객체지향프로그래밍 응용

Lecture 1

Introduction to Classes and Objects
(교과서 3장, Part 1)

1. Object-Oriented Programming
2. 재사용성 (Reusability) in OOP
3. Defining a Class
4. Defining a Member Function with a Parameter

최학남 교수

1. Object-Oriented Programming

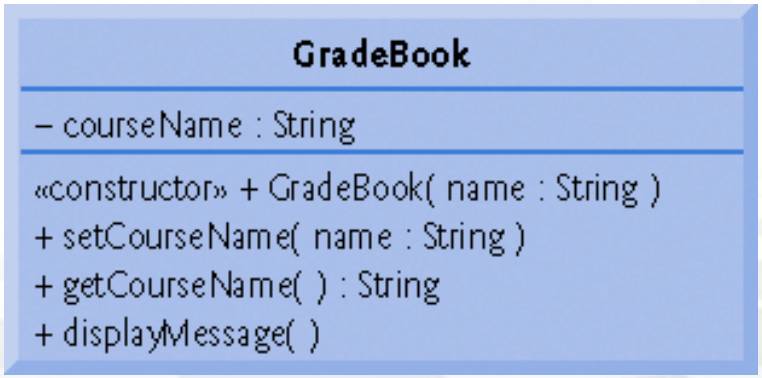
객체지향 프로그래밍

객체지향프로그래밍(object oriented prgramming)이란?

: 실세계의 객체의 특성에 대해 생각하고 이를 모방하여 컴퓨터 프로그램을 작성하는 자연스러운 방법

통합 모델링 언어

- ✓ Unified Modeling Language (UML)
- ✓ 심볼 등의 그래픽으로 표현되는 software 구조의 표현 방식
- ✓ 개발자들이 객체지향 디자인을 효율적으로 표현할 수 있도록 함



객체 (object)

- (실세계를 모방한) 재사용 가능한 (reusable) 소프트웨어 요소
- 우리 주변에 있는 객체의 사례
  - 사람, 동물, 자동차, 핸드폰, 전자레인지, …
- 속성 (attributes)
  - 크기, 모양, 색깔, 무게, …
- 행동 양식 (behaviors)
  - 아기가 울고, 기고, 자고, …
  - 자동차가 가속하고, 멈추고, 회전하고, …

객체지향 설계 (object-oriented design)

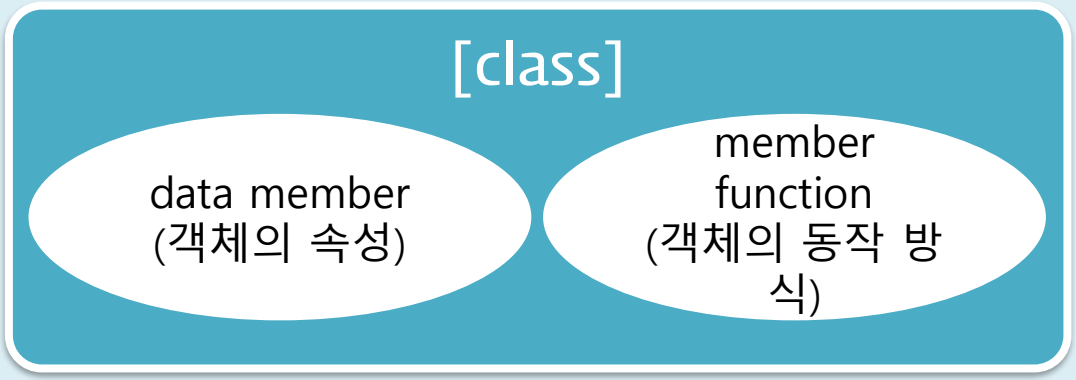
- 현실세계의 객체 체계를 모방한 소프트웨어 설계
- 객체들 사이의 통신(communication)을 모방
  - 메시지(message)를 사용하여 정보를 주고 받음
- 속성과 동작을 캡슐화 (encapsulation)
  - 정보 은닉 (information hiding)
  - 잘 정의된 인터페이스(interface)를 통한 통신

객체 지향 언어 (object-oriented language)

- 객체지향 설계 기법과 객체지향 언어를 사용한 프로그램 제작을 객체지향 프로그래밍이라 부른다. (OOP)

- C++ 은 객체지향 언어이다.

  - 프로그래머는 클래스(class)라 불리는 사용자 정의 (user-defined) 자료형을 만들 수 있다.



