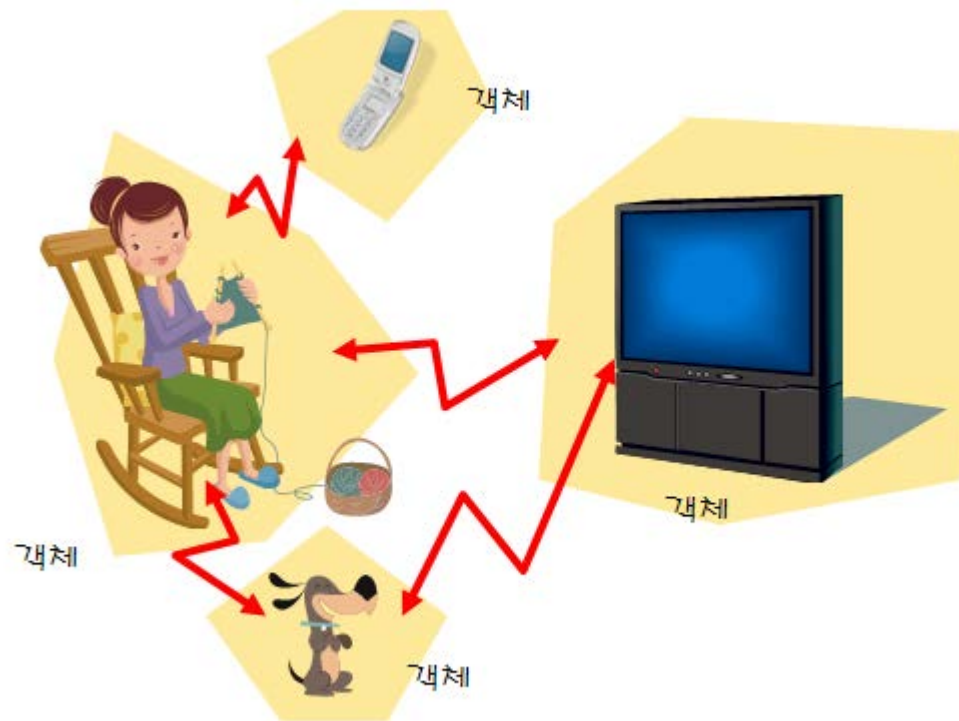




*Power C++*

## 제9장 클래스의 기초





# 이번 장에서 학습할 내용



- 클래스와 객체
- 객체의 일생
- 메소드
- 필드
- UML

직접  
클래스를  
작성해  
봅시다.





# QUIZ

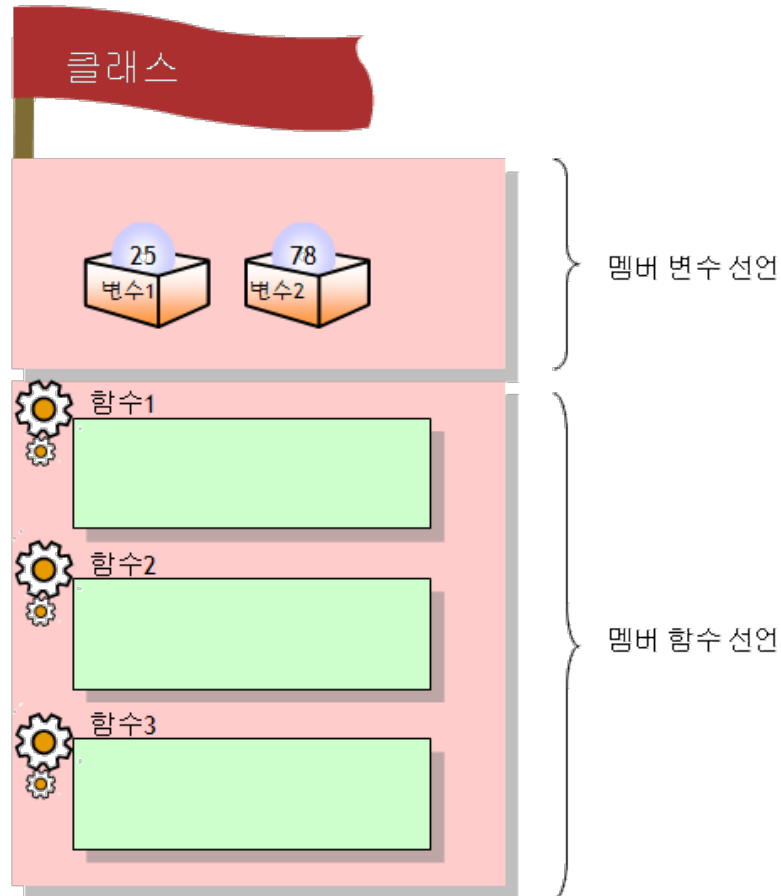
1. 객체는 **속성** 과 **동작** 을 가지고 있다.
2. 자동차가 객체라면 클래스는 **설계도** 이다.

