객체지향형 프로그래밍과 자료구조

Ch 2. C++ 클래스 기본 구조

정보통신공학과
교수 김 영 탁

(Tel : +82-53-810-2497; E-mail : ytkim@yu.ac.kr)

Outline

◆Struct (구조체)

- Struct
- Array of Struct (구조체 배열)
- Self-referential structure (자기참조 구조체)
- List Node
- Tree Node

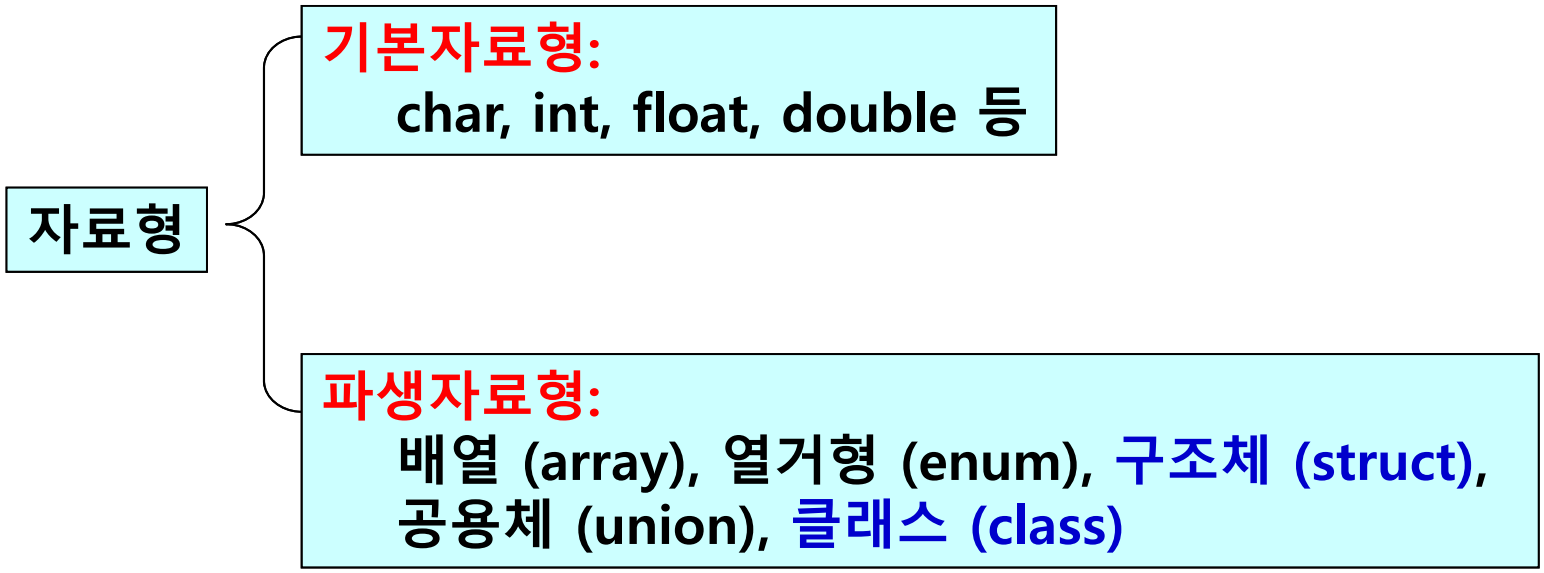
◆Class (클래스)

- Defining, member functions (멤버함수 정의)
- Public and private members
- Accessor (i.e., get_OOO()) and mutator (i.e., set_OOO())functions
- Structures vs. classes

Structure (구조체)

자료형의 분류

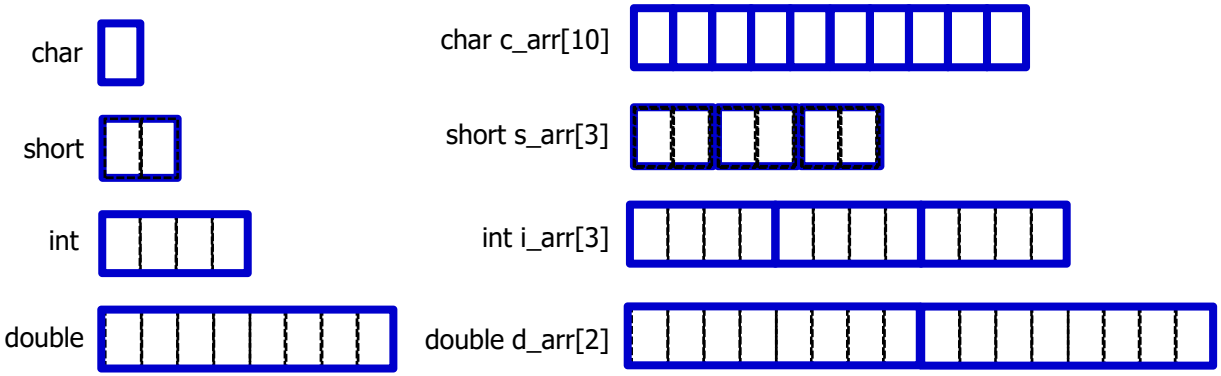
◆ Data Types



배열, 구조체, 구조체 배열 (1)

◆ 배열과 구조체

- 배열은 동일한 자료형의 원소들을 가지며, 연속적인 메모리 공간을 사용: 예 정수형 배열, 더블형 배열
- 구조체는 다양한 자료형의 원소들을 가지며, 연속적인 메모리 공간을 사용: 학생 (int 학번, char [] 이름, int 학년, double 성적)
- 기본 자료형의 배열: 동일한 자료형의 원소들을 배열로 구성



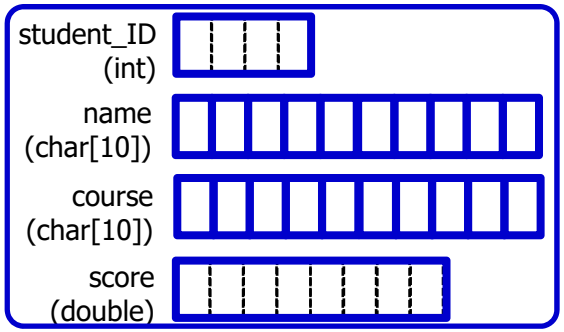
(a) 기본 자료형 (b) 기본자료형의 배열

배열, 구조체, 구조체 배열 (2)

