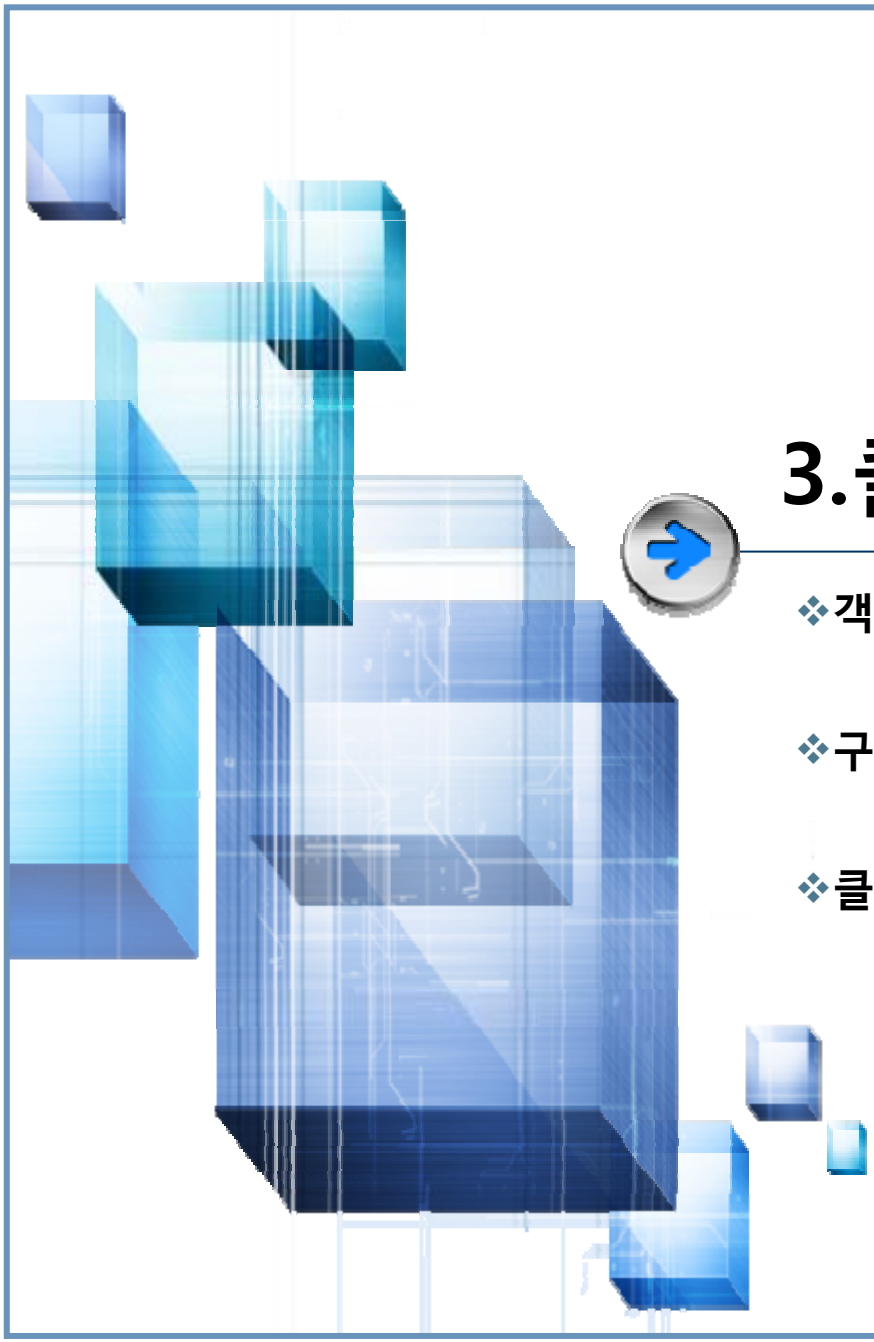


3.클래스의 기본

- ❖객체지향 프로그래밍 소개
- ❖구조체와 클래스
- ❖클래스의 정의

Jong Hyuk Park

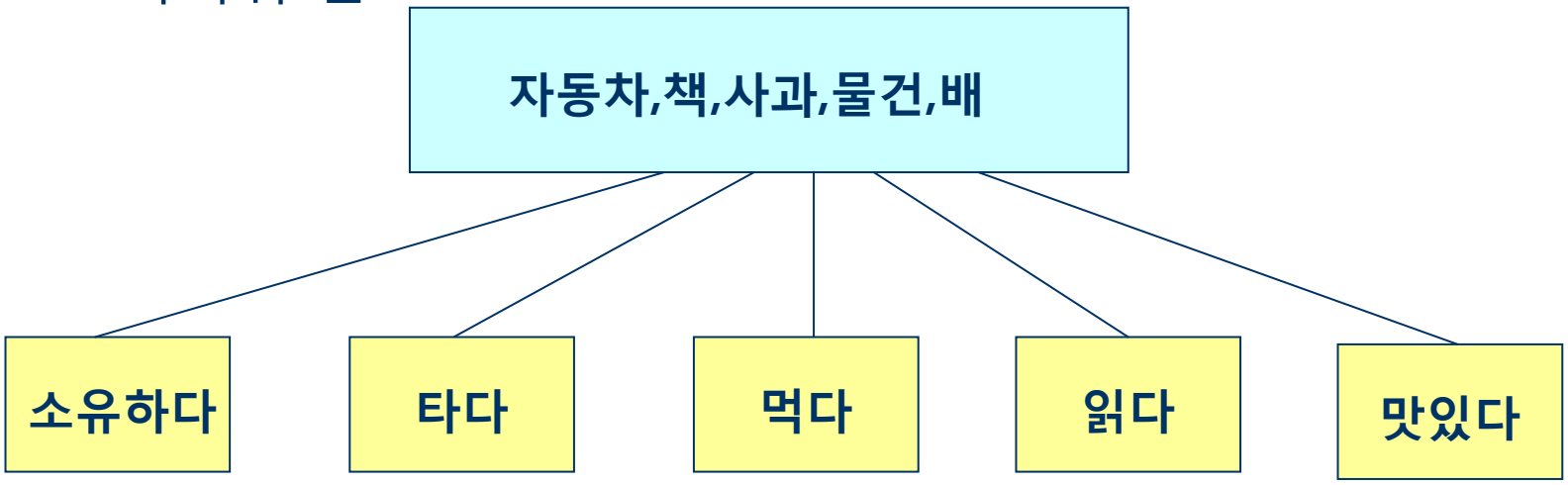
객체지향 프로그래밍 소개

Jong Hyuk Park

구조적 프로그래밍 개념