먼저 앞장에서  
학습한 클래스와  
객체의 개념을  
복습해봅시다.





# 클래스의 구성



- 클래스(class)는 객체의 설계도라 할 수 있다.
- 클래스는 멤버 변수와 멤버 함수로 이루어진다.
- 멤버 변수는 객체의 속성을 나타낸다.
- 멤버 함수는 객체의 동작을 나타낸다.



# 클래스 정의의 예



```
class Car {
```

```
public:
```

```
// 멤버 변수 선언
```

```
int speed; // 속도
```

```
int gear; // 기어
```

```
string color; // 색상
```

멤버 변수 정의!

```
// 멤버 함수 선언
```

```
void speedUp() { // 속도 증가 멤버 함수  
    speed += 10;
```

```
}
```

```
void speedDown() { // 속도 감소 멤버 함수  
    speed -= 10;
```

```
}
```

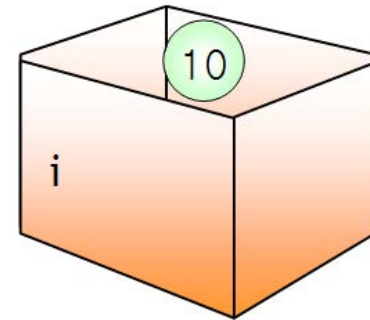
```
};
```

멤버 함수 정의!



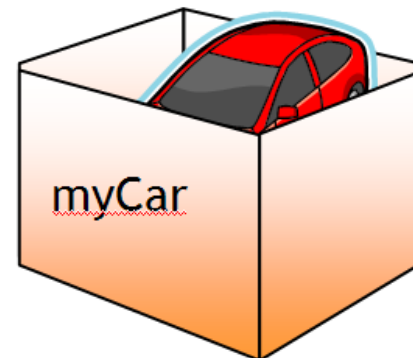
# 객체

- int 타입의 변수를 선언하는 경우  
`int i;`



- 클래스도 타입으로 생각하면 된다.
- Car 타입의 변수를 선언하면 객체가 생성된다.

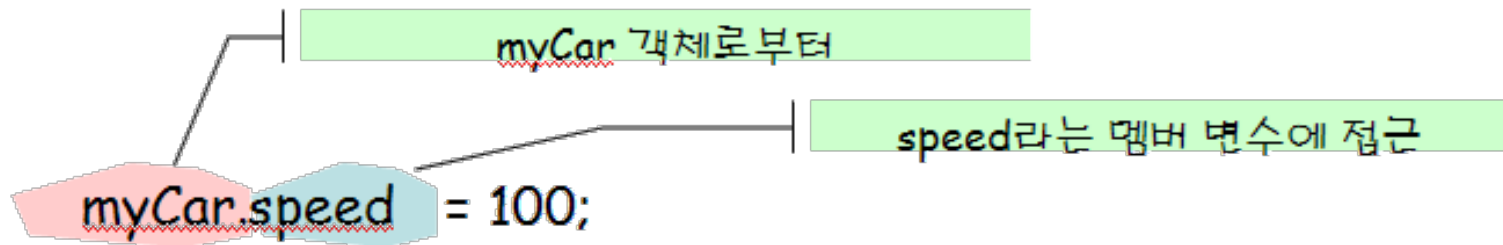
`Car myCar;`





# 객체의 사용

- 객체를 이용하여 멤버에 접근할 수 있다.



