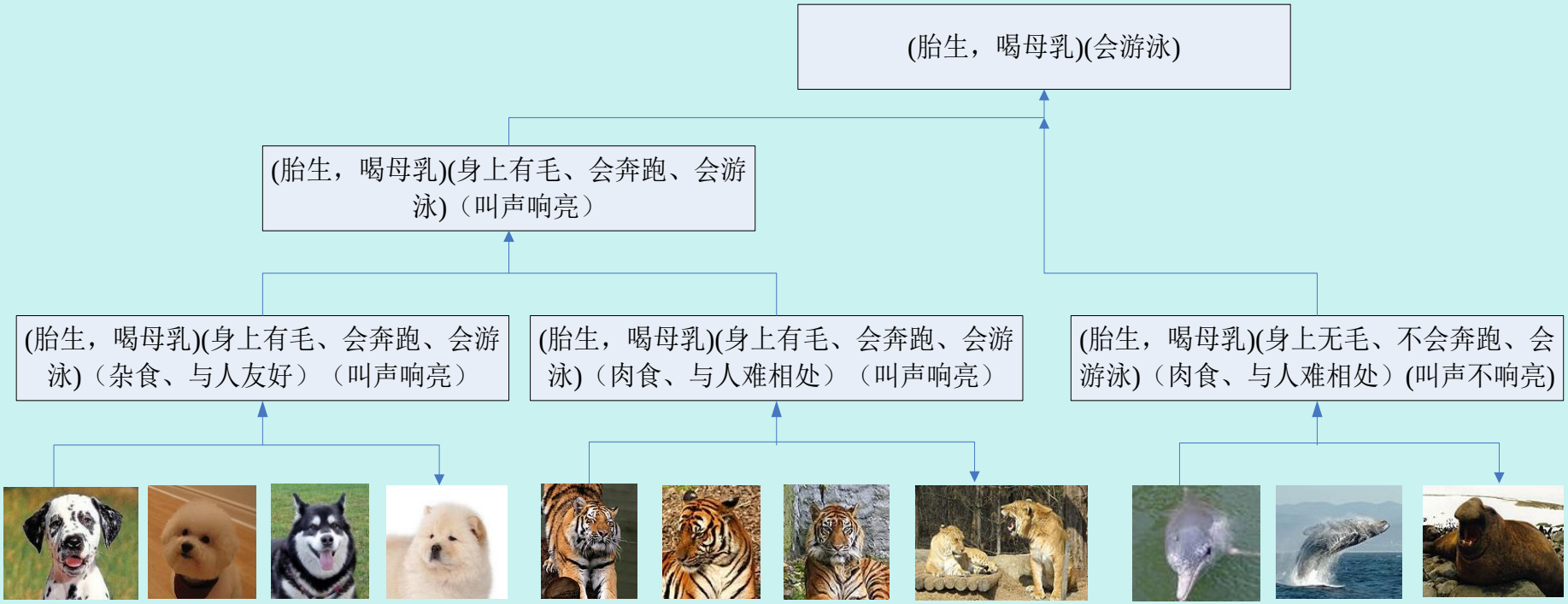
1 Classes, Objects, Member Functions and Data Members






售报亭


属性
报刊A价格
报刊B价格
…
钱箱
服务
报刊零售
款货清点


(胎生，喝母乳)(会游泳)
(胎生，喝母乳)(身上有毛、会奔跑、会游泳)（叫声响亮）
(胎生，喝母乳)(身上有毛、会奔跑、会游泳)（杂食、与人友好）（叫声响亮）
(胎生，喝母乳)(身上有毛、会奔跑、会游泳)（肉食、与人难相处）（叫声响亮）
(胎生，喝母乳)(身上无毛、不会奔跑、会游泳)（肉食、与人难相处）(叫声不响亮)

类

水泥预制板制造攻略
配方: (水、水泥、钢筋、沙子、粘合剂、无敌神水)
制作: 0度以上，先...再...然后...最后...收工。
要点: 使出吃奶力气搅拌（人或搅拌机）