❖ 기존 C와 같은 구조적 프로그래밍 언어는 동작되는 **자료**와 **처리 동작** 자체를 서로 별도로 구분

- 처리동작과 자료 사이의 관계가 서로 밀접한 연관성을 갖지 못함
- 프로그램이 커지거나 복잡해지면 프로그램이 혼란스럽게 되어 에러를 찾는 디버깅 및 프로그램의 유지보수가 어려워 짐



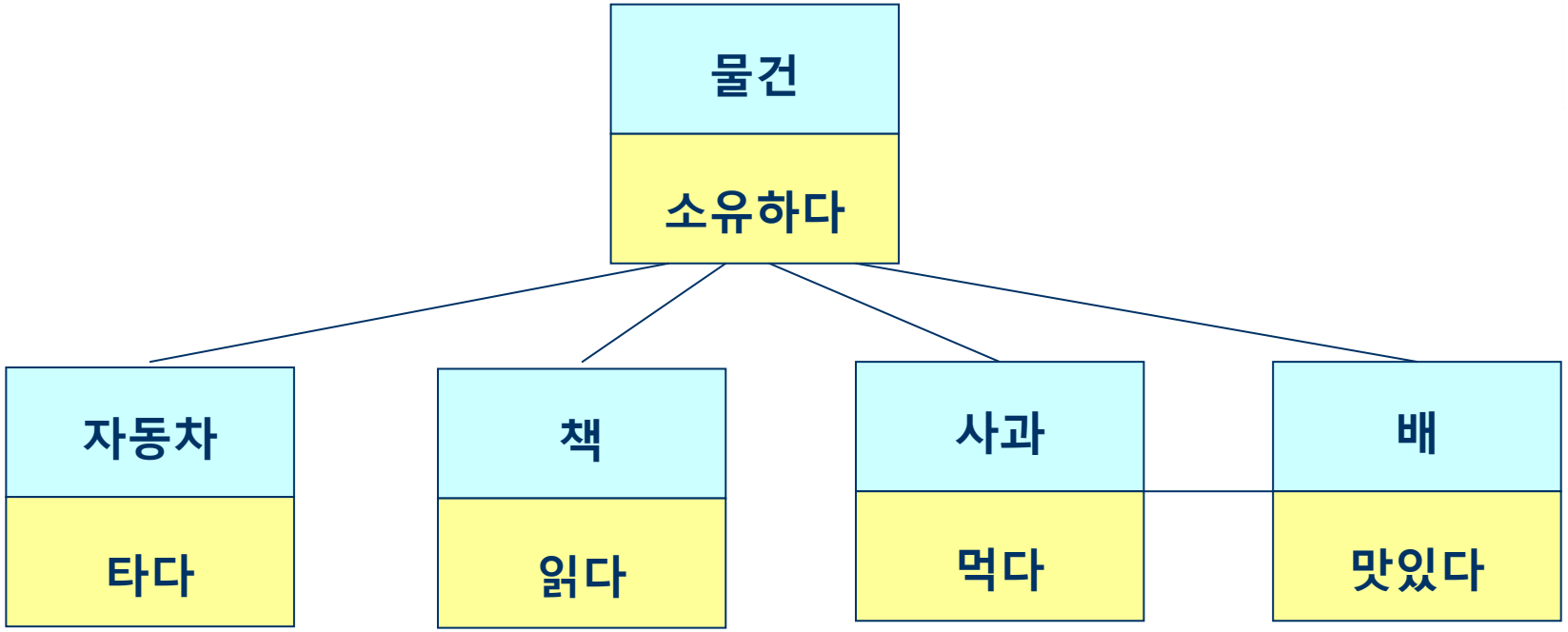
객체(object) 개념 (1)

