객체 지향형 프로그래밍과 자료구조 (실습)

실습 2. (보충설명)
Class Date, Class Person

정보통신공학과
교수 김 영 탁
(Tel : +82-53-810-2497; E-mail : ytkim@yu.ac.kr)

Outline

- 객체 지향형 프로그래밍
  - 캡슐화
  - Abstraction: 접속과 구현의 분리
  - 정보보호 (information protection)
  - 정보은닉 (information hiding)
- Class Date
- Class Person
- Class Person의 응용 프로그램
- 실행결과
- Oral test

객체 지향형 프로그래밍 (Object-Oriented Programming) 개요

◆ 캡슐화 (Encapsulation)

- 데이터와 데이터를 처리하는 멤버 함수를 함께 포함하는 캡슐로 만들어 사용
- 멤버함수는 데이터가 항상 정상적인 범위 내에서 유지될 수 있도록 관리

◆ 데이터 추상화 (Data Abstraction)

- 클래스 내부에서 데이터를 처리하는 기능이 어떻게 구현되었는가에 대한 상세한 정보는 제공하지 않고, 단지 사용 방법에 대한 추상적인 정보만 제공
- 예) 자동차의 엔진 룸 내부가 어떻게 구현되어 있고, 어떻게 동작하는가에 대한 정보 없이도, 자동차 운전석에 있는 기기와 장치를 사용하여 운전할 수 있게 하는 것과 유사함

◆ 정보 보호 (Data Protection)

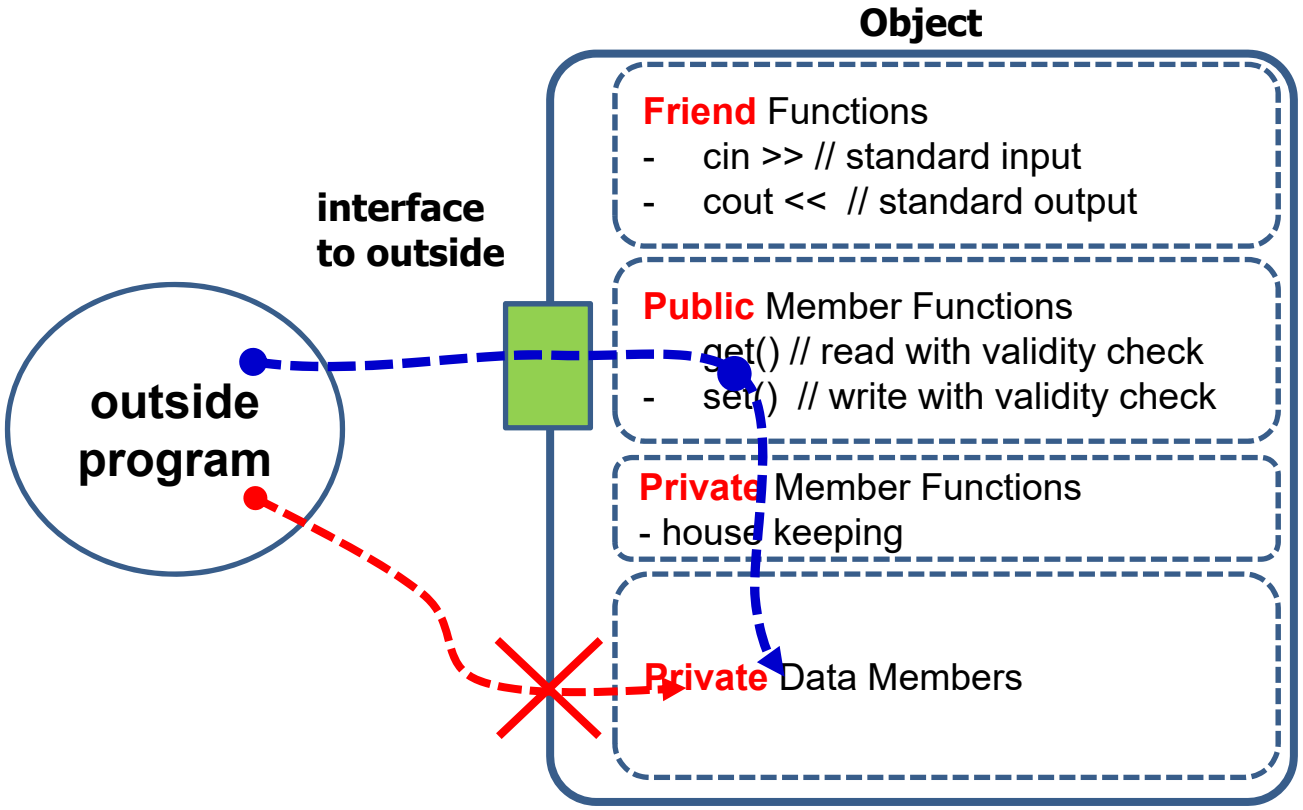
- 클래스 내부의 데이터를 외부 사용자가 직접 접속할 수 있도록 허용하지 않으며, 항상 외부로 공개된 멤버함수를 통하여 접속할 수 있도록 관리
- 정보가 비정상적으로 변경되는 것을 방지하며, 보호함