对象

- 容易理解：
  - 1. 从类（秘籍）可以生成一个或多个对象（真正的水泥板）
  - 2. 对于每个真正的对象，可以：
    - ✓ （1）查询数据成员：如每块水泥板中包含多少水泥？多少水？
    - ✓ （2）执行成员函数
  - 3. 对于类却不能干这些事

2 本节用到的示意类

- 类名：GradeBook
- Topics covered:
  - Member functions 成员函数
  - Data members 数据成员
  - Clients of a class Clients类
  - Separating interface from implementation 接口与实现分离
  - Data validation 数据的有效性

3 Defining a Class With a Member Function

- class definition
  - 通知编译器哪些数据成员和成员函数属于该类
  - 关键字 class
  - 定义体在花括号内 ({ })
    - ✓ 声明数据成员和成员函数
    - ✓ 访问修饰符 public:
      - ◈ 其他函数和其他类的成员函数可以访问

```
1  // Fig. 3.1: fig03_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 welcom
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 e
20 int main()
21 {
22    GradeBook myGradeBook; // create a GradeBook object nar
23    myGradeBook.displayMessage(); // call object's displayM
24    return 0; // indicate successful termination
25 } // end main
```

```
Welcome to the Grade Book!
```

Beginning of class definition for class

Beginning of class bo

Access specifier **public**; makes membe the public

Member function **displayMessge** returns nothing

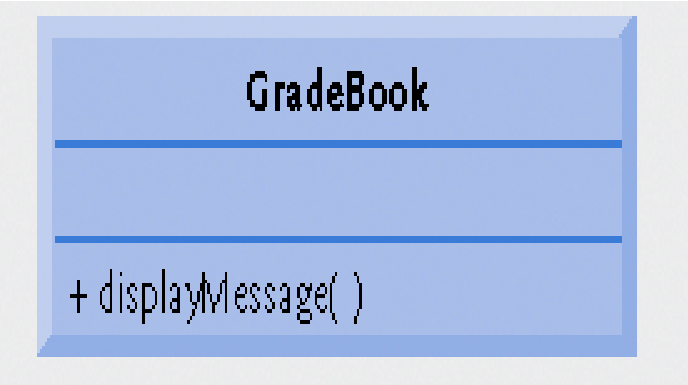
End of class body

Use dot operator to call **GradeBook**' s member function

3 Defining a Class With a Member Function

- UML class diagram for class GradeBook
  - UML is a graphical language used by programmers to represent their object-oriented systems in a standard manner.
  - UML diagram

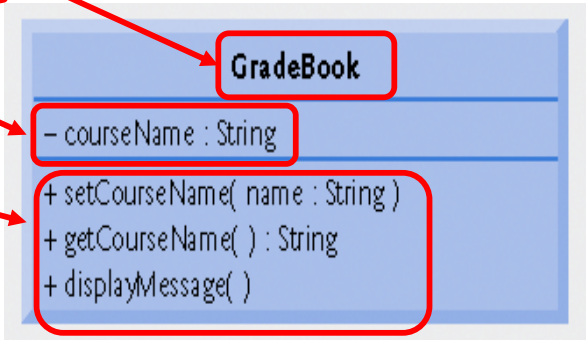
GradeBook类的UML类图



UML class diagram indicating that class GradeBook has a public displayMessage operation.