- 객체지향 프로그래밍에서는 자료와 이의 처리동작을 하나로 묶어서 다룰수 있는 **객체(object)**라는 개념을 도입
- 프로그램은 처리하는 절차보다도 동작되는 자료에 중점을 둔 객체, 객체 간의 상호관계로 표현

물건	자동차	사과	책	배
소유하다	타다	먹다	읽다	맛있다

객체(object) 개념(2)

- 물건의 객체는 다른 객체 보다 상위관계를 갖는 객체이고 이 객체에서 갖는 성질은 다른 객체도 공유
- 비슷한 성질의 사과, 배는 서로 성질을 공유하는 관계



객체지향 프로그래밍 (1)

❖ 객체지향 프로그래밍 시각

- 문제의 영역을 단순한 자료의 처리 흐름으로 보지 않음
  - 구조적 프로그래밍 서로 관련된 자료와 연산(함수)들이 서로 독립적으로 정의되어 취급
- 문제 영역 내에 존재하는 여러 연관된 객체들을 정의하고 이들 객체들이 서로 정보를 주고 받는다고 보는 시각 (객체 간의 관계)

❖ 객체지향 프로그래밍에서는 프로그램은 여러 개의 객체로 구성

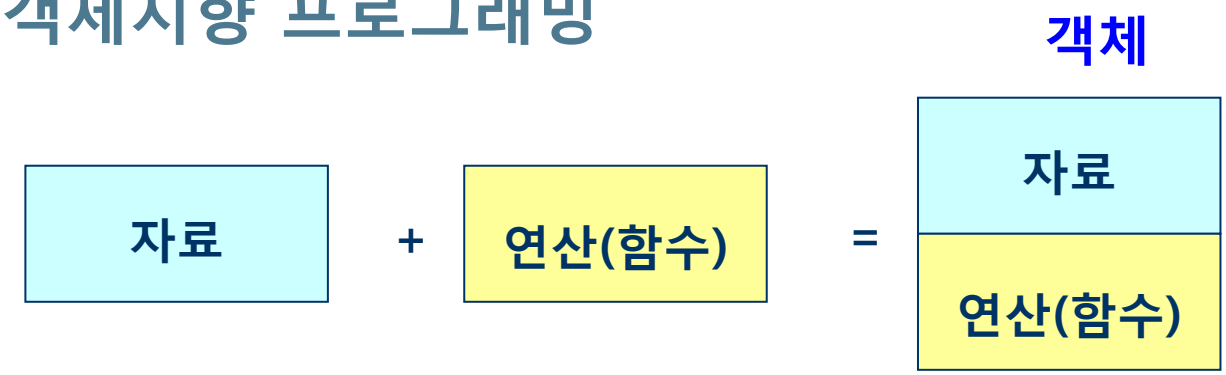
- 객체(object)는 자료(특성(attribute))와 이를 대상으로 처리하는 동작인 연산(함수,메쏘드(method))을 하나로 묶어 만든 요소로 프로그램을 구성하는 실체
- 객체란 단순히 자료를 표현하는 변수 만을 가지는 것이 아니라 그 객체가 무엇을 할 수 있는가를 정의한 함수(메쏘드)로 구성

객체지향 프로그래밍 (2)