이들 멤버 변수에 접근하기 위해서는 도트(.) 연산자를 사용한다.

```
myCar.speed = 100;
```

```
myCar.speedUp();
```

```
myCar.speedDown();
```



# 예제



```
#include <iostream>
#include <string>
using namespace std;
```

```
class Car {
public:
    // 멤버 변수 선언
    int speed; // 속도
    int gear; // 기어
    string color; // 색상
```

```
    // 멤버 함수 선언
    void speedUp() { // 속도 증가 멤버 함수
        speed += 10;
    }
```

```
    void speedDown() { // 속도 감소 멤버 함수
        speed -= 10;
    }
```

```
};
```



객체(object)

속성 또는 상태(state)

동작 또는 행위(behavior):

출발, 정지, 가속, 감속, 방향 전환

구분		2.0 VVT
전장 (mm)		4,805
전폭 (mm)		1,775
전고 (mm)		1,480
축거	전 (mm)	1,545(1,550)
	후 (mm)	1,540(1,525)
축간거리 (mm)		2,650
연진형식		2.0 VVT
배기량 (cc)		1,975
최고출력 (ps/rpm)		143/5,000 (M/T)
		134/6,000 (A/T)
최대토크 (kg m/rpm)		19.0/4,000 (M/T)
		18.4/4,000 (A/T)
연료소비율 (ℓ)		59





```
Car globalCar;  
int main()  
{
```

```
    Car localCar;
```

```
    globalCar.speed = 100;;  
    localCar.speed = 60;  
    localCar.color = "white";
```

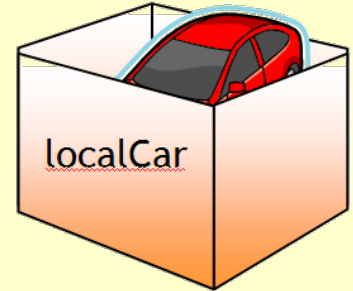
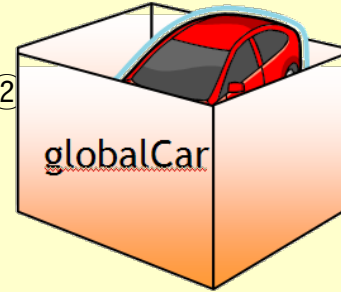
```
    cout << "현재 global 차의 속도는 " << globalCar.speed << endl;  
    cout << "현재 local 차의 속도는 " << localCar.speed << endl;
```

```
    return 0;
```

```
}
```

// ① 전역 객체

// ②



현재 global 차의 속도는 100  
현재 local 차의 속도는 60  
계속하려면 아무 키나 누르십시오 . . .



## 중간 점검 문제

1. 객체들을 만드는 설계도에 해당되는 것이 class이다.
2. 같은 종류의 객체가 여러 개 생성될 때 각 객체의 변수와 멤버 함수는 공유되는가? 아니면 각 객체마다 별도로 만들어지는가?
3. 클래스 선언 시에 클래스 안에 포함되는 것은 멤버 변수과 멤버 함수이다.
4. 객체의 멤버에 접근하는데 사용되는 연산자는 .이다.
5. 각 객체마다 별도로 가지고 있는 것은 클래스의 인스턴스이다..





# 접근 제어

클래스

*private*



선형 멤버: 클래스 안에서만 사용 가능

*public*



공용 멤버: 어디서든지 객체를 통하여 사용 가능



# private와 public

- 아무것도 지정하지 않으면 디폴트로 private

```
class Car {  
    // 멤버 변수 선언  
    int speed; // 속도  
    int gear;  // 기어  
    string color; // 색상  
}
```

private : {}를 넘어가면 접근할 수 없도록 설정.

```
private:  
    int speed;  
public:  
    void speedUp() { speed = 10;} (O)  
( {} 안에서 값을 접근하는 것은 가능)
```

```
int main()  
{  
    Car myCar;  
    myCar.speed = 100; // 오류!  
}
```



# 예제

```
#include <iostream>
#include <string>
using namespace std;
class Employee {
    string name;    // private 로 선언
    int salary;     // private 로 선언
    int age;        // private 로 선언
    // 직원의 월급을 반환
    int getSalary() { return salary;    }
public:
    // 직원의 나이를 반환
    int getAge() { return age;    }
    // 직원의 이름을 반환
    string getName() { return name;    }
}; 클래스가 끝나면 무조건 ; 넣어야함
int main()
{
    Employee e;
    e.salary = 300; // 오류! private 변수
    e.age = 26;     // 오류! private 변수
    int sa = e.getSalary(); // 오류! private 멤버 함수
    string s = e.getName(); // OK!
    int a = e.getAge();    // OK
}
```