◆ 구조체 배열

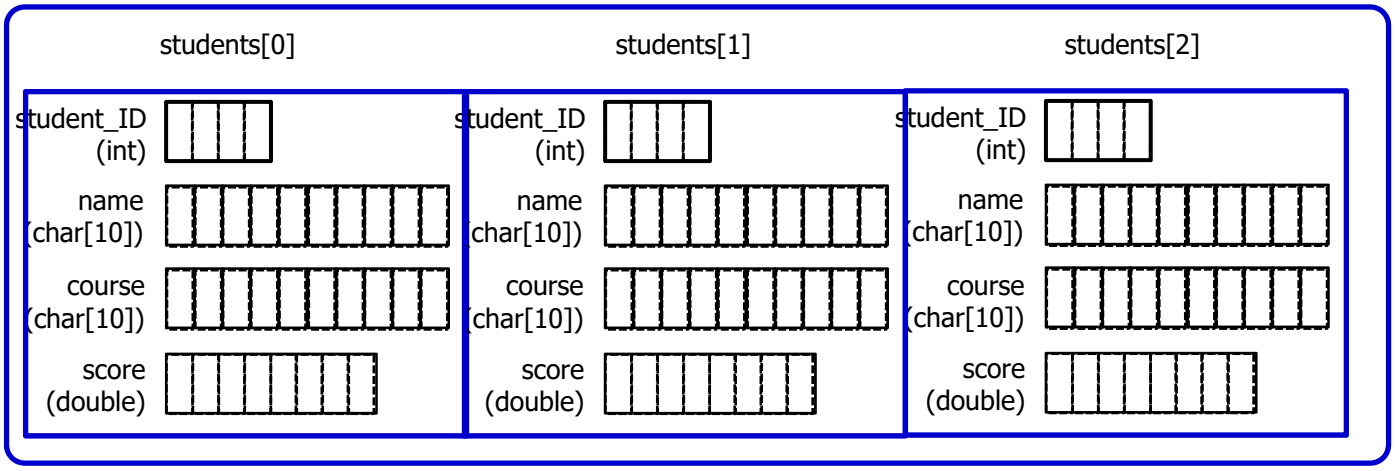
- 구조체를 원소로 배열 구성

```
typedef struct
{
    int student_ID;
    char name[10];
    char course[10];
    double score;
} Student;
```



(c) 구조체

```
Student
students[3];
```



(d) 구조체 배열

구조체의 예 (1)

typedef struct {
  int year;
  int month;
  int day;
} **Date**;
	typedef struct {
	  int hour;
	  int min;
	  int sec;
	} **Time**;
		typedef struct {
		  double real;
		  double imagin;
		} **Cmplx**; // Complex

typedef struct {
  int x;
  int y;
} **Coordinate**;
	typedef struct {
	  int width;
	  int length;
	} **Rectangle**;
		typedef struct {
		  int width;
		  int length;
		  int height;
		} **Hexahedron**;

typedef struct {
  int base;
  int height;
} **Triangle**;
	typedef struct {
	  int radius;
	} **Circle**;
		typedef struct {
		  int radius;
		  int height;
		} **Cylinder**;

구조체의 예 (2)

```
typedef struct {
   int planet_id;
   char name[20];
   double dist_sun;
   double mass;
   double radius;
} Planet;
```

```
typedef struct {
   int nationCode;
   int region;
   int switch;
   int line;
} TelNum;
```

```
typedef struct {
   int isbn;
   string bookTitle;
   string authorName;
   Date publishDate;
} Book;
```

```
typedef struct {
   string name;
   Date birthDate;
   int registrationID;
   string address;
} Person;
```

```
typedef struct {
   int student_id;
   char name[20];
   Date birthDate;
   char couse_name[20];
   double grades;
} Student;
```

```
typedef struct {
   int staff_id;
   char name[20];
   int dept_code;
   Date birthDate;
   double salary;
} Staff;
```

구조체 변수의 선언

◆ 구조체를 자료형과 같이 사용하여 변수 선언

◆ **Cmplx c1, c2;**

- Just like declaring simple types
- Variable **c1** and **c2** are now of type **Cmplx**
- They contain "member values " real and imagin
  - Each of the struct "parts"

구조체를 사용하는 프로그램의 예

```
typedef struct {
    double real;
    double imagin;
} Cmplx;  // 구조체 복소수

void printCmplx(Cmplx c)
{
    char sign;
    double mag;

    sign = (c.imagin < 0) ? '-' : '+';
    cout << setw(8) << c.real <<" " << sign << " j" << fabs(c.imagin) ;
    mag = sqrt(pow(c.real, 2.0) + pow(c.imagin, 2.0) );
    cout << "  (magnitude " << mag << ")" << endl;
}

Cmplx cmplxAdd(Cmplx c1, Cmplx c2)
{
    Cmplx res;
    res.real = c1.real + c2.real;
    res.imagin = c1.imagin + c2.imagin;
    return res;
}
```

Cmplx cmplxSub(Cmplx c1, Cmplx c2)

```
{
    Cmplx res;
    res.real = c1.real - c2.real;
    res.imagin = c1.imagin - c2.imagin;
    return res;
}
```

Cmplx cmplxMultiply(Cmplx c1, Cmplx c2)

```
{
    Cmplx res;
    res.real = (c1.real * c2.real) - (c1.imagin * c2.imagin);
    res.imagin = (c1.real * c2.imagin) + (c1.imagin * c2.real);
    return res;
}
```

Cmplx cmplxDivide(Cmplx c1, Cmplx c2)

```
{
    Cmplx res;
    double denom;
    denom = c2.real*c2.real + c2.imagin*c2.imagin;
    res.real = ((c1.real * c2.real) + (c1.imagin * c2.imagin)) / denom;
    res.imagin = ((c2.real * c1.imagin) - (c1.real * c2.imagin)) / denom;
    return res;
}
```

```
int main()
{
    Cmplx cmplxs[7];

    cout << "Input two complex numbers : c1.real, c1.imagin,
                c2.real, c2.imagin"<<endl;
    cout << "c1.real : ";
    cin >> cmplxs[0].real;
    cout << "c1.imagin : ";
    cin >>  cmplxs[0].imagin;

    cout << "c2.real : ";
    cin >> cmplxs[1].real;
    cout << "c2.imagin : ";
    cin >> cmplxs[1].imagin;
    cmplxs[2] = cmplxAdd(cmplxs[0], cmplxs[1]);
    cmplxs[3] = cmplxSub(cmplxs[0], cmplxs[1]);
    cmplxs[4] = cmplxMultiply(cmplxs[0], cmplxs[1]);
    cmplxs[5] = cmplxDivide(cmplxs[0], cmplxs[1]);
    cmplxs[6] = cmplxs[0];

    for (int i=0; i<7; i++) {
        cout << "cmplxs[" << i << "]: ";
        printCmplx(cmplxs[i]);
    }
    return 0;
}
```

```
Input two complex numbers : c1.real, c1.imagin, c2.real, c2.imagin
c1.real : 3.5
c1.imagin : 4.7
c2.real : 2.1
c2.imagin : -7.3
cmplxs[0]:      3.5 + j4.7  (magnitude 5.86003)
cmplxs[1]:      2.1 - j7.3  (magnitude 7.59605)
cmplxs[2]:      5.6 - j2.6  (magnitude 6.17414)
cmplxs[3]:      1.4 + j12  (magnitude 12.0814)
cmplxs[4]:    41.66 - j15.68  (magnitude 44.5131)
cmplxs[5]: -0.467244 + j0.613865  (magnitude 0.771458)
cmplxs[6]:      3.5 + j4.7  (magnitude 5.86003)
```

구조체 변수의 대입 (assignment) 연산

◆ 구조체

```
typedef struct {
    double real;
    double imagin;
} Cmplx;
```

◆**Declare two structure variables:**

```
Cmplx c1, c2;
```

- Both are variables of "type of struct Cmplx"
- Simple assignments are legal:

  c2 = c1;
  - Simply copies each member variable from c1 into member variables of c2

함수 호출에서의 구조체 사용

◆ 함수호출에서 구조체 변수를 인수로 전달

- Pass-by-value
- Pass-by-reference
- Or combination

◆ 함수 실행결과의 반환값을 구조체 변수로 전달

- Return-type is structure type
- Return statement in function definition sends structure variable back to caller

구조체 변수의 초기화

◆구조체 변수의 초기화

Example:

```
typedef struct {
    char name[10];
    double relativeMass;
    double distance;
} Planet;

Planet earth = {"Earth", 1.0, 150};
```