3 Defining a Class With a Member Function

- UML class diagram
  - 由三部分组成的矩形
    - ✓ 顶部包含水平居中、黑体的类的名字
    - ✓ 中部包含类的属性（数据成员）
    - ✓ 底部包含类的操作（成员函数）
      - 操作前面的 (+) 表示该操作为 public



4 Defining a Member Function with a Parameter

- 函数参数
  - 函数需要客户提供相关信息来完成任务
  - 客户在函数调用时所提供的参数值拷贝给函数的参数
  - 类似标准C语言的“实-形”转换

```
1  // Fig. 3.3: fig03_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

上一个例子的扩展版本

```
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
```

Passing an argument to the member function

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

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

主函数定义字符型变量→ 类产生对象→对变量进行赋值

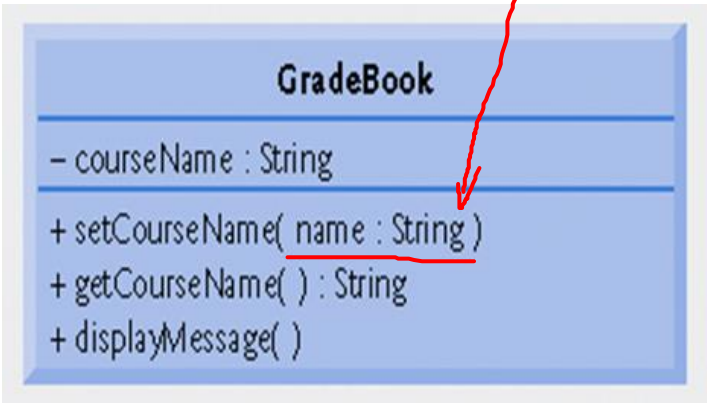
→调用对象的成员函数(实参—形参传递)

4 Defining a Member Function with a Parameter

- A string
  - 字符集合
  - C++ 标准类库 std::string
    - ✓ 需要 #include <string>
- getline 函数
  - 读取一行输入
  - 例：
    - ✓ getline(cin, nameOfCourse);

4 Defining a Member Function with a Parameter

- 参数列表
  - UML 中的表示方式
    - ✓ 在成员函数的”( )”中，**参数名称：参数类型**



5 Data Members, set Functions and get Functions

- 局部变量
  - 在函数体定义内部声明的变量
    - ✓ 不能在函数体外部使用
  - 当函数终止
    - ✓ 局部变量将被销毁
- 属性
  - 存在于对象的整个生命周期内
  - 表示为数据成员
    - ✓ 即类定义中的变量
  - 每个对象维护一份自己的属性拷贝

5 Data Members, set Functions and get Functions

- **访问修饰符 private**
  - **使得数据成员或成员函数只能由类的public成员函数访问**
  - **类成员的默认访问为 private**
  - **数据隐藏**
- Variables or functions declared after access specifier(访问修饰符) `private` is accessible only to member functions of the class for which they are declared.
- Declaring data members with access specifier `private` is known as data hiding.

```
1  // Fig. 3.5: fig03_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 namespace std;
7
8
9
10 #include <string> // program uses C++ standard string class
11
12
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

```
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
46    GradeBook myGradeBook; // create a GradeBook object
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.
这个变量只能被类自已定义的函数所访问，不能被类外的函数访问，哪怕是main()也不行!

该处其实是访问类的私有变量courseName,但它不能直接访问，只能通过类提供的公有函数来访问(类中的private数据成员)。