# 멤버 변수

- 멤버 변수: 클래스 안에서 그러나 멤버 함수 외부에서 정의되는 변수



```
class Date {  
public:  
    void printDate() {  
  
        cout << year << "." << month << "." << day << endl;  
    }  
  
    int getDay() {  
        return day;  
    }  
  
    // 멤버 변수 선언  
    int year;  
    string month;  
    int day;  
}
```

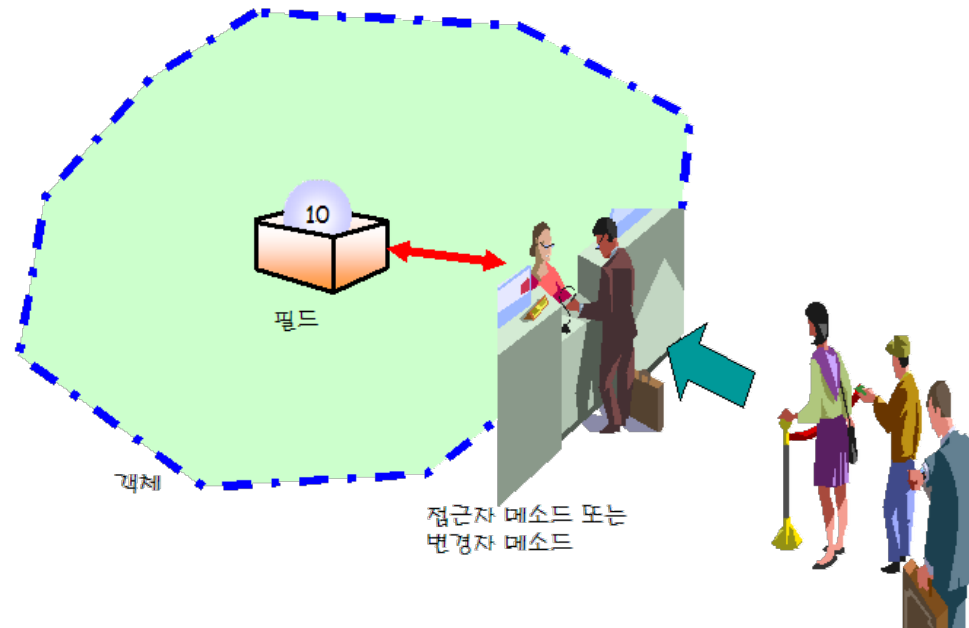
선언 위치와는 상관없이 어디서나  
사용이 가능하다.



# 접근자와 설정자

- 접근자(accessor): 멤버 변수의 값을 반환  
(예) `getBalance()`
- 설정자(mutator): 멤버 변수의 값을 설정  
(예) `setBalance()`;

public 설정





# 예제



```
class Car {  
private:  
    // 멤버 변수 선언  
    int speed;        //속도  
    int gear;         //기어  
    string color;     //색상  
    ...  
public:  
    // 접근자 선언  
    int getSpeed() {  
        return speed;  
    }  
    // 설정자 선언  
    void setSpeed(int s) {  
        speed = s;  
    }  
}
```





# 예제



```
// 접근자 선언
int getGear() {
    return gear;
}
// 변경자 선언
void setGear(int g) {
    gear = g;
}
// 접근자 선언
string getColor() {
    return color;
}
// 변경자 선언
void setColor(string c) {
    color = c;
}
};
```

const 필수



# 접근자와 설정자의 장점

- 설정자의 매개 변수를 통하여 잘못된 값이 넘어오는 경우, 이를 사전에 차단할 수 있다.
- 멤버 변수값을 필요할 때마다 계산하여 반환할 수 있다.
- 접근자만을 제공하면 자동적으로 읽기만 가능한 멤버 변수를 만들 수 있다.

```
void setSpeed(int s)
{
    if( s < 0 )
        speed = 0;
    else
        speed = s;
}
```



# 멤버 함수의 외부 정의

클래스 내부에 멤버  
함수 정의

```
class Car
{
public:
    int speed;
    ...
    int getSpeed(){
        return speed;
    }
    ...
}
```

클래스 내부에 함수  
원형을 써준다

```
class Car
{
public:
    int speed;
    ...
    int getSpeed();
    ...
    ...
}
```

클래스 외부에 함수  
를 정의한다.

```
int Car::getSpeed(){
    return speed;
}
```



# 내부 정의와 외부 정의의 차이

- 멤버 함수가 클래스 내부에 정의되면 자동적으로 인라인(inline) 함수가 된다.
- 멤버 함수가 클래스 외부에 정의되면 일반적인 함수와 동일하게 호출한다.