2. 재사용성 (Reusability) in OOP

재사용성 (Reusability) in OOP

- 새로운 클래스 작성 또는 프로그램 작성시 기존 클래스의 재사용은 시간과 노력을 줄여준다.
- 재사용은 또한 프로그래머가 더 신뢰성 있고 효율적인 시스템을 구축할 수 있도록 돕는다.
  - 기존의 클래스와 컴포넌트는 이미 테스트 되었고 디버깅되었으며 실행되었던 것이기 때문

3. Defining a Class

Class - A Example

```
1  // Fig. 9.1: Time.h
2  // Declaration of class Time.
3  // Member functions are defined in Time.cpp
4
5  // prevent multiple inclusions of header file
6  #ifndef TIME_H
7  #define TIME_H
8
9  // Time class definition
10 class Time
11 {
12 public:
13    Time(); // constructor
14    void setTime( int, int, int ); // set hour, minute and second
15    void printUniversal(); // print time in universal-time format
16    void printStandard(); // print time in standard-time format
17 private:
18    int hour; // 0 - 23 (24-hour clock format)
19    int minute; // 0 - 59
20    int second; // 0 - 59
21 }; // end class Time
22
23 #endif
```

클래스 정의 (class definition)

- 컴파일러에게 클래스에 속한 멤버함수와 데이터 멤버를 알려준다.
- 클래스의 이름은 class 키워드와 함께 사용하여 시작한다.
- 클래스의 본문(body)은 대괄호 안에 작성되어야 한다. ({})
  - ✓ 데이터 멤버 (멤버 변수)와 멤버 함수의 명기
  - ✓ 접근 지정자 (access-specifier) publ :
    - 이것은 다른 함수와 다른 클래스에서 본 지정자로 지정된 멤버함수나 데이터 멤버함수에 접근이 가능하다는 것을 의미

클래스 정의 예제

```
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 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 **displayMessge** returns nothing

End of class body

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

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

Common Programming Error 3.1

Forgetting the semicolon at the end of a class definition is a syntax error.

멤버 함수 (member function) 정의

- 함수의 반환형 (return type)
  - ✓ 함수가 종료되었을 때 반환할 값의 자료형 (일반적인 함수와 같은 의미)
  - ✓ `void` 는 함수가 아무런 값도 반환하지 않는다는 것을 의미
- 함수 본문은 함수가 어떤 일을 수행하는지 기술함
  - ✓ 중괄호 (`{}`) 에 의해 제한된다

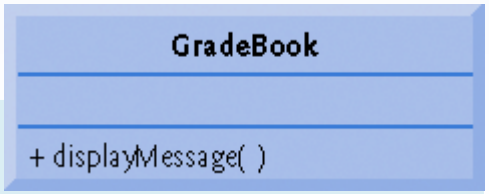
Common Programming Error 3.2

Returning a value from a function whose return type has been declared **void** is a compilation error.

클래스 사용

- 클래스는 사용자 정의 자료형 (user-defined type)
  - ✓ C++ 객체생성시 사용
    - 클래스형의 변수
- dot operator (.)
  - ✓ 객체의 멤버 함수 및 데이터 멤버의 접근에 사용된다
  - ✓ 예제
    - myGradeBook.displayMessage()

      : GradeBook 객체의 멤버함수 displayMessage를 호출



4. Defining a Member Function with a Parameter

매개변수와 인수

- 함수 매개변수 (parameter)

  : 함수가 업무를 실행하는데 필요한 정보

- 함수 인수 (argument)

  ✓ 함수의 매개변수를 위해 함수 호출시 공급된 값

  - 인수 값은 함수 매개변수로 복사된다.

참고: string 객체

- string 객체
  - ✓ 문자열 저장 및 활용을 위한 클래스
  - ✓ C++ 표준 라이브러리의 일부임 (std::string)
    - 헤더파일 <string>에서 정의됨
- getline 함수
  - ✓ newline 문자를 만날 때 까지 입력된 문자열을 읽어들임
  - ✓ 예: getline(cin, nameOfCourse);
    - 표준 입력(키보드)으로부터 문자열 입력을 받아 string 객체인 nameOfCourse에 저장

예제 (매개변수와 인수 사용)

```
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

예제 (매개변수와 인수 사용)

```
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!
```

2주차 1교시 수업을 마칩니다.

객체지향프로그래밍 응용

Lecture 1

Introduction to Classes and Objects
(교과서 3장, Part 2)

1. Defining a Member Function with a Parameter
2. Data Members, *set* Functions and *get* Functions

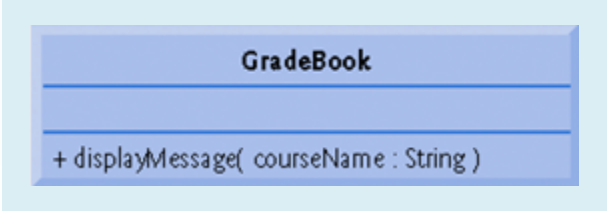
지난 시간 강의 복습

- ✓ 객체지향 프로그래밍 (object-oriented programming, OOP)
- ✓ 클래스 (class)
- ✓ 멤버 함수 (member functions), 멤버 변수 (member variables)
- ✓ 코드 재사용성
- ✓ 매개 변수와 인수

1. Defining a Member Function with a Parameter

매개변수 리스트 (parameter lists)

- ✓ 함수 이름 뒤의 ( )안에 위치하여 함수가 실행되는데 필요한 정보를 넘겨받음
- ✓ 복수의 매개변수는 , 으로 구분되며 임의의 개수가 가능
- ✓ 함수 인자의 개수와 순서, 자료형은 함수 매개변수의 개수, 순서, 자료형과 일치해야 함
- ✓ 매개변수가 있는 경우 UML class diagram은 아래와 같이 표현됨



Common Programming Error 3.5

Defining a function parameter again as a local variable in the function is a compilation error.