```
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
```

```
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!
```

该处是修改类的private数据成员(私有变量)，不能直接访问，只能通过类提供的公有函数进行访问

意味着：

- 用户可以通过类设定的成员函数访问类的成员变量。
- 如果是类的成员函数，则公有、私有变量都可以访问；
- 如果不是类的成员函数(如main()函数)，只能通过类提供的公有成员函数访问类的私的变量

5 Data Members, set Functions and get Functions

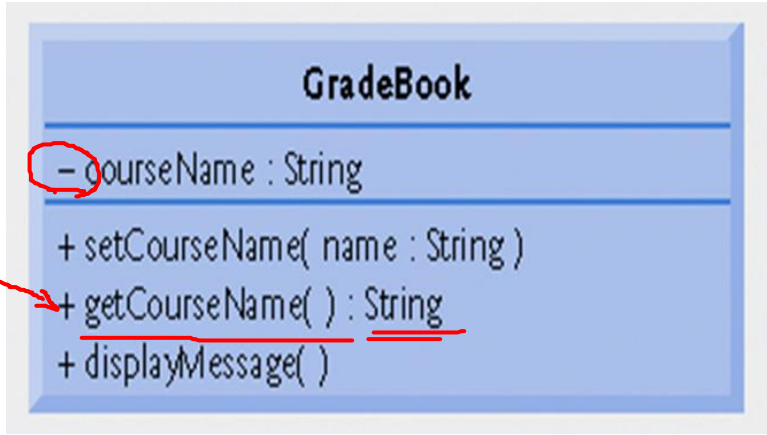
软件工程知识：根据经验，数据成员应该声明为private，成员函数应该声明为public。如果某些成员函数只是被该类内部定义的其他成员函数访问，而不会涉及类外的函数(如main())的访问，那么它们更适合声明为private。

5 Data Members, set Functions and get Functions

- **Software engineering with *set* and *get* functions**
  - public 成员函数允许类的客户使用 set 和 get 函数访问类的 private 数据成员
  - *set* 函数有时被称为 mutator (更换器)，*get* 有时被称为 accessor (访问器)
  - 允许类的创建者来控制客户如何访问 private 数据

5 Data Members, set Functions and get Functions

- UML diagram
  - 标识操作的返回值类型
    - ✓ 在函数名的”（）”后加”：”和返回值类型
  - 数据成员名称前面的“-”标识该成员为private成员



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

6 Initializing Objects with Constructors

- **Constructors (构造函数)**
  - 用来在对象创建时进行初始化对象中的数据的函数
    - ✓ 当对象创建时被隐式调用：实例化对象时自动被调用，不能由用户写调用语句
      - 什么是隐式？
      - 什么是显式？
    - ✓ 必须与类同名
    - ✓ 不能有返回值
  - 缺省的构造函数没有参数
    - ✓ 当类自已没有定义构造函数时，编译器会自动提供缺省的构造函数（不可能不被调用）

```cpp
// Fig. 3.7: fig03_07.cpp
// Instantiating multiple objects of the GradeBook class and using
// the GradeBook constructor to specify the course name
// when each GradeBook object is created.
#include <iostream>
using std::cout;
using std::endl;

#include <string> // program uses C++ standard string class
using std::string;

// GradeBook class definition
class GradeBook
{
public:
   // constructor initializes courseName with string supplied as argument
   GradeBook( string name )
   {
      setCourseName( name ); // call set function to initialize courseName
   } // end GradeBook constructor

   // function to set the course name
   void setCourseName( string name )
   {
      courseName = name; // store the course name in the object
   } // end function setCourseName

```

Constructor has same name as class and no return type(构造函数与类同名，且没有返回值)

Initialize data member

```
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
```

该成员变量是私有变量，只能被类中定义的函数所访问，不能被其它函数访问