# 예제



```
#include <iostream>
using namespace std;

class Car {
public:
    int getSpeed();
    void setSpeed(int s);
    void honk();
private:
    int speed;           //속도
};

int Car::getSpeed()
{
    return speed;
}

void Car::setSpeed(int s)
{
    speed = s;
}
```



# 예제



```
void Car::honk()
{
    cout << "빵 빵!" << endl;
}

int main()
{
    Car myCar;
    myCar.setSpeed(80);
    myCar.honk();
    cout << "현재 속도는 " << myCar.getSpeed() << endl;
    return 0;
}
```



빵 빵!  
현재 속도는 **80**  
계속하려면 아무 키나 누르십시오 ...



# 멤버 함수 중복 정의



```
#include <iostream>
using namespace std;
```

```
class Car {
public:
    void setSpeed();
    void setSpeed(int s);
private:
    int speed;           //속도
};
```

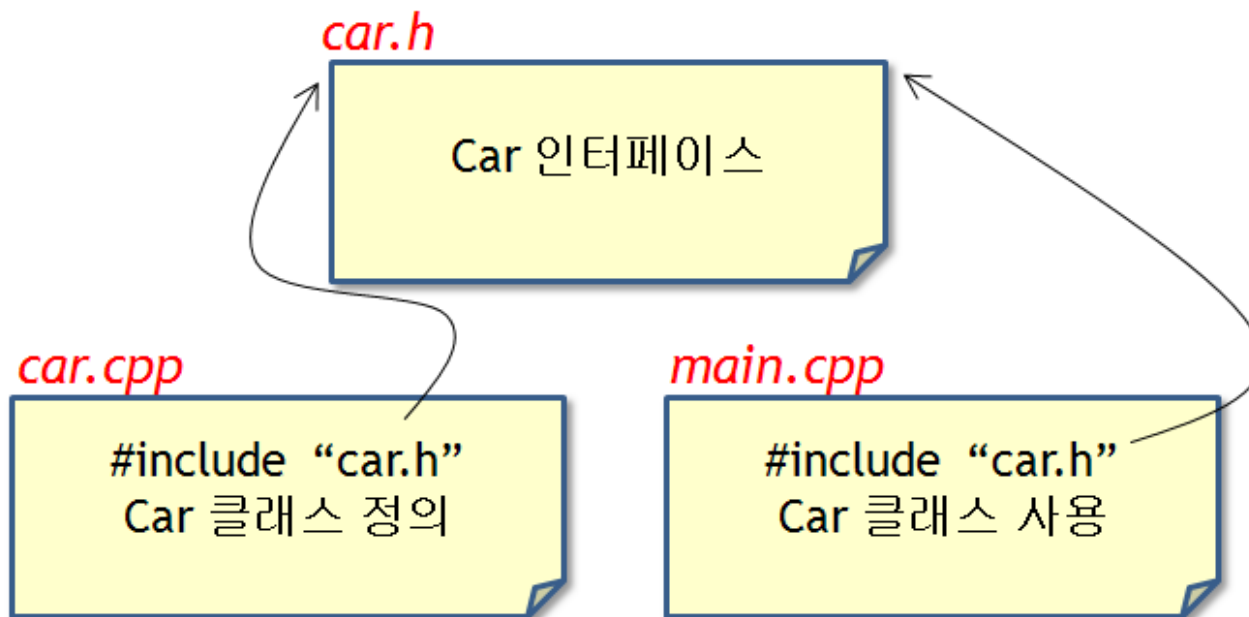
```
void Car::setSpeed()
{
    speed = 0;
}
void Car::setSpeed(int s)
{
    speed = s;
}
```

중복 정의



# 클래스 선언과 구현의 분리

- 클래스의 선언과 구현을 분리하는 것이 일반적







# 예제



car.h

```
class Car {  
public:  
    int getSpeed();  
    void setSpeed(int s);  
    void honk();  
private:  
    int speed;           //속도  
};
```

클래스를 선언한다.