- Declaration provides initial data to all three member variables

구조체 배열

- 구조체 변수들을 배열로 구성
- **Example)**

```
Planet solarSystem[SOLAR_PLANETS] =
    {   {"Mercury", 0.0558, 57.9},
        {"Venus", 0.815, 108},
        {"Earth",1.0, 150},
        {"Mars", 0.107, 228},
        {"Jupiter", 318, 778},
        {"Saturn", 95.1, 1430},
        {"Uranus", 14.5, 2870},
        {"Neptune", 17.2, 4500},
        {"Pluto", 0.11, 5900}
    };
```

Handling attributes of each element of struct array

```
void printPlanets(Planet solarPlanets[], int num_planet)
{
    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);

    for (int i=0; i<num_planet; i++) {
        cout << setw(2) << i;
        cout << "  Name: ";
        cout << setiosflags(ios::left) <<setw(8) << solarPlanets[i].name;
        cout << resetiosflags(ios::left);
        cout << " Rel Mass: ";
        cout << setw(7) <<  setprecision(3) <<solarPlanets[i].relativeMass;
        cout << " Dist from Sun: ";
        cout << setw(6) << setprecision(1) << solarPlanets[i].distance ;
        cout << endl;
    }  // end for
}
```

Sorting Array of Structure

```
void sortPlanets_byRelMass(Planet solarPlanets[], int num_planet)
{
    Planet temp;
    int i, j, m;
    double min_RelMass;
    for (i=0; i<num_planet-1; i++) {
        m = i;
        min_RelMass = solarPlanets[i].relativeMass;
        for (j=i+1; j<num_planet; j++) {
            if (min_RelMass > solarPlanets[j].relativeMass) {
                m = j;
                min_RelMass = solarPlanets[j].relativeMass;
            }
        } // end inner for
        if (m != i) { // if new minimum found, swap
            temp = solarPlanets[i];
            solarPlanets[i] = solarPlanets[m];
            solarPlanets[m] = temp;
        }
    } // end outer for
}
```

```
void sortPlanets_byName(Planet solarPlanets[], int num_planet)
{
    Planet temp;
    int i, j, m;
    char min_Name[10] = {0};
    for (i=0; i<num_planet-1; i++)
    {
        m = i;
        strcpy(min_Name, solarPlanets[i].name);
        for (j=i+1; j<num_planet; j++) {
            if (strcmp(min_Name, solarPlanets[j].name) > 0)
            {
                m = j;
                strcpy(min_Name, solarPlanets[j].name);
            }
        } // end inner for
        if (m != i) { // if new minimum found, swap
            temp = solarPlanets[i];
            solarPlanets[i] = solarPlanets[m];
            solarPlanets[m] = temp;
        }
    } // end outer for
}
```

```
int main()
{
    Planet solarSystem[SOLAR_PLANETS] =
   {  {"Mercury", 0.0558, 57.9}, {"Venus", 0.815, 108}, {"Earth",1.0, 150},
      {"Mars", 0.107, 228}, {"Jupiter", 318, 778}, {"Saturn", 95.1, 1430},
      {"Uranus", 14.5, 2870}, {"Neptune", 17.2, 4500}, {"Pluto", 0.11, 5900}  };

    cout << "Initial state" << endl;
    printPlanets(solarSystem, SOLAR_PLANETS);

    sortPlanets_byRelMass(solarSystem, SOLAR_PLANETS);
    cout << "After sorting by relative mass:" << endl;
    printPlanets(solarSystem, SOLAR_PLANETS);

    sortPlanets_byDist(solarSystem, SOLAR_PLANETS);
    cout << "After sorting by distance from sun:" << endl;
    printPlanets(solarSystem, SOLAR_PLANETS);

    sortPlanets_byName(solarSystem, SOLAR_PLANETS);
    cout << "After sorting by name using strcmp and strcpy:" << endl;
    printPlanets(solarSystem, SOLAR_PLANETS);
    cout << endl;

    return 0;
}
```

◆ result of execution

```
Initial state
0  Name: Mercury  Rel Mass:   0.056 Dist from Sun:   57.9
1  Name: Venus    Rel Mass:   0.815 Dist from Sun:  108.0
2  Name: Earth    Rel Mass:   1.000 Dist from Sun:  150.0
3  Name: Mars     Rel Mass:   0.107 Dist from Sun:  228.0
4  Name: Jupiter  Rel Mass: 318.000 Dist from Sun:  778.0
5  Name: Saturn   Rel Mass:  95.100 Dist from Sun: 1430.0
6  Name: Uranus   Rel Mass:  14.500 Dist from Sun: 2870.0
7  Name: Neptune  Rel Mass:  17.200 Dist from Sun: 4500.0
8  Name: Pluto    Rel Mass:   0.110 Dist from Sun: 5900.0

After sorting by relative mass:
0  Name: Mercury  Rel Mass:   0.056 Di
1  Name: Mars     Rel Mass:   0.107 Di
2  Name: Pluto    Rel Mass:   0.110 Di
3  Name: Venus    Rel Mass:   0.815 Di
4  Name: Earth    Rel Mass:   1.000 Di
5  Name: Uranus   Rel Mass:  14.500 Di
6  Name: Neptune  Rel Mass:  17.200 Di
7  Name: Saturn   Rel Mass:  95.100 Di
8  Name: Jupiter  Rel Mass: 318.000 Di

After sorting by distance from sun:
0  Name: Mercury  Rel Mass:   0.056 Dist from Sun:   57.9
1  Name: Venus    Rel Mass:   0.815 Dist from Sun:  108.0
2  Name: Earth    Rel Mass:   1.000 Dist from Sun:  150.0
3  Name: Mars     Rel Mass:   0.107 Dist from Sun:  228.0
4  Name: Jupiter  Rel Mass: 318.000 Dist from Sun:  778.0
5  Name: Saturn   Rel Mass:  95.100 Dist from Sun: 1430.0
6  Name: Uranus   Rel Mass:  14.500 Dist from Sun: 2870.0
7  Name: Neptune  Rel Mass:  17.200 Dist from Sun: 4500.0
8  Name: Pluto    Rel Mass:   0.110 Dist from Sun: 5900.0

After sorting by name using strcmp and strcpy:
0  Name: Earth    Rel Mass:   1.000 Dist from Sun:  150.0
1  Name: Jupiter  Rel Mass: 318.000 Dist from Sun:  778.0
2  Name: Mars     Rel Mass:   0.107 Dist from Sun:  228.0
3  Name: Mercury  Rel Mass:   0.056 Dist from Sun:   57.9
4  Name: Neptune  Rel Mass:  17.200 Dist from Sun: 4500.0
5  Name: Pluto    Rel Mass:   0.110 Dist from Sun: 5900.0
6  Name: Saturn   Rel Mass:  95.100 Dist from Sun: 1430.0
7  Name: Uranus   Rel Mass:  14.500 Dist from Sun: 2870.0
8  Name: Venus    Rel Mass:   0.815 Dist from Sun:  108.0
```

Struct Date

◆ Example of struct

```
typedef struct
{
   int year;
   int month;
   int day;
} Date;

Date d1, d2;

d1.year = 2013;
d1.month = 9;
d1.day = 2;

d2.year = -999; // no compilation error; but is this logically correct ?
d2.month = 500; // no compilation error; but is this logically correct ?
d2.day = -45; // no compilation error; but is this logically correct ?
```