```
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
```

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

Creating objects implicitly calls the constructor

隐式调用构造函数

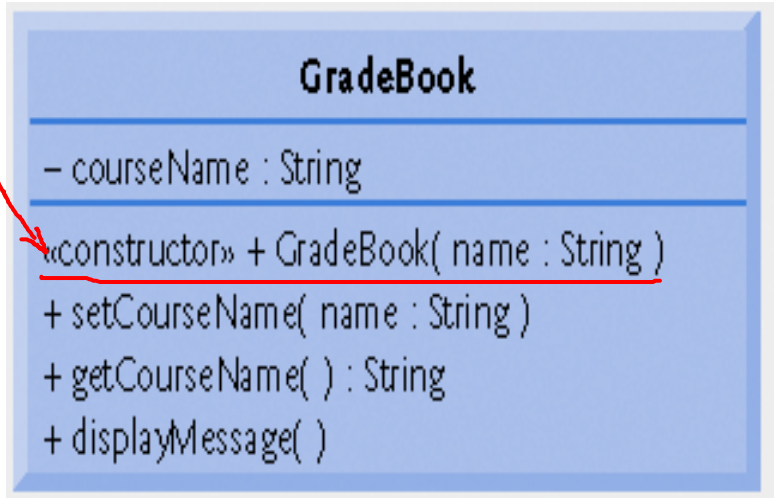
不能直接调用变量CourseName

6 Initializing Objects with Constructors

错误预防技巧：除非没有必要初始化类的数据成员，否则需要提供构造函数，这样可以保证当类的每个新对象被创建时，类的数据成员都是用有意义的值进行了初始化。毕竟由系统自动提供的初始值未必都是那么恰当。

6 Initializing Objects with Constructors

- Constructors in a UML class diagram
  - 在操作部分出现
  - 为了与其他操作进行区分
    - ✓ 在构造函数名前加：<<constructor>>
  - 通常放置在其他操作之前



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

7 Placing a Class in a Separate File for Reusability
（将类定义+实现与主函数放在不同的文件中便于重用）

- Header files （.h）
  - 放置类的定义
    - ✓ 允许编译器在其他地方识别该类
- Source files (.cpp)： 源文件
- Driver files
  - 用来测试软件的程序 (如：测试 classes)
  - 包含 main() 函数，可以被执行

```
1  // Fig. 3.9: GradeBook.h
2  // GradeBook class definition in a separate file from main.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  #include <string> // class GradeBook uses C++ standard string class
8  using std::string;
9
10 // GradeBook class definition
11 class GradeBook
12 {
13 public:
14    // constructor initializes courseName with string supplied as argument
15    GradeBook( string name )
16    {
17       setCourseName( name ); // call set function to initialize courseName
18    } // end GradeBook constructor
19
20    // function to set the course name
21    void setCourseName( string name )
22    {
23       courseName = name; // store the course name in the object
24    } // end function setCourseName
25
```

Class definition is in a header file

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

本例特点：类的成员函数定义与实现放在同一个文件中。

```
1  // Fig. 3.10: fig03_10.cpp
2  // Including class GradeBook from file GradeBook.h for use in main.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  #include "GradeBook.h" // include definition of class GradeBook
8
9  // function main begins program execution
10 int main()
11 {
12    // create two GradeBook objects
13    GradeBook gradeBook1( "CS101 Introduction to C++ Programming" );
14    GradeBook gradeBook2( "CS102 Data Structures in C++" );
15
16    // display initial value of courseName for each GradeBook
17    cout << "gradeBook1 created for course: " << gradeBook1.getCourseName()
18       << "\ngradeBook2 created for course: " << gradeBook2.getCourseName()
19       << endl;
20    return 0; // indicate successful termination
21 } // end main
```

Including the header file causes the class definition to be copied into the file

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

本例中将类（定义+实现）与主程序分开；
例3.7是将它们全部放入一个文件中，可移植性不好。

7 Placing a Class in a Separate File for Reusability

- Creating objects（创建对象）
  - 编译器必须知道对象的大小
    - ✓ 对象的大小为类的数据成员的大小
    - ✓ 编译器创建一份类的成员函数的拷贝，该拷贝为所有类的对象所共享

7 Placing a Class in a Separate File for Reusability

错误预防技巧：

To ensure that the preprocessor can locate header files correctly, #include preprocessor directives should place the names of user-defined header files in quotes (e.g., “GradeBook.h”) and place the names of C++ Standard Library header files in angle brackets (e.g., <iostream>).

【用户自己定义的头文件放在“”标注内，C++标准库的头文件放在<>的标注内】

8 Separating Interface from Implementation (将类的定义与类函数的实现分开)

Interface (接口)

- 描述客户能够使用类的哪些服务，如何请求这些服务
  - ✓ 包含成员函数名称、返回类型、参数类型的类定义，即：函数原型
- 类的接口包含类的public成员函数 (services)

8 Separating Interface from Implementation

- 如果服务的实现改变，只要接口保持不变，客户端的代码就无需改变
- 在头文件中声明该类提供的接口
- 在另外一个源文件内来实现类的成员函数
- 在该源文件中用作用于标识符或者作用域解析运算符（::）将成员函数与类联系起来
- 客户代码无需知晓实现细节

因此，可以将前面的 .h 文件(例3.9)继续细化