❖ 구조적 프로그래밍

자료 + 연산(함수) = 프로그램

❖ 객체지향 프로그래밍



객체 + 객체 = 프로그램

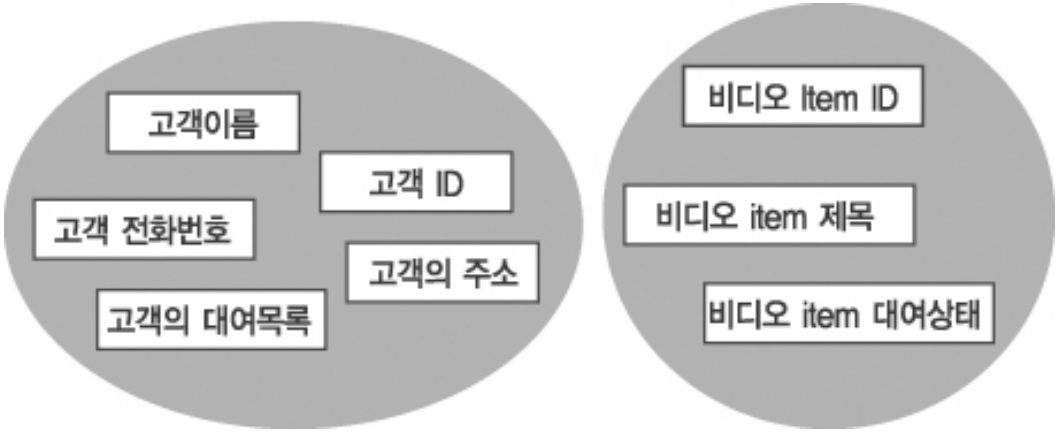
구조체와 클래스

Jong Hyuk Park

구조체

❖ 구조체의 유용성

- 관련 있는 데이터를 하나의 자료형으로 그룹화
- 따라서, 프로그램의 구현 및 관리가 용이
- 함께 움직이는 데이터들을 묶어주는 효과



** 부류를 형성하는 데이터들을 하나의 자료형으로 정의
→ 관리 및 프로그램에 도움을 주겠다는 의미

구조체와 클래스

❖C 언어의 구조체에 대한 불만

- 모든 사용자 정의 자료형에 대한 불만
- 기본 자료형으로 인식해 주지 않는다.

** C언어와 C++의 차이점

사용자 정의자료형 ~~struct~~ **Person**

VS

기본자료형

```
struct Person{
    int age;
    char name[10];
};

int main()
{
   int a=10;
   Person p;    // c에서는 struct Person p ;

   return 0;
}
```

C++의 철학과 맞물리면?

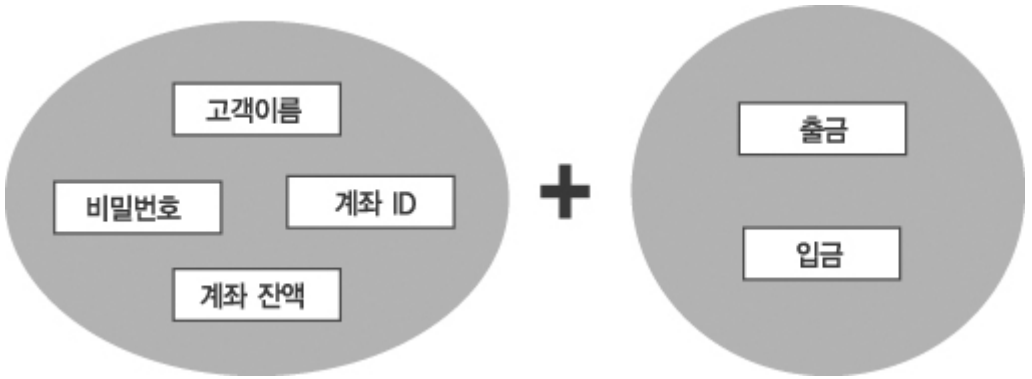
** 기본자료형과 사용자정의자료형 모두 동일하게 인식하자 **

예제) oop1.cpp 확인

구조체+함수

❖ 함수를 넣으면 좋은 구조체

- 프로그램=데이터+데이터 조작 루틴(함수)
- 잘 구성된 프로그램은 데이터와 더불어 함수들도 그룹화
- 객체지향 프로그래밍 기법



** 부류를 형성하는 데이터 : 데이터 부류

기능 분류

구조체+함수 예 (1)

예제) oop2.cpp

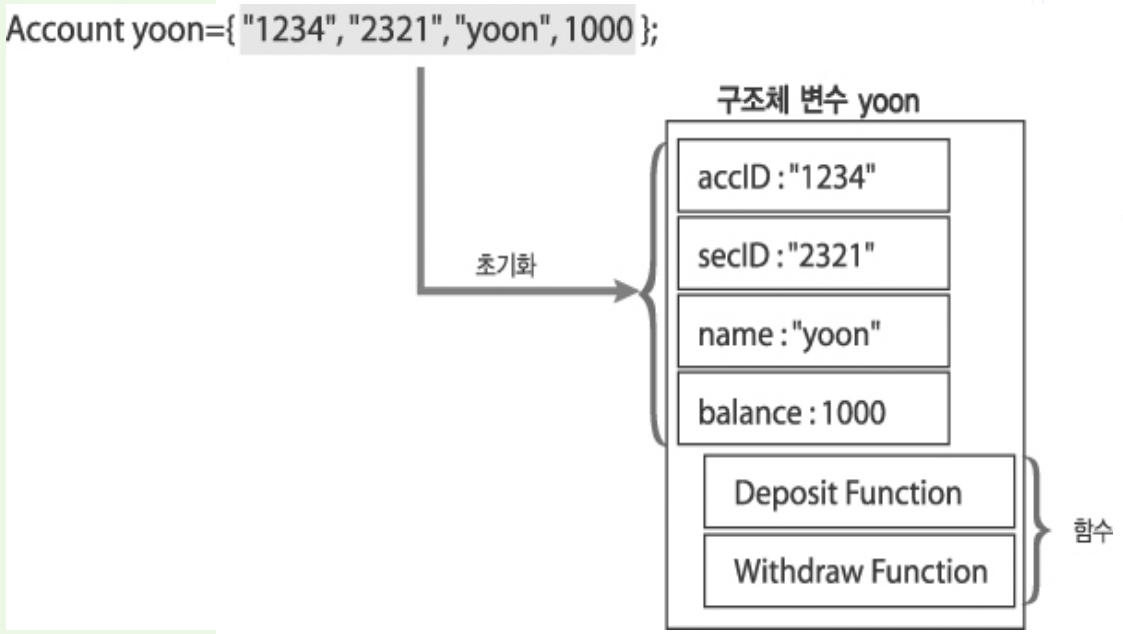
```
struct Account {
    char accID[20];
    char secID[20];
    char name[20];
    int balance;
};

void Deposit(Account &acc, int money)
{
    acc.balance+=money;
}

void Withdraw(Account &acc, int money)
{
    acc.balance-=money;
}
```