◆ 정보 은닉 (Information Hiding)

- 클래스 내부에서 데이터를 처리하는 기능을 구현하는 방법은 새로운 기술이 개발됨에 따라 계속 바뀔 수 있으며, 따라서 구현 기술에 대한 상세한 내용은 사용자에게 알려주지 않고, 은닉시킴
- 새로운 기술의 도입을 쉽게 할 수 있게 하며, 새로운 기술이 도입되어도 사용하는 방법에서는 변경이 없도록 함

객체 (Object)의 캡슐화 (Encapsulation)

◆ Encapsulation and protection of private data



캡슐화 (Encapsulation)

◆하나의 캡슐 안에 멤버 데이터와 멤버함수를 함께 포함

- 멤버 데이터: 사용자 정보를 포함, 각 데이터에는 정상적인 범위 (range of data)가 지정되어 있음
- 멤버 함수 : 데이터를 처리하는 기능을 구현

◆예) 사람에 관련된 정보

- int age; // 사람의 나이를 표현할 때, 정상적인 범위는 0 ~ 150
- 나이에 관련 데이터 연산(Operations):
  - +,-,*,/,%, logical, etc.
  - get_age()
  - set_age()

◆예) queue

- Data Element: 큐에 저장되는 개체 (숫자, 패킷 등)
- Operations: enqueue(), dequeue()

C++ 클래스의 기본 멤버 함수

◆ C++ 클래스의 기본 멤버 함수

클래스의 기본 멤버 함수	설 명
생성자 (constructor)	클래스를 사용하여 객체가 생성될 때 실행되며, 초기화 기능 수행
소멸자 (destructor)	클래스로 만들어진 객체가 소멸될 때 실행됨
데이터 멤버 접근자 (accessor)	클래스의 데이터 멤버 값을 읽기 위하여 접근하는 기능 제공. 예) getOOO()
데이터 멤버 변경자/ 설정자 (mutator)	클래스의 데이터 멤버 값을 변경하기 위한 기능 제공. 예) setOOO()

C++ 클래스 멤버 접근 권한 제한

◆ 클래스 멤버의 접근 권한 제한

클래스 멤버 접근 지정자	설 명
public	외부로 공개된 멤버이며, 주로 외부 공개 인터페이스 멤버 함수에 사용
private	외부에서는 직접 접근이 허용되지 않으며, 멤버 함수에게만 직접 접근이 허용됨. 주로 데이터 멤버는 private으로 접근을 제한하여 데이터 보호 기능을 제공함. 클래스 외부에서 private 데이터를 사용하기 위해서는 반드시 public으로 공개된 인터페이스를 거쳐야 함.
protected	이 클래스를 상속 받는 후손 (successor) 클래스들에게는 직접 접근이 제한적으로 허용됨.