C 프로그램과 구조체의 제한점

◆ C 프로그램과 구조체의 제한점

- 구조체로 표현된 파생 자료형의 데이터 값이 정상적인 범위에 있는가를 보장하는 방법이 없음
  - 예) 나이 (age): 정상적인 범위 1 ~ 150
- 구조체와 관련함수의 통합관리 기능이 없음
- 구조체 데이터와 관련함수의 재 사용성 부족
- C 프로그램의 함수 오버로딩 (function overloading)과 연산자 오버로딩 (operator overloading) 기능 부족
- 자료구조와 알고리즘의 추상화를 위한 템플릿 개념 없음
- 데이터 접근자 (accessor)의 접근권한 제한 지정기능 없음
- 외부 공개 인터페이스와 내부 구현 방법의 분리 개념 없음

C++ Class

큰 규모의 소프트웨어 개발에서 필요한 핵심 기능

◆ Fundamentally required skill for large-scale system design and implementations

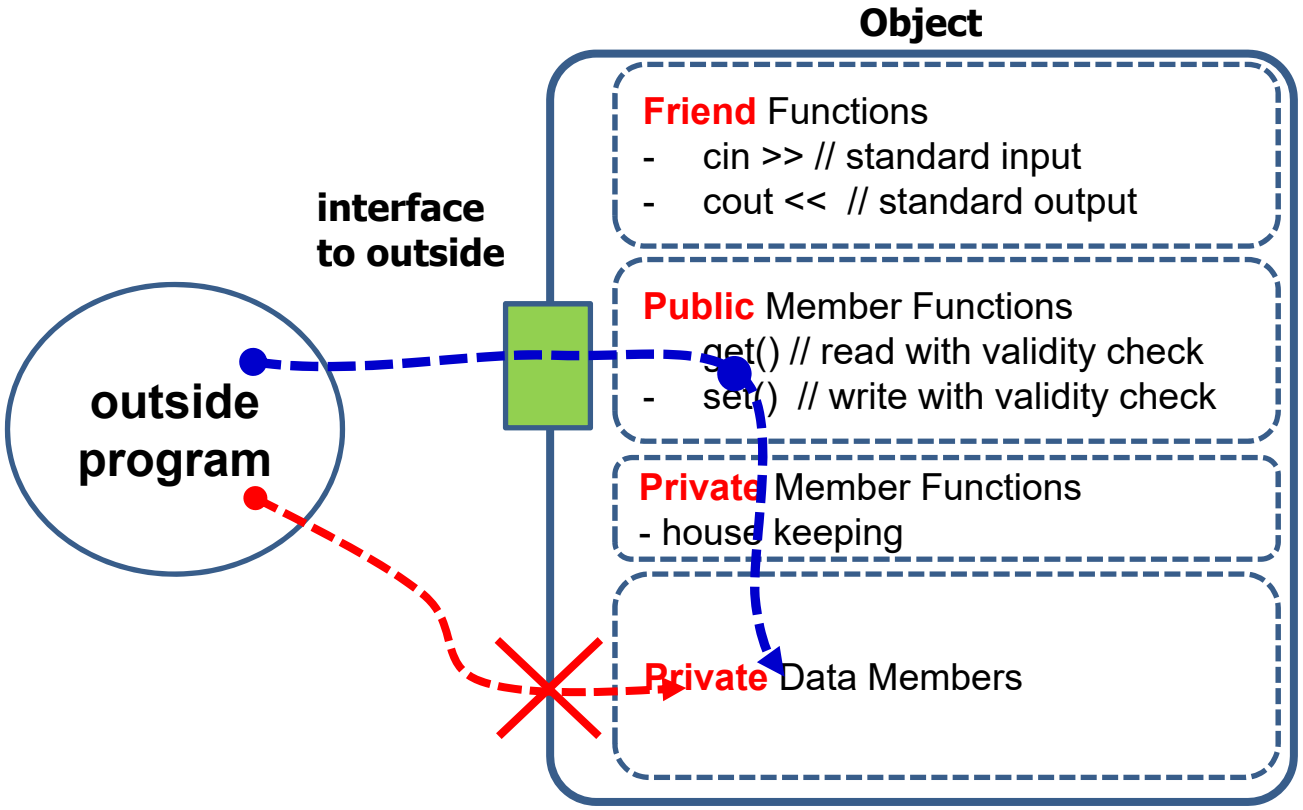
- **System stability** (시스템 안정성): 시스템이 항상 정상적인 범위/상태에 존재하도록 유지.
- **Software re-usability** (소프트웨어 재사용성): 기존에 개발된 소프트웨어 모듈을 재사용함으로써 쉽게 새로운 시스템 개발. 시스템 개발 기간과 비용을 절감.
- **User-friendly Interface** (사용자 친화형 접속): 사용자가 쉽게 이해하고, 사용할 수 있는 인터페이스
- **System expandability** (시스템 확장성): 새로운 최신 기술이 개발되었을 때, 이를 쉽게 수용할 수 있는 확장성.
- **System Manageability** (시스템 관리 용이성): 시스템을 쉽게 관리할 수 있는 구조

객체 지향형 프로그래밍 (Object-Oriented Programming) 개요

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

객체 (Object)의 캡슐화 (Encapsulation)

◆ Encapsulation and protection of private data



C++ 클래스 선언 예

```
/* Planet.h */

class Planet
{
public:
    Planet(string nm, double rM, double dist); // 생성자 (constructor)
    Planet(); // 기본 생성자 (default constructor)
    ~Planet(); // 소멸자 (destructor)
    string getName() { return name; }  // inline 멤버함수
    void setName(string nm) { name = nm; }
    double getRelMass() { return relativeMass; }
    void setRelMass(double rM) { relativeMass = rM; }
    double getDist() { return distance; }
    void setDist(double dist) { distance = dist; }
    void fprint(ostream& fout);
private:
    string name; // 데이터 멤버
    double relativeMass; // 데이터 멤버
    double distance; // 데이터 멤버
};
```

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()