예제) oop3.cpp의 일부

```
struct Account {
    char accID[20];
    char secID[20];
    char name[20];
    int balance;

    void Deposit(int money){
        balance+=money;
    }
    void Withdraw(int money){
        balance-=money;
    }
};
```

구조체+함수 예 (2)

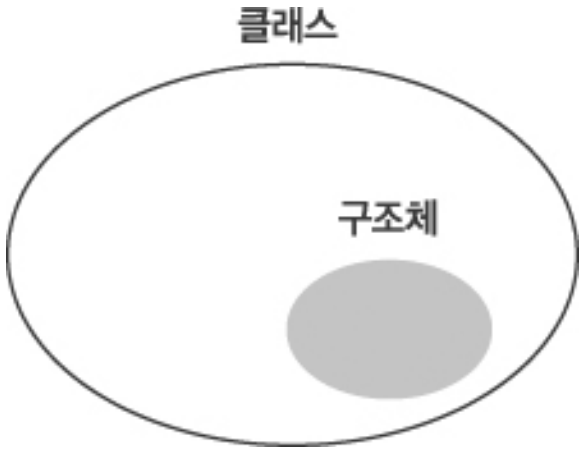
```
struct Account {
      char accID[20];
      char secID[20];
      char name[20];
      int balance;

      void Deposit(int money){
          balance+=money;
      }
      void Withdraw(int money){
          balance-=money;
      }
};

int main(void)
{
    Account yoon={"1234", "2321", "yoon", 1000};
    yoon.Deposit(100);
    cout<<"잔     액 : "<<yoon.balance<<endl;
    return 0;
}
```



클래스 (1)

❖ 객체지향 프로그래밍에서는 구조체가 아니라 **클래스(class)**

- 클래스 = 멤버 변수 + 멤버 함수
- 변수가 아니라 객체(object)



데이터 추상화와 클래스

❖ 사물의 관찰 이후의 데이터 추상화

- 현실 세계의 사물을 데이터적인 측면과 기능적인 측면을 통해서 정의하는 것

예제) 코끼리를 자료형으로 정의
➔ 특징들을 뽑아내야 한다.

1.발이 네개 → 데이터
2.코의길이가 5m 내외 → 데이터
3.몸무게는 1톤이상 → 데이터
4.코를 이용해서 목욕을 함 → 기능
5.코를 이용해서 물건을 집기도 함 → 기능