```
1  // Fig. 3.11: 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(原型)
只有定义了4个函数，没有函数的具体实现

软件工程知识：

在头文件里定义的成员函数原型中，参数只需要提供他们的类型，即使提供了参数的名字编译器在编译源程序时也会忽略他们。但平时有些程序员习惯性地加个名字，以备检查，这并不影响程序的最终编译结果。

```
1  // Fig. 3.12: 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 a
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**

在GradeBook.cpp文件中实现上述头文件中的4个函数

```
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
```

在GradeBook.cpp文件中实现上述头文件中的4个函数

```
1  // Fig. 3.13: fig03_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
```

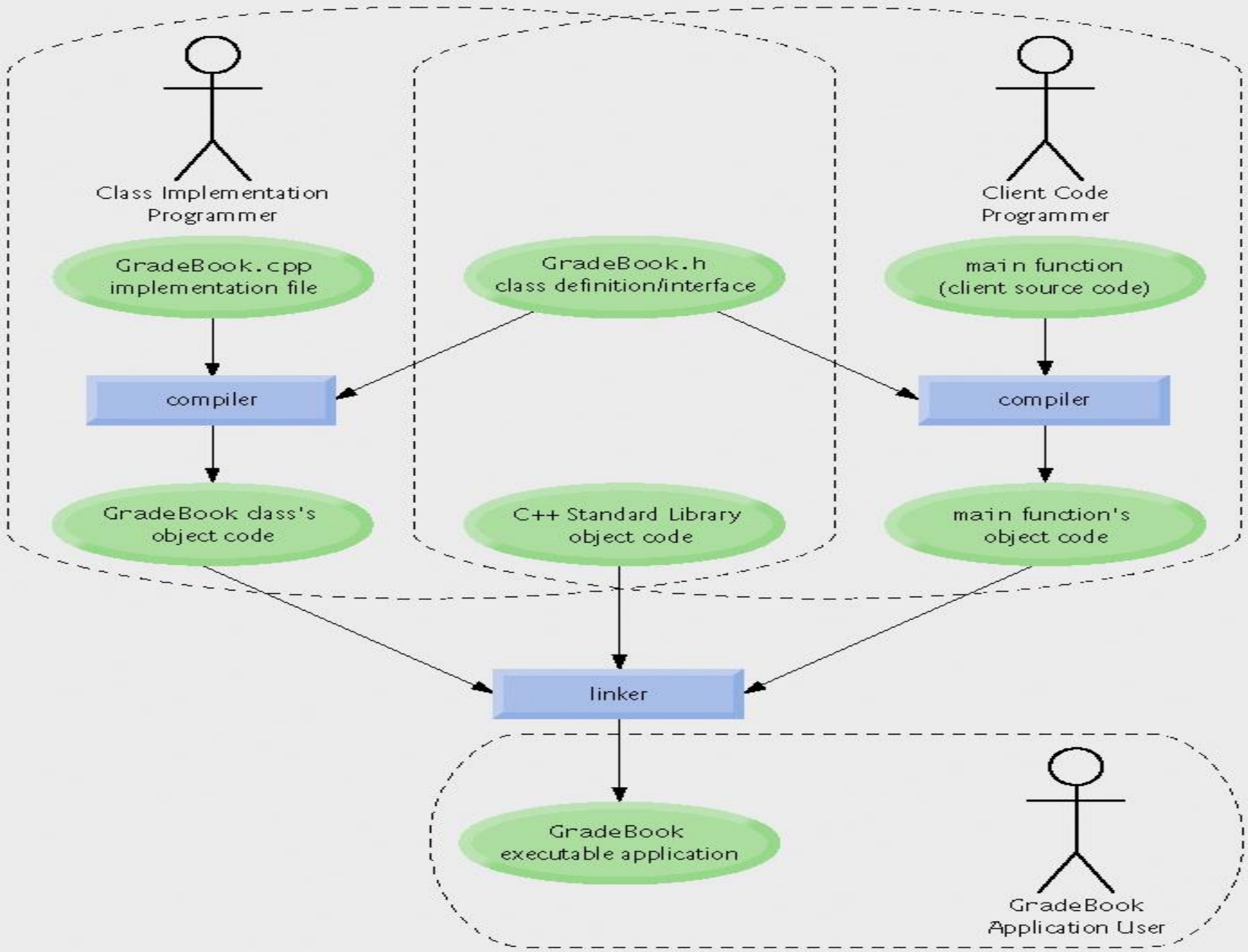
使用刚才定义的头文件

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

3.11-3.12-3.13 三个文件共同完成了一件任务

8 Separating Interface from Implementation

- The Compilation and Linking Process
  - 源文件被编译为类的目标代码
    - ✓ 只需向客户提供头文件和目标代码
  - 客户必须在自己的代码中包含头文件
  - 创建可执行程序
    - ✓ 客户程序的目标代码必须与类的目标代码和在应用程序中用到的 C++ 标准类库的目标代码进行连接


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

9 Validating Data with set Functions

- *set* functions can validate (验证)data
- 扩展类GradeBook的成员函数SetCourseName()，以便进行有效性检查。
  - 保持对象的数据成员具有有效值
  - 可以通过返回值来指示设置数据的有效性
- string member functions
  - length – 返回字符串的长度
  - substr – 返回字符串的子串