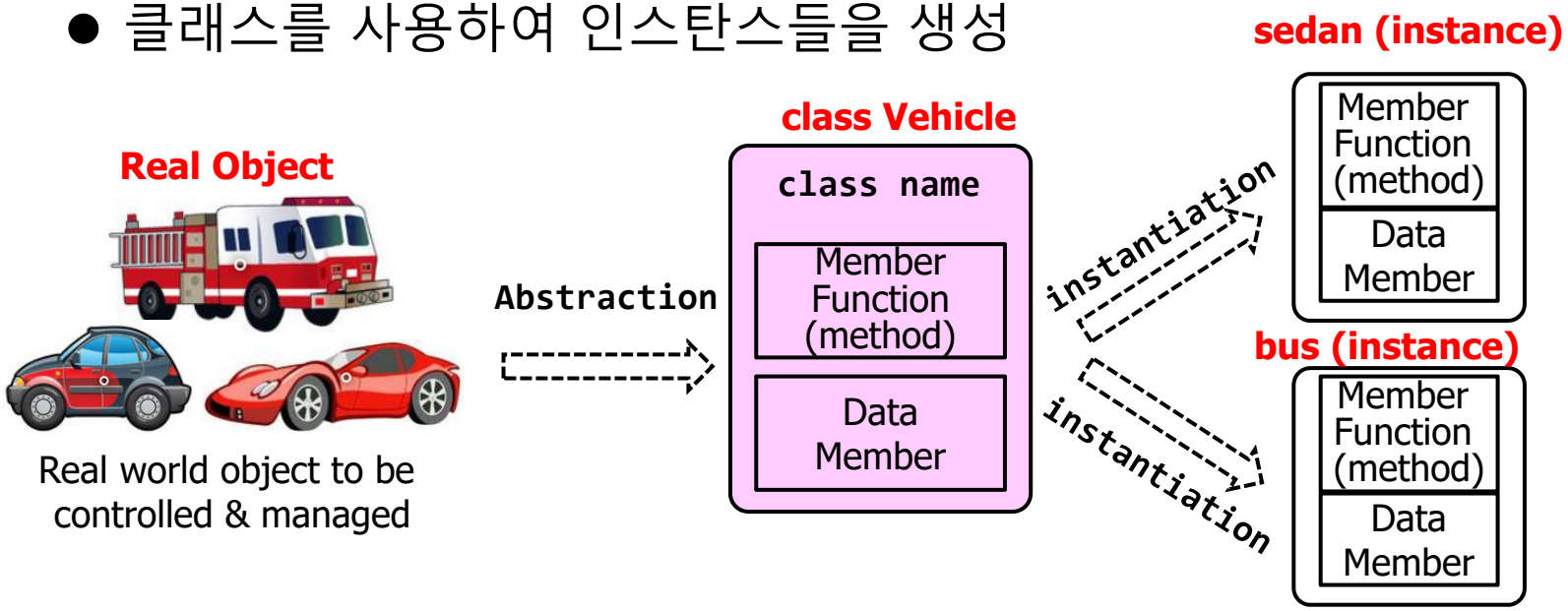
◆예) queue

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

Object, Class, Instance

◆ Object, Class, Instance

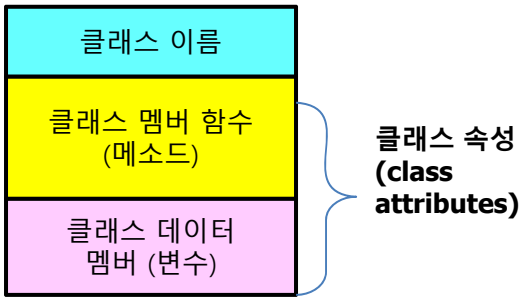
- C++ 와 파이썬은 객체 지향형 프로그래밍 언어임
- 실제 생활환경의 객체들이 클래스로 추상화 모델링되어 소프트웨어로 표현되고 구현됨
- 클래스는 데이터 멤버와 데이터 멤버들을 관리하는 멤버 함수 (메소드)를 포함함
- 클래스를 사용하여 인스탄스들을 생성



클래스 속성 (class attribute)

◆ 클래스 속성 (class attribute)

- 클래스 속성 (class attribute)은 클래스에 포함된 데이터 멤버와 멤버함수로 구성됨
- 클래스의 멤버함수를 메소드(method)라고도 함
- 클래스의 데이터 속성 (데이터 멤버)들은 클래스에 포함된 변수임
- C++에서는 클래스 속성의 접근에 대한 권한은 구분함
  - public: 외부의 모두에게 개방된 속성
  - protected: 해당 클래스로 부터 상속받는 자식/후손 클래스에게만 한정적으로 개방됨
  - private: 외부에 개방되지 않고, 클래스 내부적으로만 사용됨



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(속성)에 대한 접근 권한 제한

◆ 클래스 멤버 (속성)에 대한 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

Public and Private Example (1)

- Private 데이터 멤버를 가지는 클래스

```
class Date
{
public:
        void input();
        void output();
private:
        int year;
       int month;
        int day;
};
```

- Data members are private now
- Objects have no direct access to private data members

Public and Private Example (2)

- **Given previous example**
- **Declare object:**
  Date today;
- **Object *today* can be accessed ONLY by public members**
  - **cin >> today.month; // NOT ALLOWED!**
  - **cout << today.day; // NOT ALLOWED!**
  - Must instead call public operations:
    - **today.input();**
    - **today.output();**

C++ 클래스와 응용 프로그램 (1)

```
/**  Cmplx.h  */
#ifndef CMPLX_H
#define CMPLX_H
#include <iostream>
using namespace std;

class Cmplx
{
public:
    Cmplx(double real=0.0, double imagin=0.0); // constructor
    void inputCmplx();
    void printCmplx();
    const Cmplx addCmplx(const Cmplx &);
    const Cmplx subtractCmplx(const Cmplx &);
    void setReal(double r) { real = r; }
    double getReal() { return real;  }
    void setImagin(double im) { imagin = im; }
    double getImagin() { return imagin;  }
private:
    double real;
    double imagin;
};

#endif
```

```
/**   Cmplx.cpp  (1) */
#include <iostream>
#include "Cmplx.h"

using namespace std;

Cmplx::Cmplx(double r, double i)
:real(r), imagin(i)  // initialization section
{
}

void Cmplx::inputCmplx()
{
      cin >> this->real;
      cin >> this->imagin;
}

void Cmplx::printCmplx()
{
      cout << "Complex (" << this->real << ", "
            << this->imagin << ")" << endl;
}
```

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

const Cmplx
Cmplx::addCmplx(const Cmplx &c)
{
      Cmplx result;

      result.real = real + c.real;
      result.imagin = imagin + c.imagin;

      return result;
}

const Cmplx
Cmplx::subtractCmplx(const Cmplx &c)
{
      Cmplx result;

      result.real = real - c.real;
      result.imagin = imagin - c.imagin;

      return result;
}
```

◆ main()

```
/**  main.cpp    */
#include <iostream>
#include "Cmplx.h"

using namespace std;

void main()
{
   Cmplx c1, c2, c3, c4, c5(123.456, 789.123);

   cout << " input c1.real and c1.imagin: ";
   c1.inputCmplx();
   cout << " c1 : "; c1.printCmplx(); cout << endl;

   cout << " input c2.real and c2.imagin: ";
   c2.inputCmplx();
   cout << " c2 : "; c2.printCmplx(); cout << endl;

   c3 = c1.addCmplx(c2);
   cout << " c3 = c1 + c2 : "; c3.printCmplx(); cout << endl;

   c4 = c1.subtractCmplx(c2);
   cout << " c4 = c1 - c2 : "; c4.printCmplx(); cout << endl;

   cout << " c5 = "; c5.printCmplx(); cout << endl;
}
```

```
input c1.real and c1.imagin: 1.1 2.2
c1 : Complex (1.1, 2.2)

input c2.real and c2.imagin: 3.3 4.4
c2 : Complex (3.3, 4.4)

c3 = c1 + c2 : Complex (4.4, 6.6)

c4 = c1 - c2 : Complex (-2.2, -2.2)

c5 = Complex (123.456, 789.123)
```

C++ 클래스와 응용 프로그램 (2)

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

#ifndef DATE_H
#define DATE_H
#include <iostream>
#include <cstdlib>
using namespace std;

class Date
{
public:
    Date();  // default constructor
    Date(int y, int m, int d); // constructor
    void inputDate();
    void printDate();
    void setDate(int y, int m, int d);
    void setMonth(int newMonth);
    int getYear() {return year; }
    int getMonth() {return month; }
    int getDay() {return day; }
```

```
/* Date.h (2) */
    int getYearDay();
    int getWeekDay();
    int getElapsedDays();
       // get elapsed days from AD 1. 1. 1.
private:
    bool isLeapYear();
    bool isLeapYear(int y);
    bool isValidDate(int y, int m, int d);
    int year;
    int month;
    int day;
};

#endif
```

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

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

enum WEEKDAY {SUN, MON, TUE, WED,
     THR, FRI, SAT};

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

//constructor with given arguments
Date::Date(int y, int m, int d)
{
   setDate(y, m, d);
}