프로그램

변수

함수

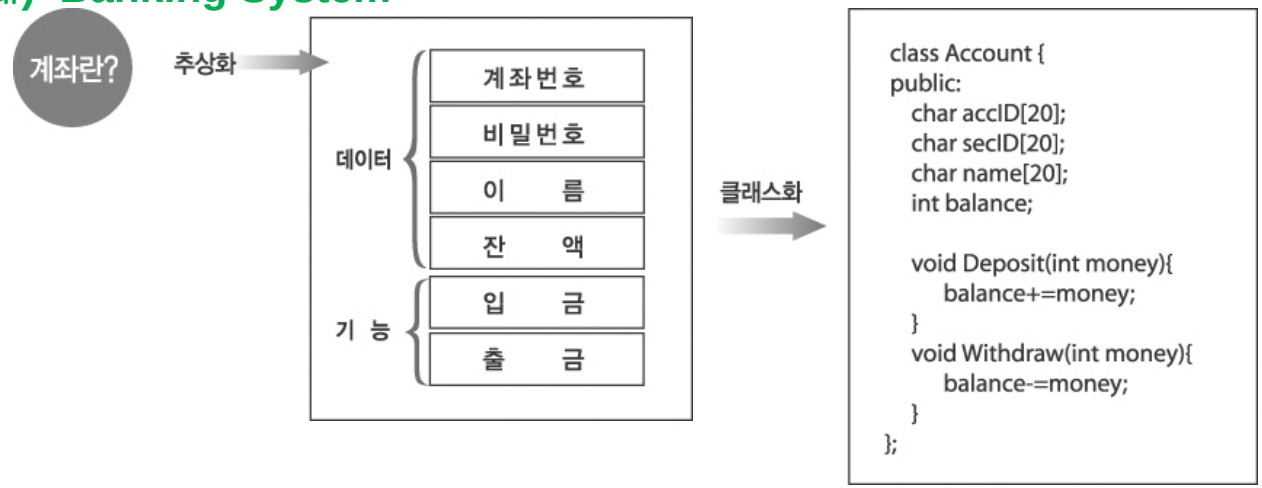
데이터 추상화

데이터 추상화와 클래스

❖ 데이터 추상화 이후의 클래스화

- 추상화된 데이터를 가지고 사용자 정의 자료형을 정의하는 것

예제) **Banking System**



객체 생성

예제) oop4.cpp

❖ 선언된 클래스를 사용하여 객체를 생성

- 클래스화 이후의 인스턴스화(instantiation)

```
class Account {
public:
   char accID[20];
   char secID[20];
   char name[20];
   int balance;

   void Deposit(int money){
       balance+=money;
   }
   void Withdraw(int money){
       balance-=money;
   }
};
```

인스턴스화

```
int main(void
{
   Account yoon={"1234", "2321", "yoon", 1000};
   .....
}
```

클래스의 정의

Jong Hyuk Park

클래스 선언 형식

- C++ 언어에서 클래스의 정의는 **C 언어의 구조체(struct) 선언과 비슷**
  - 다른 점은 클래스의 멤버로서 자료 뿐만 아니라 연산을 위한 함수도 포함된다는 점

```
class 클래스명 {
[private:]
        자료 선언;
        함수 선언;
public:
        자료 선언;
        함수 선언;
} [객체 변수];
클래스명 객체변수, ……;
```

```
class Test {
 private:
        int a;
        void inc();
public:
        char s[10];
} var;


Test t1, t2;
```

클래스 멤버의 접근 제어

❖클래스의 내부 접근과 외부 접근

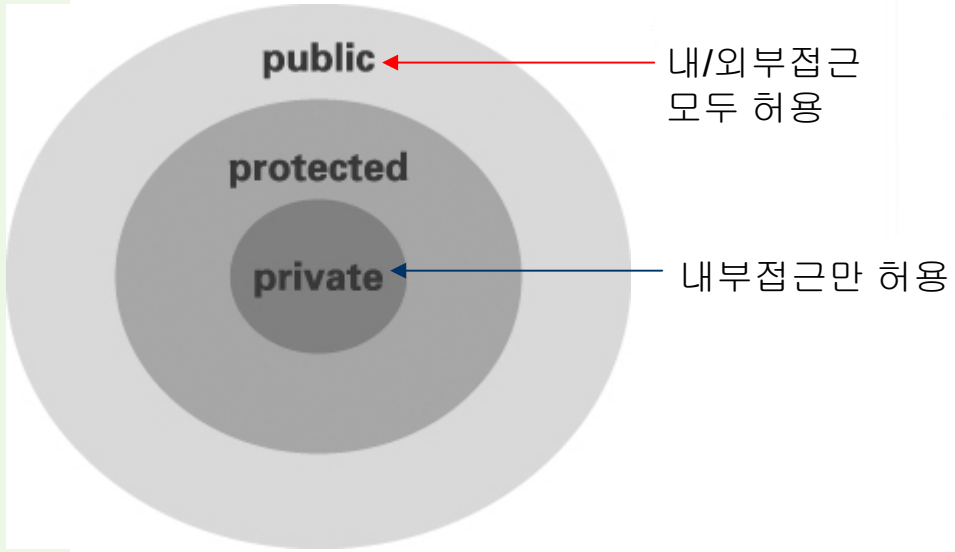
```
class Counter {
public:
    int val;
    void Increment(void)
    {
        val++;                          //내부 접근
    }
};

int main(void)
{
   Counter cnt;
   cnt.val=0;                           //외부 접근
   cnt.Increment();                     //외부 접근
   cout<<cnt.val<<endl;                 //외부 접근
   return 0;
}
```

같은 클래스내에 존재하는 멤버에 의한 접근

내부접근 외의 모든 접근

클래스 멤버의 접근 제어

```
const int OPEN=1;
const int CLOSE=2;

class Door{
private:
    int state;
public:
    void Open(){ state=OPEN; }
    void Close(){ state=CLOSE; }
    void ShowState(){ ...생략...  }
};

int main()
{
    Door d;
    //d.state=OPEN;
    d.Open();
    d.ShowState();
    return 0;
}
```



C++의 접근 제어 키워드: 접근의 범위

클래스 영역 (1)