# 예제



car.cpp

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

int Car::getSpeed()
{
    return speed;
}
void Car::setSpeed(int s)
{
    speed = s;
}
void Car::honk()
{
    cout << "뽕뽕!" << endl;
}
```

클래스를 정의한다.



# 예제



main.cpp

```
#include <iostream>
#include "car.h"           // 현재 위치에 car.h를 읽어서 넣으라는 것을 의
                           // 미한다.
using namespace std;

int main()
{
    Car myCar;
    myCar.setSpeed(80);
    myCar.honk();
    cout << "현재 속도는 " << myCar.getSpeed() << endl;
    return 0;
}
```

클래스를 사용한다.



# 멤버 함수 예제



```
#include <iostream>
#include <string>
#include <math>
using namespace std;

class DiceGame {
    int diceFace;
    int userGuess;

    void RollDice()
    {
        diceFace = (int)(rand() * 6) + 1;
    }
    int getUserInput(string prompt)
    {
        int r;
        cout << prompt;
        cin >> r;
        return r;
    }
}
```



# 멤버 함수 예제



```
void checkUserGuess()
{
    if( diceFace == userGuess )
        cout << "맞았습니다";
    else
        cout << "틀렸습니다";
}
public:
void startPlaying()
{
    userGuess = getUserInput("예상값을 입력하시오: ");
    RollDice();
    checkUserGuess();
}
};
int main()
{
    DiceGame game;
    game.startPlaying();
}
```



예상값을 입력하시오:  
3  
틀렸습니다



## 중간 점검 문제

1. 멤버 함수 안에서 `private` 멤버 변수를 사용할 수 있는가? ○
2. 멤버 함수는 클래스의 외부에서 정의될 수 있는가? ○
3. 멤버 함수는 별도의 소스 파일에서 정의될 수 있는가? ○





# UML

- UML(Unified Modeling Language)

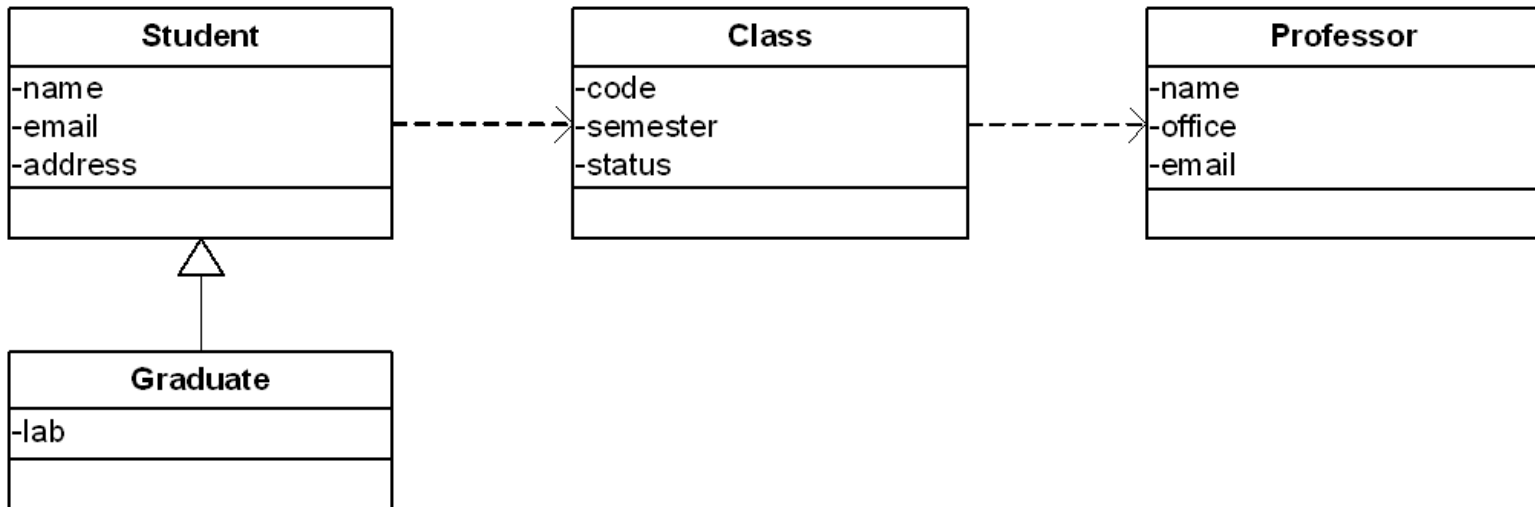

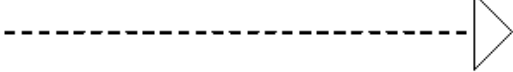

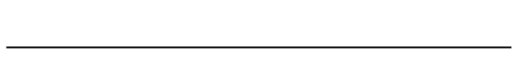

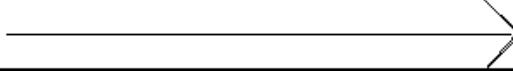


그림 8.9 UML의 예



# 클래스와 클래스의 관계

상속( <b>inheritance</b> )	
인터페이스 상속( <b>interface inheritance</b> )	
의존( <b>dependency</b> )	
집합( <b>aggregation</b> )	
연관( <b>association</b> )	
유향 연관( <b>direct association</b> )	

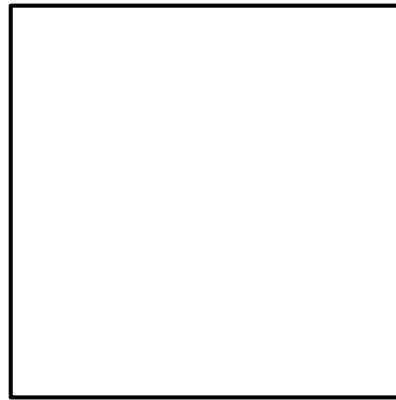
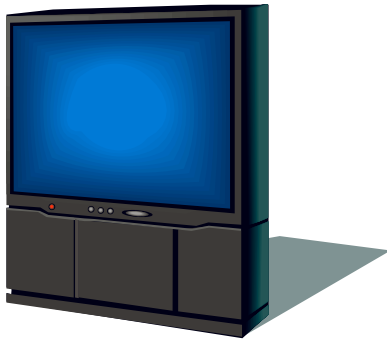




## 중간점검



1. TV를 나타내는 클래스를 정의하고 UML의 클래스 다이어그램으로 표현하여 보라.





# 구조체

- 구조체(structure) = 클래스