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

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

void Date::setDate(int y, int m, int d)
{
   year = y;
   if ((m >= 1) && (m <= 12))
      month = m;
   else
   {
      cout << "Illegal month value!
                  Program aborted.\n";
      exit(1);
   }
   if ((d >= 1) && (d <= 31))
      day = d;
   else
   {
      cout << "Illegal day value!
                  Program aborted.\n";
      exit(1);
   }
}

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

bool Date::isLeapYear()
{    . . . .  }
```

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

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

   if (isLeapYear())
      days_month[2] = 29;
   for (int m = 1; m < month; m++)
       yearDay += days_month[m];
   yearDay += day;
   return yearDay;
}

int Date::getElapsedDays()
{
   int yearDay;
   int elpsDay = 0;

   for (int y = 1; y < year; y++)
   {
       . . . .
   }
   yearDay = getYearDay();
   elpsDay += yearDay;

   return elpsDay;
}
```

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

int Date::getWeekDay()
{
       int weekDay_AD010101 = MON;
          // AD Jan. 1, 1 was Monday
       int yearDay;
       int weekDay;
       int elapsedDays = 0;

       . . . .

       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 << "Illegal date! Program aborted.\n";
       exit(1);
   }
}
```

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

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 (5) */

void Date::printDate()
{
    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;
    char todayWeekDayName[10];

    if ((month >= 1) && (month <= 12))
       cout << string(monthName[month]);
    cout << " " << day << ", " << year;

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

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

int main()
{
    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);

    cout << "AD Jan. 1, 1 is ";
    AD010101.printDate();
    cout << endl;

    cout << "Today is ";
    today.printDate();
    cout << endl;

    cout << "New year's day of this year was ";
    newYearDay.printDate();
    cout << endl;

    //christmas.setDate(12, 25, 2020);
    Date christmas(year, 12, 25);
    cout << "Christmas of this year is ";
    christmas.printDate();
    cout << endl;

    if (today.getMonth() == christmas.getMonth() &&
    today.getDay() == christmas.getDay())
    {
        cout << "Today is Christmas!
                Happy Christmas to you all !!\n";
    } else {
        cout << "Sorry, today is not Christmas!";
        daysToChristmas =
        christmas.getElapsedDaysFromAD010101() -
        today.getElapsedDaysFromAD010101();
        cout << " You must wait "
           << daysToChristmas << " day(s) to Christmas !"
           << endl;
    }
    return 0;
}
```

```
AD Jan. 1, 1 is    January  1,    1 (    Monday)
Today is  September  8, 2021 ( Wednesday)
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 108 day(s) to Christmas !
```

UML (Unified Modeling Language)

UML (Unified Modeling Language)

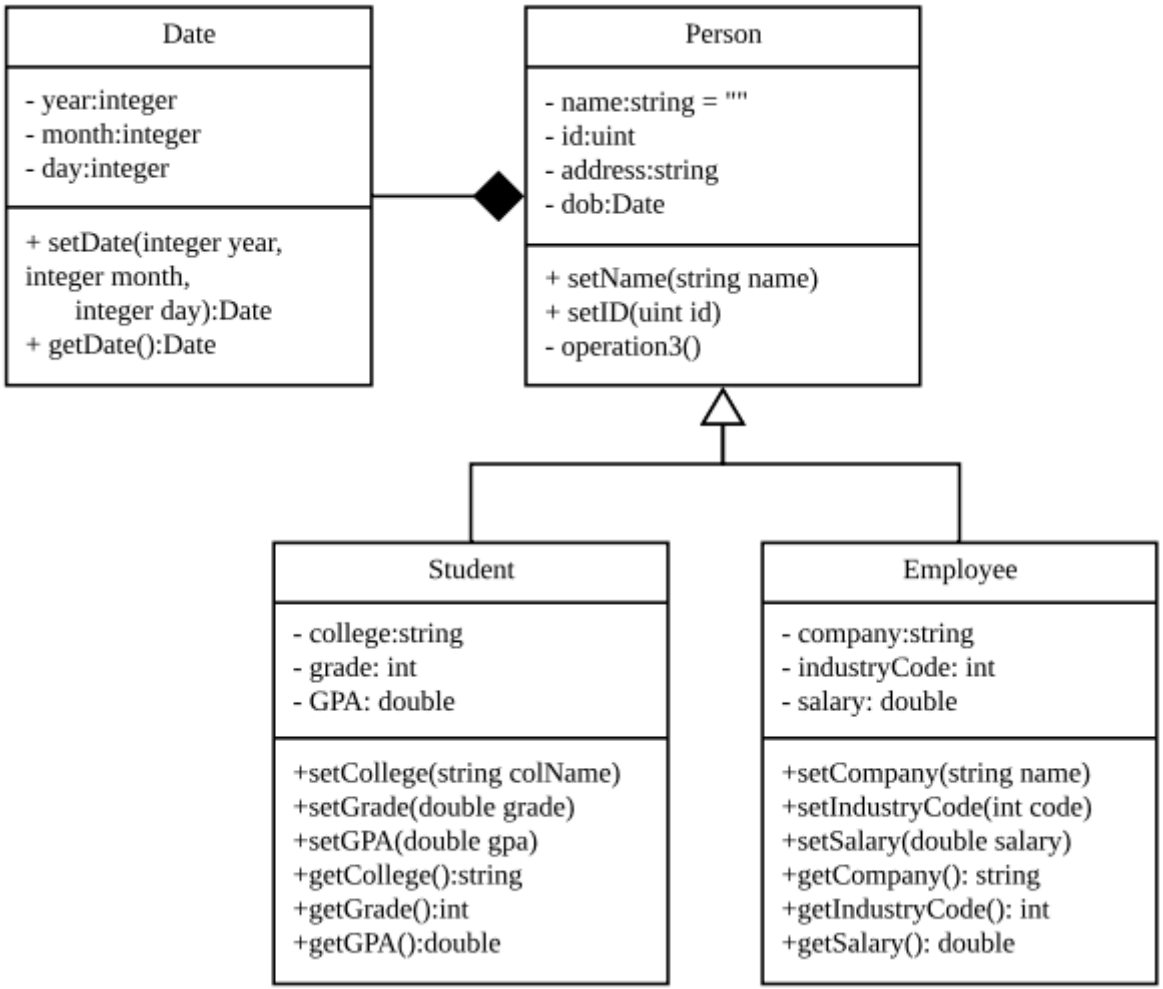
◆ UML (Unified Modeling Language)

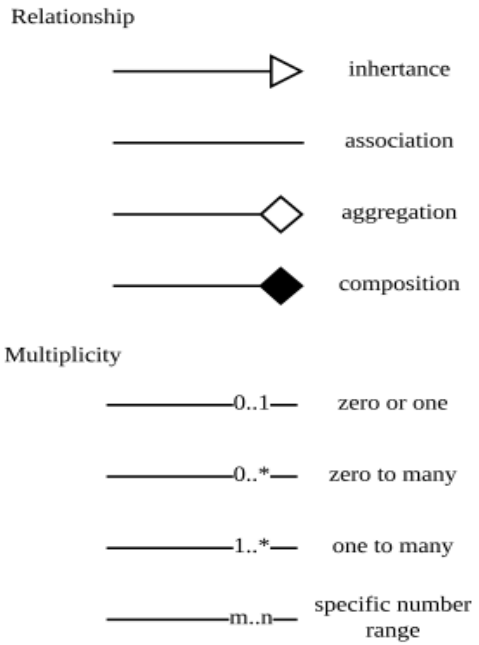
- 컴퓨터 프로그램의 모델링을 위한 언어
- Model-based development
- OMG (Object Management Group)에서 관리