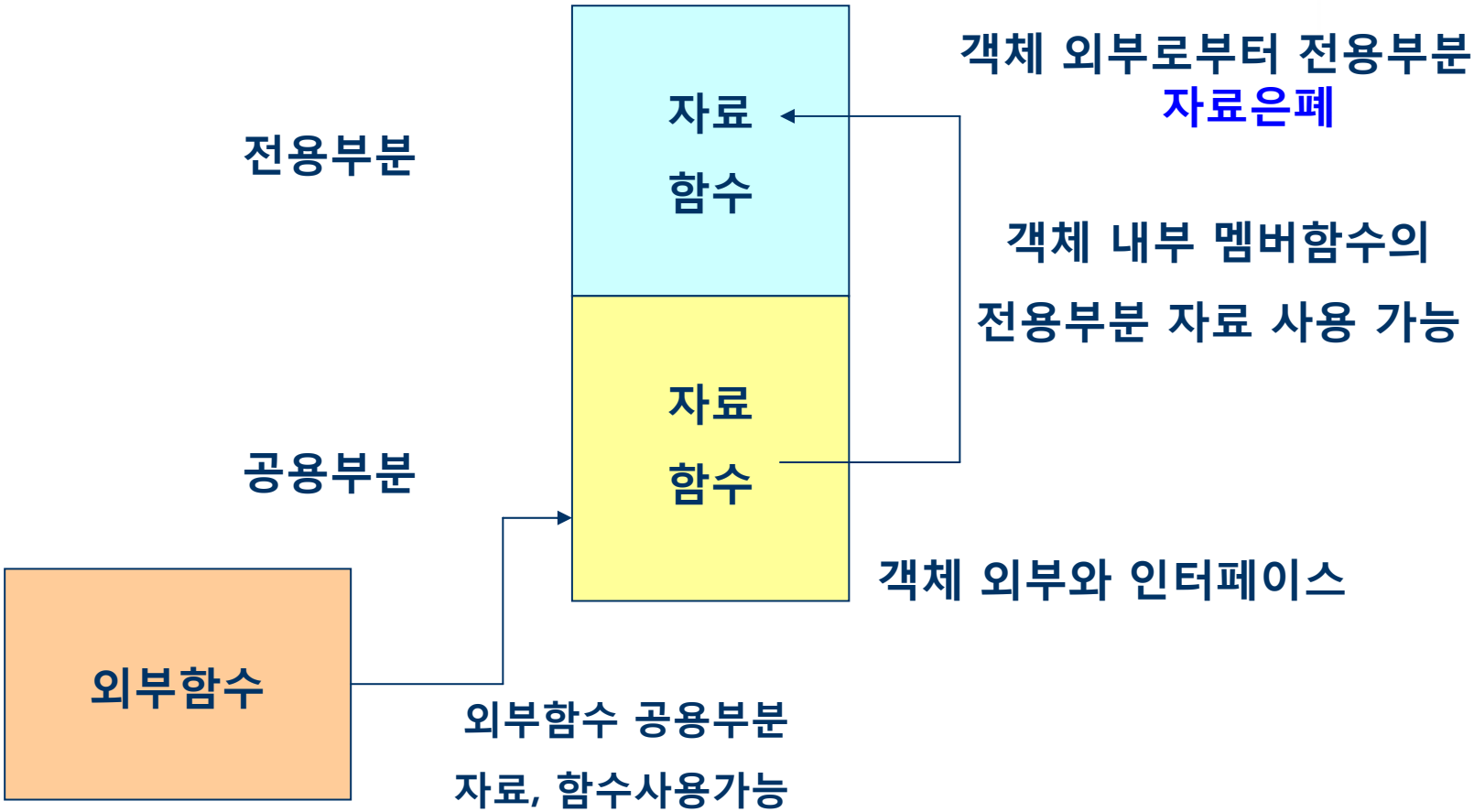
- 클래스를 정의할 때 private과 public 영역으로 구분

- **private은 전용부분**
  - private 영역에서 정의된 자료형이나 함수는 오직 해당 객체 내부의 멤버함수만이 사용
  - 외부에 대해 자료의 정보가 은폐
  - 전용부분에는 함부로 변경되어서는 안될 자료와 객체 외부에서 호출되어서는 안될 멤버함수를 정의

- **public은 공용부분**
  - 객체 외부에서 사용될 수 있는 자료나 함수가 정의
  - 클래스 정의 시 키워드 private이 생략되면 public 키워드가 나올 때까지의 부분을 전용멤버로 간주

- 클래스 영역으로는 위 두 영역 외에 **protected**로 구분되는 보호부분이 있는 데 7장에서 파생 클래스를 설명할 때 소개

클래스 영역 (2)


전용부분
자료
함수
객체 외부로부터 전용부분
자료은폐
객체 내부 멤버함수의
전용부분 자료 사용 가능
공용부분
자료
함수
객체 외부와 인터페이스
외부함수
외부함수 공용부분
자료, 함수사용가능

클래스 멤버함수의 선언

❖ 클래스의 멤버로서 자료 뿐만 아니라 **함수도 정의**

- 함수의 본체 내용은 직접 클래스 내부나 또는 클래스 외부에서 기술
- 클래스 내부에서 함수의 본체가 기술된 멤버함수는 모두 인라인 함수(inline function)로 정의

```
class 클래스명 {

        ……

        자료형 함수명(인수 선언) { 함수 본체 }

        ……

}
```