```
1  // Fig. 3.15: GradeBook.h
2  // GradeBook class definition presents the public interface of
3  // the class. Member-function definitions appear in GradeBook.cpp.
4  #include <string> // program uses C++ standard string class
5  using std::string;
6
7  // GradeBook class definition
8  class GradeBook
9  {
10 public:
11    GradeBook( string ); // constructor that initializes a GradeBook object
12    void setCourseName( string ); // function that sets the course name
13    string getCourseName(); // function that gets the course name
14    void displayMessage(); // function that displays a welcome message
15 private:
16    string courseName; // course name for this GradeBook
17 }; // end class GradeBook
```

```
1  // Fig. 3.16: 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
```

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
```

```
1  // Fig. 3.17: fig03_17.cpp
2  // Create and manipulate a GradeBook object; illustrate validation.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  #include "GradeBook.h" // include definition of class GradeBook
8
9  // function main begins program execution
10 int main()
11 {
12    // create two GradeBook objects;
13    // initial course name of gradeBook1 is too long
14    GradeBook gradeBook1( "CS101 Introduction to Programming in C++" );
15    GradeBook gradeBook2( "CS102 C++ Data Structures" );
16
```

```
17    // display each GradeBook's courseName
18    cout << "gradeBook1's initial course name is: "
19       << gradeBook1.getCourseName()
20       << "\ngradeBook2's initial course name is: "
21       << gradeBook2.getCourseName() << endl;
22
23    // modify myGradeBook's courseName (with a valid-length string)
24    gradeBook1.setCourseName( "CS101 C++ Programming" );
25
26    // display each GradeBook's courseName
27    cout << "\ngradeBook1's course name is: "
28       << gradeBook1.getCourseName()
29       << "\ngradeBook2's course name is: "
30       << gradeBook2.getCourseName() << endl;
31    return 0; // indicate successful termination
32 } // 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
```