Good Programming Practice 3.1

To avoid ambiguity, do not use the same names for the arguments passed to a function and the corresponding parameters in the function definition.

Good Programming Practice 3.2

Choosing meaningful function names and meaningful parameter names makes programs more readable and helps avoid excessive use of comments.

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

지역변수와 멤버변수

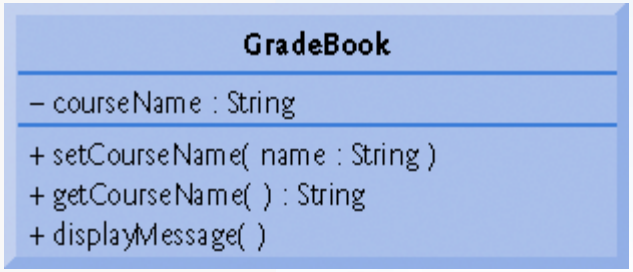
- 지역 변수 (local variables)
  - ✓ 함수 안에서 선언된 변수
    - 함수 밖에서 사용 할 수 없다.
  - ✓ 함수가 소멸될 때 같이 소멸된다.

- 멤버 변수 (member variables)
  - ✓ 객체가 살아있는 동안만 존재
  - ✓ 데이터 멤버로 표현
    - 클래스 정의에 선언된 변수
  - ✓ 클래스의 각 객체는 속성의 복사본을 보유한다.

private 멤버 변수 예제

```
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 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
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
```

<UML Diagram>



set function modifies **private** data

get function accesses **private** data

private 멤버 변수 예제

```
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
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

Accessing **private** data outside class definition

private 멤버 변수 예제

```
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 succes
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!
```

Good Programming Practice 3.3

Place a blank line between member-function definitions to enhance program readability.

접근 지정자 private

- 접근 지정자 private
  - ✓ 데이터 멤버나 멤버 함수에 오직 같은 클래스의 멤버 함수로만 접근이 가능하게 한다.
  - ✓ private 은 클래스 멤버의 기본 (default) 접근 지정자
  - ✓ 자료은닉 (자료의 캡슐화)
- 기타 접근 지정자
  - ✓ public, protected 등

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.)

Common Programming Error 3.6

An attempt by a function, which is not a member of a particular class, to access a **private** member of that class is a compilation error.

Good Programming Practice 3.4

Despite the fact that the **public** and **private** access specifiers may be repeated and intermixed, list all the **public** members of a class first in one group and then list all the **private** members in another group. This focuses the client's attention on the class's **public** interface, rather than on the class's implementation.

Good Programming Practice 3.5

If you choose to list the **private** members first in a class definition, explicitly use the **private** access specifier despite the fact that **private** is assumed by default. This improves program clarity.

Error-Prevention Tip 3.1

Making the data members of a class **private** and the member functions of the class **public** facilitates debugging because problems with data manipulations are localized to either the class's member functions.

Good Programming Practice 3.3

Forgetting to return a value from a function that is supposed to return a value is a compilation error.

소프트웨어공학과 set 와 get 함수

- ✓ public 멤버 함수로 선언
- ✓ private 데이터 멤버의 값을 설정(set) 또는 읽을(get) 수 있도록 정의된 인터페이스 역할
  - 클래스의 설계자는 private 데이터에 접근할 수 있는 방법을 제시해야 함
- ✓ 또한 같은 클래스의 다른 멤버 함수에 의해서도 사용 될 수 있어야 함
- ✓ 클래스의 모든 private 멤버는 set, get 함수를 통해 접근해야 함 (다음주 수업 내용)

Good Programming Practice 3.6

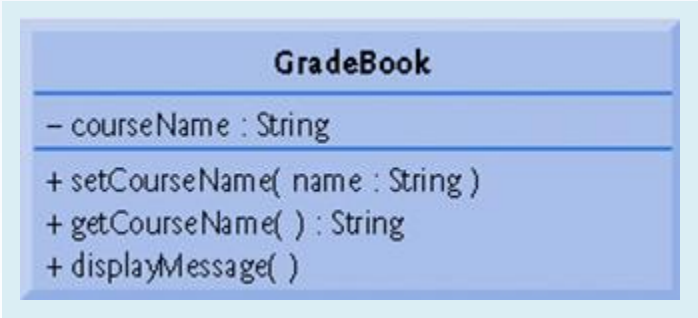
Always try to localize the effects of changes to a class's data members by accessing and manipulating the data members through their ***get*** and ***set*** functions. Changes to the name of a data member or the data type used to store a data member then affect only the corresponding ***get*** and ***set*** functions, but not the callers of those functions.

Software Engineering Observation 3.3

It is important to write programs that are understandable and easy to maintain. Change is the rule rather than the exception. Programmers should anticipate that their code will be modified.

UML diagram 작성방법

- ✓ 함수의 반환형은 함수이름과 ( )뒤에 : 을 붙이고 반환형을 명시
- ✓ - 기호는 private 멤버를, + 기호는 public 멤버를 의미



2주차 2교시 수업을 마칩니다.