클래스 사용 예 (1)

```
#include <iostream>
using std::cout;
using std::endl;

class Test {
        char c;
public:
        int   x;
        float y;
        void  set_c(char a)   { c = a; }  // set_c() 멤버함수 정의
        char  get_c()    { return c; }    // get_c() 멤버함수 정의
};
int main(void)
{
   Test o;              // 클래스 Test 형의 객체 o 선언

   o.x = 12;
   o.y = 3.14;
   o.set_c('H');        // set_c() 멤버함수 호출
   cout << o.x << " , " << o.y << endl;
   cout << o.get_c() << endl;
   return 0;
}
```

12 , 3.14
H

전용부분 자료 c의 접근은 클래스 내부의 멤버함수 set_c(), get_c()에서만 허용되고 클래스 외부에서는 이들 공용부분 멤버함수의 호출에 의해서 간접적으로 접근된다.

클래스 사용 예 (2)

```
#include <iostream>
using std::cout;
using std::endl;

class Counter {
        int value;
public:
        void set(int n)  { value = n; }
        void inc()       { ++value; }
        void dec()       { --value; }
        int  val()       { return value; }
};
int main(void)
{
   Counter cnt1, cnt2;
   cnt1.set(10);
   cnt1.inc();
   cout << "cnt1 value : " << cnt1.val() << endl;

   cnt2.set(0);
   cnt2.dec();
   cout << "cnt2 value : " << cnt2.val() << endl;

   return 0;
}
```

cnt1 value : 11
cnt2 value : -1

클래스 Counter를 사용하여 서로 독립된 객체cnt1, cnt2가 선언되어 서로 독립적으로 수행

클래스 멤버함수의 외부 정의

❖ 함수의 본체가 긴 경우에 클래스의 멤버함수를 클래스 내부가 아닌 클래스 **외부에서 정의**하는 것이 편리

```
class 클래스명 {
        ……
        자료형 함수명(인수 선언);
        ……
 }
[inline] 자료형 클래스명::함수명(인수 선언)
{
    함수본체
}
```

클래스 멤버함수의 외부 정의

```
class Door{
private:
    int state;

public:
    void Open();
    void Close();
    void ShowState();
};

void Door::Open(){
   state=OPEN;
}
void Door::Close(){
   state=CLOSE;
}
void Door::ShowState(){
   ... 생  략 ...
}
```

Door 클래스 내에 선언되어 있는 Open 함수의 정의

```
void Door::Open( ) {
   state=OPEN;
}
```

클래스 멤버함수의 외부 정의

❖ 멤버 함수의 인-라인화

```
const int OPEN=1;
const int CLOSE=2;

class
Door{
private:
    int state;
public:
    void Open(){
        state=OPEN;
    }
    void Close(){
        state=CLOSE;
    }
    void ShowState() {
     ...생략...
    }
};
```

equal!

```
class Door{
private:
    int state;

public:
    void Open();
    void Close();
    void ShowState();
};

inline  void Door::Open(){
   state=OPEN;
}
inline  void Door::Close(){
   state=CLOSE;
}
inline  void Door::ShowState(){
   ... 생  략 ...
}
```

- 기본적으로 클래스의 멤버함수 내부정의는 "멤버함수의 인라인화"하라는 의미임.
- 외부정의시 인라인화를 원할경우: 오른쪽과 같이 사용 (동일한 의미임)

클래스 멤버함수의 외부 정의

❖헤더 파일을 이용한 파일의 분할

Door.h

```
#include <iostream>
using std::cout;
using std::endl;

const int OPEN=1;
const int CLOSE=2;

class Door{
private:
   int state;

public:
   void Open();
   void Close();
   void ShowState();
};
```

Door.cpp

```
#include "Door.h"

void Door::Open(){
   state=OPEN;
}
void Door::Close(){
   state=CLOSE;
}
void Door::ShowState(){
   cout<<"현재 문의 상태 :";
   … 생략…
}
```

Main.cpp

```
#include "Door.h"

int main()
{
   Door d;

   d.Open();
   d.ShowState();

   return 0;
}
```

클래스 사용 예 (3)

```
#include <iostream>
using std::cout;
using std::endl;

class Square {
        int side;
public:
        void set(int);
        int  area();
};

inline void Square::set(int x)
{
   side = x;
}

int Square::area()
{
   return side*side;
}
```

```
main()
{
   Square a;

   a.set(12);
   cout << "area : " << a.area() << endl;
}
```

area : 144

클래스 외부에 정의되는 멤버함수를 확장함수로 선언하고자 할 때에는 inline 키워드를 명시해야 한다.

과제 및 실습

HW #1. P113, 연습문제 3-1: 계산기 기능 프로그램

실습. 2장 과제의 은행계좌 관리 프로그램에 다음을 수정하여라.

- 구조체로 선언된 Account를 클래스로 전환
- 계좌번호, 잔액, 고객이름은 private 선언
- 각 함수들은 클래스의 멤버함수로 선언하고, 함수는 클래스 외부에서 정의

Q & A

Jong Hyuk Park