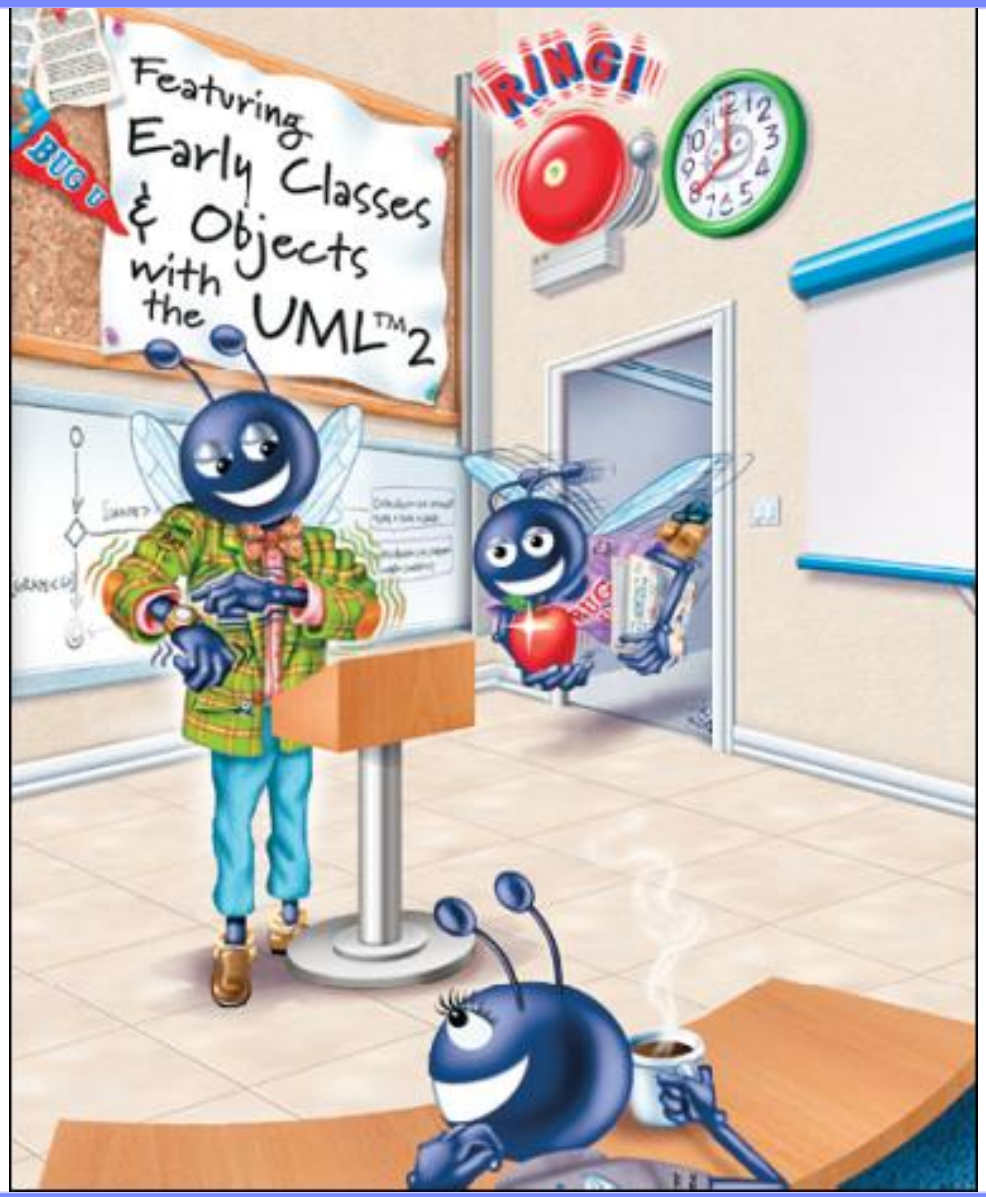
软件工程知识：

把数据成员设置成private，而由public成员函数控制访问数据成员的权力，尤其是写的权力，将有助于保证数据的完整性。

软件工程知识：

设置private数据值的成员函数应当核实所设置的新值是否正确，如果不正确，设置函数应该将private数据成员置于适当的状态中。

--- 结束 ---



Thank you!