Public and Private Members 구분을 통한 정보의 보호 (Information Protection)

◆클래스에 포함되는 사용자 데이터는 기본적으로 private 선언되어 보호됨

- Upholds principles of OOP
- 사용자가 직접 접속하는 것을 방지하며, 관련 멤버 함수를 통해서만 사용할 수 있게 하여, 정보를 보호함

◆Public으로 선언되는 항목은 외부에 알려지는 사항들

- 주로 사용자들에게 제공되는 API (application programming interface)들이 public member function으로 선언되어 공개됨
- public member function은 사용자들이 사용할 수 있도록 proto-type 형태로 공개됨
- Private Data Member는 항상 외부로 공개되어 있는 public member function을 통하여 접근될 수 있게 함으로써 정보를 보호하며, 내부 데이터가 항상 정상적인 범위에서 유지될 수 있게함

Interface (외부 접속)와 Implementation (구현)의 분리, 정보 은닉 (information hiding)

◆ Interface (외부 접속)와 Implementation (구현)의 분리가 왜 필요한가?

- 하나의 기능은 다양한 기술로 구현될 수 있음 (예: 자동차 엔진을 가솔린, 디젤, 전기모터, 수소연료 방식으로 구현할 수 있음)
- 자동차 엔진이 여러 가지 다른 방식으로 구현되어도, 실제 이를 사용하는 운전석의 기본 기능은 동일함 (핸들, 기어변속, 브레이크, 엑셀레이터 등)
- 현재 사용가능 한 기술로 구현되어 있는 모듈을 향후, 새로운 구현기술을 사용하여 더 저렴한 비용으로 구현할 수 있음

◆ 상세 구현 정보 은닉 (Information Hiding)

- Interface (외부 접속)와 Implementation (구현)을 분리시킴으로써, 외부 접속 (interface)는 그대로 유지한 채, 내부적인 구현 방법만 변경할 수 있게 함
- 새로운 기술을 쉽게 도입할 수 있는 장점을 제공함

class Date 선언

```
/* Date.h (1) */
#ifndef DATE_H
#define DATE_H

#include <iostream>
#include <fstream>
#include <cstdlib>
using namespace std;

#define WEEKDAY_AD01Jan01 MON // the weekday of AD Jan 1.
#define DAYS_PER_WEEK 7
#define Secs_in_Minute 60
#define Secs_in_Hour (Secs_in_Minute * 60)
#define Secs_in_DAY (Secs_in_Hour * 24)
#define LOCAL_GMT_OFFSET_HOUR 9

class Date
{
public:
      Date();  // default constructor
      Date(int y, int m, int d); // constructor
      ~Date(); // destructor
```

```
/* Date.h (2) */

    ~Date(); // destructor
    void inputDate();
    void fprintDate(ostream& fout);
    void setDate(int y, int m, int dy);
    void setRandDateAttributes();
    void setMonth(int m);
    void setYear(int y);
    int getYear() { return year; }
    int getMonth() { return month; } //Returns 1 for January, 2 for February, etc.
    int getDay() { return day; }
    int getYearDay();
    int getYearDay(int m, int d);
    int getWeekDay();
    int getElapsedDaysFromAD010101(); //  get elapsed days from AD 1. 1. 1.
    int getElapsedDaysFromAD010101(Date);
    void fprintCalendar_Month(ostream& fout);
private:
    bool isLeapYear(); // check whether the year is leap year
    bool isLeapYear(int y); // check whether the given year y is leap year
    bool isValidDate(int y, int m, int d);
    int year;
    int month;
    int day;
};

#endif
```