◆ UML Diagrams

Diagram 종류	설명
클래스 다이어그램 (Class Diagram)	Class의 기본 정보 (데이터 멤버(attribute), 멤버 함수(member function))를 표시하며, 클래스 간의 상속 (inheritance) 관계, 포함 (composition, aggregation) 관계를 표시. 각 구성 요소의 visibility (private ('-'), public ('+'), protected ('#')) 표시 데이터 멤버의 자료 유형 (data type) 표시. 멤버 함수 호출에서의 인수 (parameter) 목록과 반환 자료 유형 표시.
기능 순서도 (Sequence Diagram)	클래스에 포함된 주요 기능이 실행되기 위하여 수행되는 세부 기능블록들의 상호 연동을 기능 절차에 따라 표현.
상태 천이도 (State Transition Diagram)	클래스로 생성되는 객체의 동적 상태의 변화를 표현. 객체의 생성으로부터 주요 상태변화가 발생하는 조건과 상태 변화에서 함께 발생되는 동작 (예를 들어 메시지 전송 등)을 표현.

Class Diagram



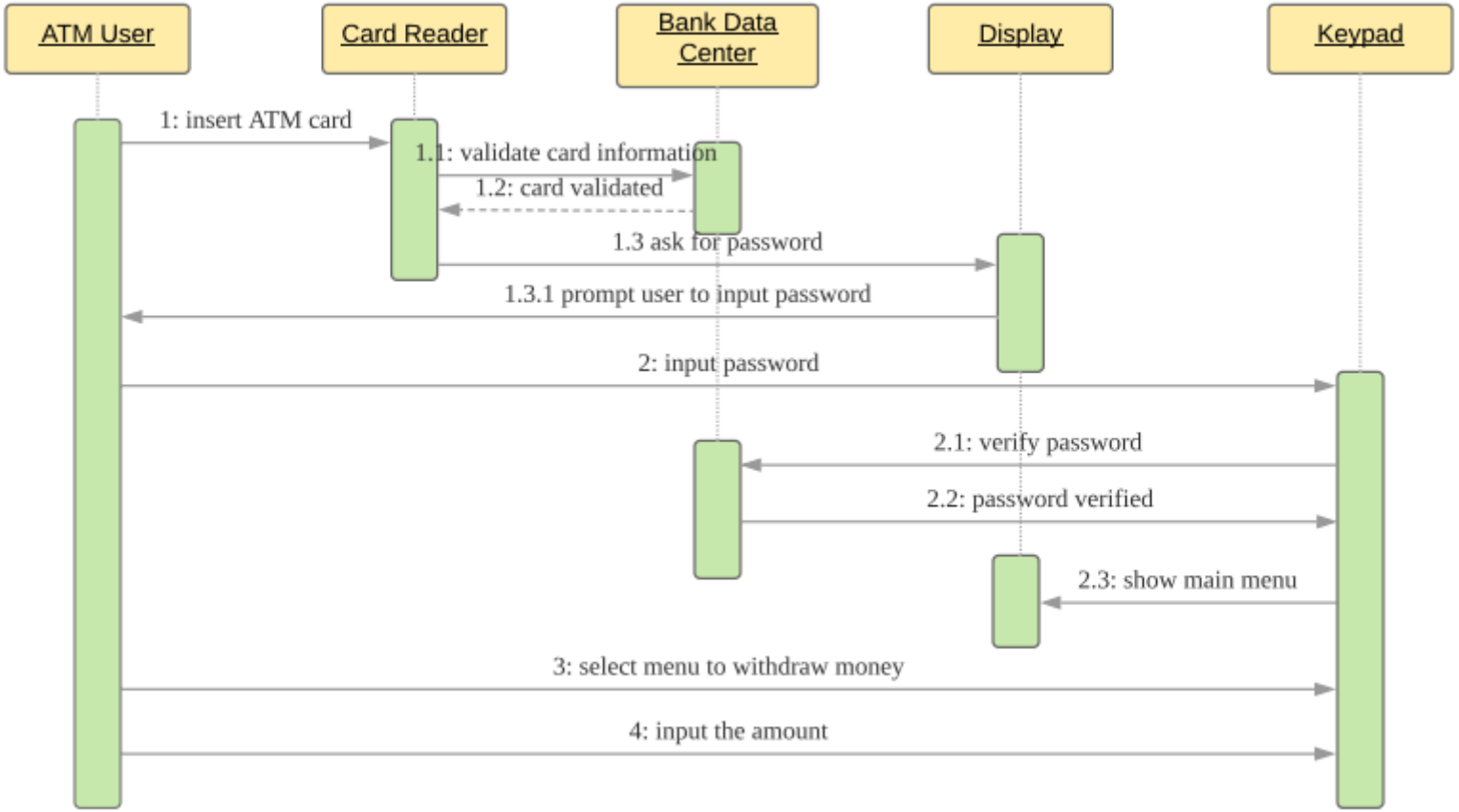


UML Sequence Diagram

◆ Preparation steps for UML Sequence Diagram

- step 1 – define the functional blocks (objects, roles) that send/receive messages as interactions
- step 2 – define the actor that initiates the interaction
- step 3 – draw the first message from the actor to a sub-system
- step 4 – draw message(s) to other sub-system(s)
- step 5 – draw return message to the actor
- step 6 – send/respond to other related actor(s)

Example Sequence Diagram



State Transition Diagram

◆ State transition diagram

- possible states of the object instance
- events that cause a transition from one state to another
- actions that result from a state change (transition)