class Date 멤버함수

```
/* Date.cpp  (1) */

#include <iostream>
#include <string>
#include <iomanip>
#include "Date.h"

enum WEEKDAY { SUN, MON, TUE, WED, THR, FRI, SAT };
enum MONTH { JAN = 1, FED, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC, NUM_MONTHS };
const char *weekDayName[DAYS_PER_WEEK] = { "Sunday", "Monday", "Tuesday", "Wednesday",
    "Thursday", "Friday", "Saturday" };
const char *weekDayNameShort[DAYS_PER_WEEK] = { "SUN", "MON", "TUE", "WED", "THR", "FRI",
    "SAT" };
const char *monthName[13] = { "", "January", "February", "March", "April", "May", "June", "July",
    "August", "September", "October", "November", "December" };

//default constructor
Date::Date()
{
    year = 0;
    month = 0;
    day = 0;
}
```

```
/* Date.cpp  (2) */

//default destructor
Date::~Date()
{
    // cout << "Date object instance is destructed" << endl;
}

bool Date::isValidDate(int y, int m, int d)
{
    int days_month[13] = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
    if (isLeapYear(y))
        days_month[2] = 29;

    if ((m >= 1) && (m <= 12)  &&  (d >= 1) && (d <= days_month[m]))
    {
        return true;
    } else {
        cout << "Illegal date! (" << m << ", " << d << ") ==>  Program aborted.\n";
        return false;
    }
}
```

```
/* Date.cpp  (3) */

void Date::setDate(int y, int m, int d)
{
    if (isValidDate(y, m, d))
    {
      year = y;
      month = m;
      day = d;
    } else {
      cout << "Invalid date (" << y << ", " << m << ", " << d << ")";
      cout << " Program aborted !!\n";
      exit(1);
    }
}

void Date::setRandDateAttributes()
{
    int days_month[13] = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

    year = rand() % 2000 + 1000;
    month = rand() % 12 + 1;
    if (isLeapYear(year) && month == 2)
        days_month[2] = 29;
    day = rand() % days_month[month] + 1;
}
```

```
/* Date.cpp  (4) */

void Date::setMonth(int newMonth)
{
    if ((newMonth >= 1) && (newMonth <= 12))
        month = newMonth;
    else
    {
      cout << "Illegal month value! Program aborted.\n";
      exit(1);
    }
    day = 1;
}

void Date::setYear(int y)
{
    year = y;
}

int Date::getYearDay()
{
    int days_month[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
    int yearDay = 0;

      . . . . .
    return yearDay;
}

int Date::getYearDay(int month, int day)
{
    . . . . .
}
```

```
/* Date.cpp  (6) */

int Date::getElapsedDaysFromAD010101()
{
    int elpsDay = 0;
    elpsDay = getElapsedDaysFromAD010101(*this);

    return elpsDay;
}

int Date::getElapsedDaysFromAD010101(Date d)
{
    int yearDay;
    int elpsDay = 0;

    for (int y = 1; y < d.year; y++)
    {
        if (isLeapYear(y))
            elpsDay += 366;
        else
            elpsDay += 365;
    }
    yearDay = getYearDay(d.month, d.day);
    elpsDay += yearDay;

    return elpsDay;
}
```

```
/* Date.cpp  (7) */

int Date::getWeekDay()
{
    int weekDay_AD010101 = MON; // 1. 1. 1. is Monday
    int weekDay;
    int elapsedDays = 0;

    elapsedDays = getElapsedDaysFromAD010101();
    weekDay = (elapsedDays + weekDay_AD010101 - 1) % 7;

    //cout << ", Elapsed days from AD Jan. 1, 1 (" << elapsedDays << ")";

    return weekDay;
}



void Date::inputDate()
{
    int y, m, d;

    cout << "Enter date in year month day : ";
    cin >> y >> m >> d;
    if (isValidDate(y, m, d))
    {
       year = y;
       month = m;
       day = d;
    } else {
        cout << "Ilegal date! Program aborted.\n";
        exit(1);
    }
}
```

```
/* Date.cpp  (8) */

bool Date::isLeapYear(int y)
{
    . . . . .
}

bool Date::isLeapYear()
{
    return isLeapYear(year);
}

void Date::fprintDate(ostream& fout)
{
    const char *weekDayName[7] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",
    "Friday",  "Saturday"};
    const char *monthName[13] = { "", "January", "February", "March", "April", "May", "June", "July",
         "August", "September", "October", "November", "December" };
    int yearDay = 0;
    int weekDay;

    if ((month >= 1) && (month <= 12))
    fout << setw(10) << string(monthName[month]);
    fout << " " << setw(2) << day << ", " << setw(4) << year;

    yearDay = getYearDay();
    weekDay = getWeekDay();
    fout << " (" << setw(10) << string(weekDayName[weekDay]) << ")";
}
```

시간 관련 함수

분류	함수 원형과 인자	함수 설명
시간 계산	time_t time(time_t *timeptr);	1970년 1월 1일 자정부터 경과된 현재 시간을 초단위로 계산
시간을 문자열로 변환	char *asctime(strcut tm *time);	구조체 tm형식의 시간을 문자열로 변환
	char *ctime(time_t *time);	함수 time()로부터 계산된 현재 시간을 문자열로 변환
시간을 구조체로 변환	struct tm *localtime(time_t *time);	지역 시간(local time)을 구조체 tm의 형식으로 가져오는 함수
	struct tm *gmtime(time_t *time);	Greenwich Mean Time(GMT)을 구조체 tm 형식으로 가져옴
시간 차이 계산	clock_t clock(void);	clock tick으로 경과된 시간
	double difftime(time_t time2, time_t time1);	두 시간의 차이를 초단위로 계산
시간 지연	void Sleep(unsigned millisecond); void delay(unsigned millisecond);	인자가 지정하는 만큼의 밀리초 (1/1000초) 단위의 시간을 지연

시간 관련 함수의 사용 예

```
/* main() for Date_and_Time.c */

#include <stdio.h>
#include <time.h>

void main()
{
      time_t  currentTime;   // time_t 구조체 변수
      struct tm  *info;    // tm 구조체 포인터

      time(& currentTime);
      info = localtime(& currentTime);
      printf("Current local time and data: %s", asctime(info));
}
```

```
Current local time and date: Sat Mar 24 14:12:21 2018
계속하려면 아무 키나 누르십시오 . . .
```

class Person

```
/* Person.h */
#ifndef PERSON_H
#define PERSON_H
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string>
#include "Date.h"
using namespace std;

#define MAX_NAME_LEN 15
#define MIN_NAME_LEN 5
class Person
{
public:
    Person() : name(string("nobody")), birthDate(Date(1, 1, 1)) { }
    Person(string n, Date bd) : name(n), birthDate(bd) { }
    void setName(string n) { name = n; }
    string getName() { return name; }
    void setBirthDate(Date bd) { birthDate = bd; }
    Date getBirthDate() { return birthDate; }
    void setRandPersonAttributes();
    void fprintPerson(ostream& fout);
private:
    Date birthDate;
    string name;
};
#endif
```

class Person 멤버함수 구현

```
/* Person.cpp */
#include "Person.h"

void Person::setRandPersonAttributes()
{
    char str[MAX_NAME_LEN + 1];
    int name_len, i = 0;
    birthDate.setRandDateAttributes();
    name_len = rand() % (MAX_NAME_LEN - MIN_NAME_LEN) + MIN_NAME_LEN;
    str[0] = rand() % 26 + 'A';
    for (i = 1; i<name_len; i++)
      str[i] = rand() % 26 + 'a';
    str[i] = '\0';
    name = string(str);
}

void Person::fprintPerson(ostream& fout)
{
    fout << " Person [name: ";
    fout.setf(ios::left);
    fout << setw(16) << name;
    fout << ", birth date: ";
    fout.unsetf(ios::left);
    birthDate.fprintDate(fout);
    fout << "]";
}
```