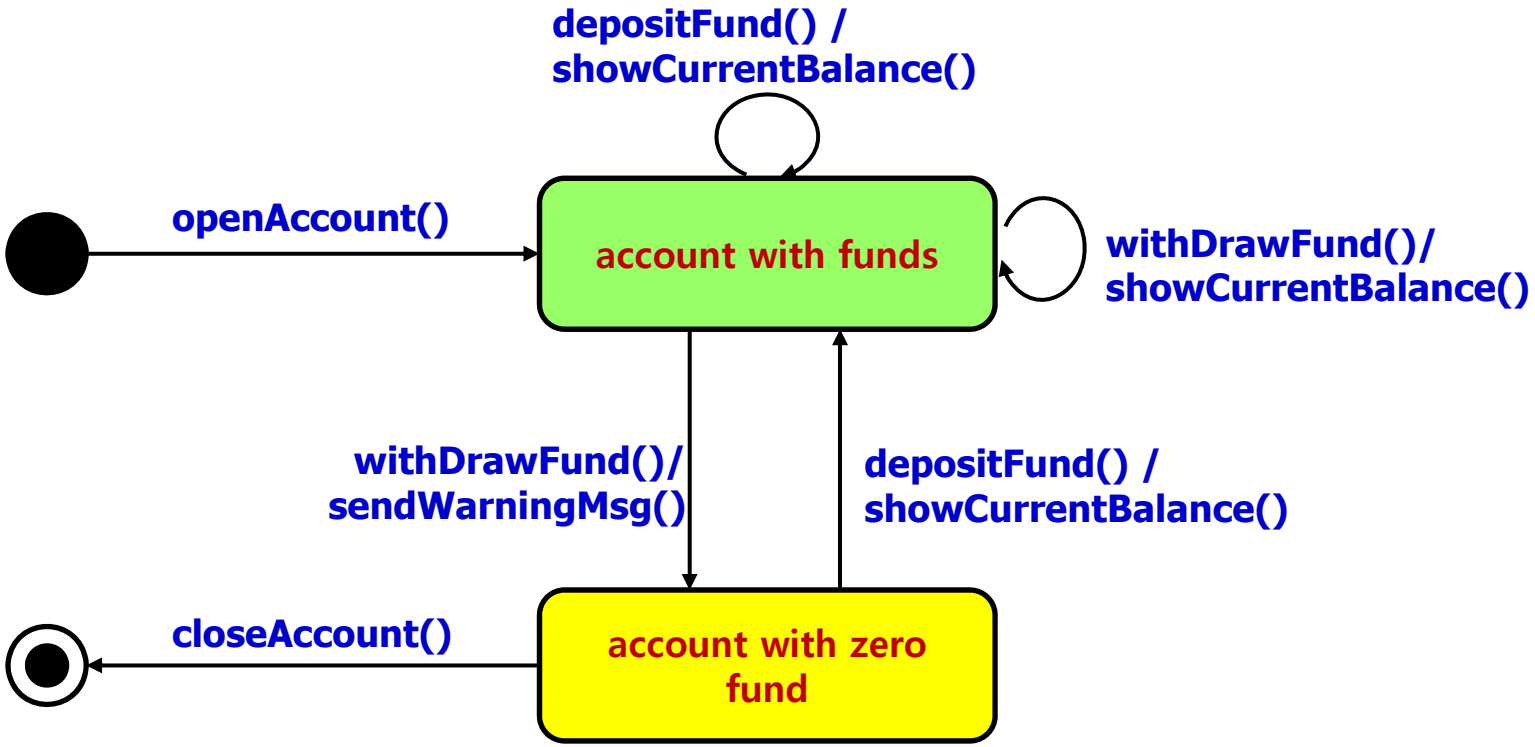
State Transition Diagram

◆ Steps to draw a state machine diagram

- Step 1 – define states (situation or status) of an object instance
- Step 2 – describe states
- Step 3 – define transition triggers
- Step 4 – define condition(s) of transition
- Step 5 – define actions of transition
- Step 6 – draw transitions

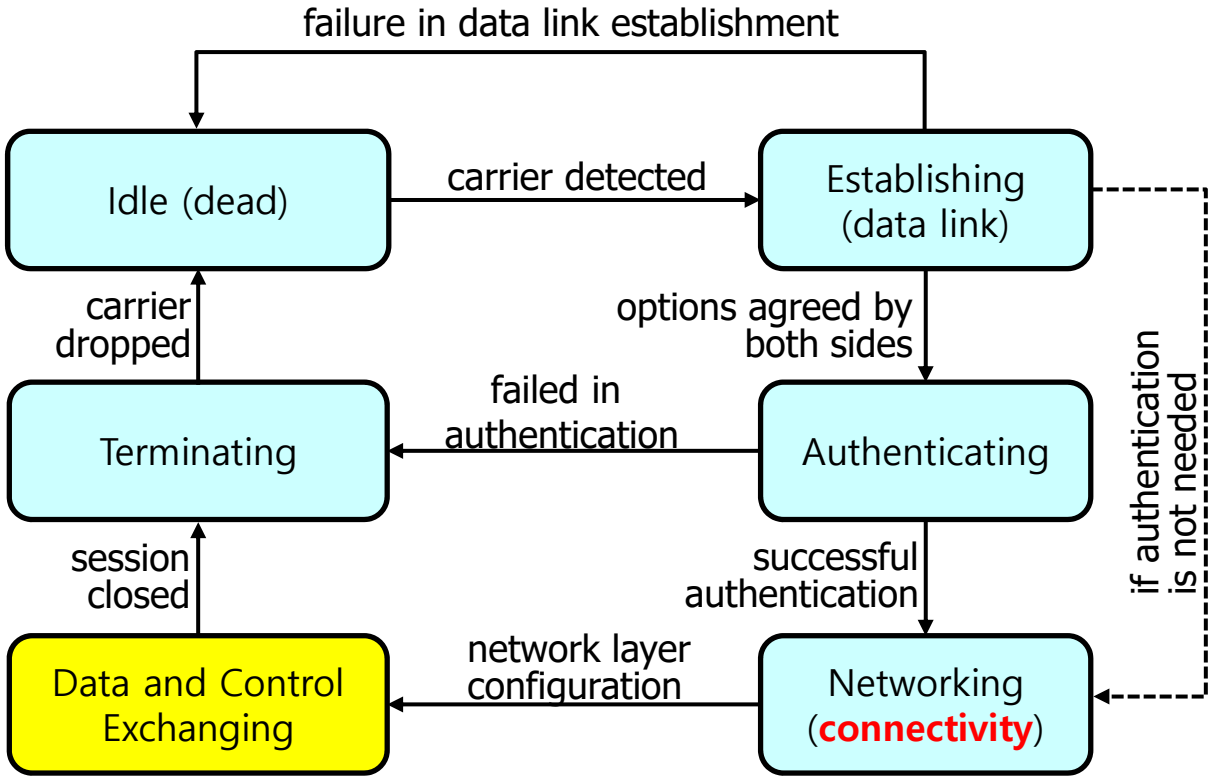
State Transition Diagram

◆ State Transition Diagram for Bank Account



State Transition Diagram

◆ PPP (point-to-point) Protocol State Transition



Thinking Objects

◆Focus for programming changes

- Before → procedure/algorithms center stage
- Object-Oriented Programming (OOP) → data/state is focus

◆Algorithms still exist

- They simply focus on their data
- Are "made" to "fit" the data

◆Designing software solution

- Define variety of objects and how they interact

Homework 2

Homework 2

2.1 Object-oriented programming

(1) C++ Object-Oriented programming에서 encapsulation이란 무슨 의미인가? Encapsulation을 함으로써 어떤 장점이 있는가?

(2) C++ Object-Oriented programming에서 information hiding이란 무슨 의미인가? 무엇을 누구에게 숨기는 것인가?

(3) C++ Object-oriented programming에서 data protection이란 무슨 의미인가? 어떻게 데이터를 보호하는가?

(4) C++ Object-oriented programming에서 information hiding과 data protection의 차이점은 무엇인가?

2.2 Programming with class Date

(1) Design a class Date that has private data members of year, month, and day. The class Date has public member functions of getYearDay(), getWeekDay(), getElapsedDays(), and other necessary accessor and mutator functions.

(2) Implement the member functions.

(3) Use following main() to test the class Date.

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

#include <iostream>
#include "Date.h"
using namespace std;

int main()
{
	Date AD010101(1, 1, 1), newYearDay(2021, 1, 1), today, christmas(2021, 12, 25);
	int year;
	int daysToChristmas;

	cout << "AD Jan. 1, 1 is ";
	AD010101.print();
	cout << endl;
```

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

        cout << "New year's day of 2020 is ";
        newYearDay.print();
        cout << endl;

        cout << "Christmas of this year is ";
        christmas.print();
        cout << endl;

        today.input();
        cout << "Today is ";
        today.print();
        cout << endl;

        if (today.getMonth() == christmas.getMonth() &&
        today.getDay() == christmas.getDay())
        {
           cout << "Happy Christmas !!\n";
        }
        else {
           cout << "Sorry, today is not Christmas!";
           daysToChristmas = christmas.getElapsedDays() - today.getElapsedDays();
           cout << " You must wait " << daysToChristmas << " day(s) to Christmas !" << endl;
        }
        return 0;
}
```