main() 함수

```
/* main.cpp (1) */
#include <iostream>
#include <fstream>
#include <time.h>
#include <string>
#include "Date.h"
#include "Person.h"
using namespace std;

#define NUM_PERSON 10

int main()
{
    ofstream fout;
    fout.open("output.txt");
    if (fout.fail())
    {
        cout << "Error in opening output.txt !!" << endl;
        exit;
    }
```

```
/* main.cpp (2) */

    /* part 1 - handling class Date */
    Date AD010101(1, 1, 1);
    int year, month, day;
    int daysToChristmas;
    time_t currentTime;
    struct tm *time_and_date;

    time(&currentTime);
    time_and_date = localtime(&currentTime);
    year = time_and_date->tm_year + 1900;
    month = time_and_date->tm_mon + 1;
    day = time_and_date->tm_mday;
    Date newYearDay(year, 1, 1), today(year, month, day);

    fout << "AD Jan. 1, 1 is ";
    AD010101.fprintDate(fout);
    fout << endl;

    fout << "Today is ";
    today.fprintDate(fout);
    fout << endl;

    fout << "New year's day of this year was ";
    newYearDay.fprintDate(fout);
    fout << endl;
```

```
/* main.cpp (3) */

    Date christmas(year, 12, 25);
    fout << "Christmas of this year is ";
    christmas.fprintDate(fout);
    fout << endl;

    if (today.getMonth() == christmas.getMonth() && today.getDay() == christmas.getDay())
    {
        fout << "Today is Christmas!  Happy Christmas to you all !!\n";
    }
    else {
        fout << " Sorry, today is not Christmas!\n";
        daysToChristmas = christmas.getElapsedDaysFromAD010101()
                - today.getElapsedDaysFromAD010101();
        fout << " You must wait " << daysToChristmas << " day(s) to Christmas !" << endl;
    }
    fout << endl;
```

```
/* main.cpp (4) */

    /* part 2 - handling class Person */
    Person p1(string("Hong, Gil-Dong"), Date(2000, 1, 1)), p2;
    fout << "Person p1 : " << endl;
    p1.fprintPerson(fout);
    fout << endl;

    fout << "Person p2 : " << endl;
    p2.setName(string("Lee, Soo-Jeong"));
    p2.setBirthDate(Date(2018, 9, 1));
    p2.fprintPerson(fout);
    fout << endl;

    fout << endl << "Generating array of persons ... " << endl;
    Person *persons;
    persons = (Person *) new Person[NUM_PERSON];
    for (int i = 0; i < NUM_PERSON; i++)
    {
        persons[i].setRandPersonAttributes();
        persons[i].fprintPerson(fout);
        fout << endl;
    }
    fout << endl;

    delete[] persons;
    fout.close();

    return 0;
}
```

실행 결과

```
AD Jan. 1, 1 is    January  1,    1 (    Monday)
Today is  September  4, 2021 (  Saturday)
New year's day of this year was    January  1, 2021 (    Friday)
Christmas of this year is   December 25, 2021 (  Saturday)
 Sorry, today is not Christmas!
 You must wait 112 day(s) to Christmas !

Person p1 :
 Person [name: Hong, Gil-Dong  , birth date:    January  1, 2000 (  Saturday)]
Person p2 :
 Person [name: Lee, Soo-Jeong  , birth date:  September  1, 2018 (  Saturday)]

Generating array of persons ...
 Person [name: Humea           , birth date:   December 11, 1041 (  Saturday)]
 Person [name: Fdxfir          , birth date:       June  6, 1464 (    Monday)]
 Person [name: Xggbwkf         , birth date:      April 25, 2436 (    Friday)]
 Person [name: Xwfnfozvsrtkj   , birth date:       July 16, 2447 (   Tuesday)]
 Person [name: Ggxrpnrvystm    , birth date:       June 29, 2711 (  Thursday)]
 Person [name: Yycqp           , birth date:    January  6, 2842 (    Monday)]
 Person [name: Effmz           , birth date:    October  5, 2890 (  Thursday)]
 Person [name: Kasvwsrenzk     , birth date:    January 22, 1623 (    Sunday)]
 Person [name: Xtlsgy          , birth date:       July 21, 1118 (    Sunday)]
 Person [name: Dpooefxzbco     , birth date:      March 23, 2977 (    Sunday)]
```

Oral Test

Q 2.1 C++ 객체 지향형 프로그래밍에서 캡슐화 (encapsulation)이란 무엇이며, 어떤 장점이 있는가를 예를 들어 설명하라. (핵심포인트: 캡슐에 포함되는 항목, 캡슐화에 따른 데이터 추상화, 정보 은닉, 정보 보호)

Q 2.2 C++ 객체 지향형 프로그래밍에서 외부 접속 (external interface)과 내부 구현 (internal implementation)을 분리한다는 것이 무엇이며 어떤 장점이 있는가를 예를 들어 설명하라. (핵심포인트: 왜 접속과 구현을 분리하는 것이 좋은가?)

Q 2.3 C++ 객체 지향형 프로그래밍에서 정보 보호 (information protection)은 어떻게 하는지 예를 들어 설명하라. (핵심 포인트: 정보보호를 어떻게 하는가?)

Q 2.4 C++ 객체 지향형 프로그래밍에서 정보 은닉 (information hiding)은 어떻게 하며, 왜 필요한가에 대하여 예를 들어 설명하라. (핵심 포인트: 정보은닉을 어떻게 하며, 왜 해야 하는가?)