```
struct BankAccount { // 은행 계좌
    int accountNumber; // 계좌 번호
    int balance; // 잔액을 표시하는 변수
    double interest_rate; // 연이자
    double get_interrest(int days){
        return (balance*interest_rate)*((double)days/365.0);
    }
};
```

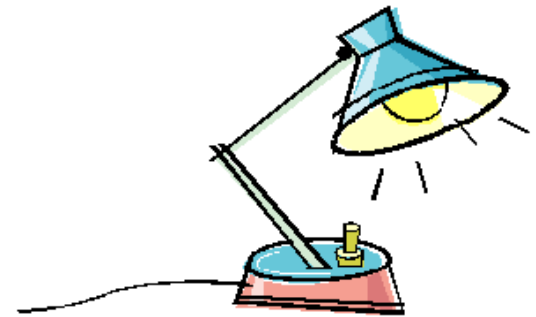
모든 멤버가 디폴트로 **public**이 된다.



# 예제

집에서 사용하는 데스크 램프를 클래스로 작성하여 보면 다음과 같다.

DeskLamp
-isOn : bool
+turnOn() +turnOff()





# 예제



```
#include <iostream>
#include <string>
using namespace std;

class DeskLamp {
private:
    // 인스턴스 변수 정의
    bool isOn;                // 켜짐이나 꺼짐과 같은 램프의 상태

public:
    // 멤버 함수 선언
    void turnOn(); // 램프를 켜다.
    void turnOff(); // 램프를 끄다.
    void print();   // 현재 상태를 출력
};

void DeskLamp::turnOn()
{
    isOn = true;
}
```



# 예제



```
void DeskLamp::turnOff()
{
    isOn = false;
}
void DeskLamp::print()
{
    cout << "램프가 " << (isOn == true ? "켜짐" : "꺼짐") << endl;
}
int main()
{
    // 객체 생성
    DeskLamp lamp;

    // 객체의 멤버 함수를 호출하려면 도트 연산자인 .을 사용한다.
    lamp.turnOn();
    lamp.print();
    lamp.turnOff();
    lamp.print();
    return 0;
}
```

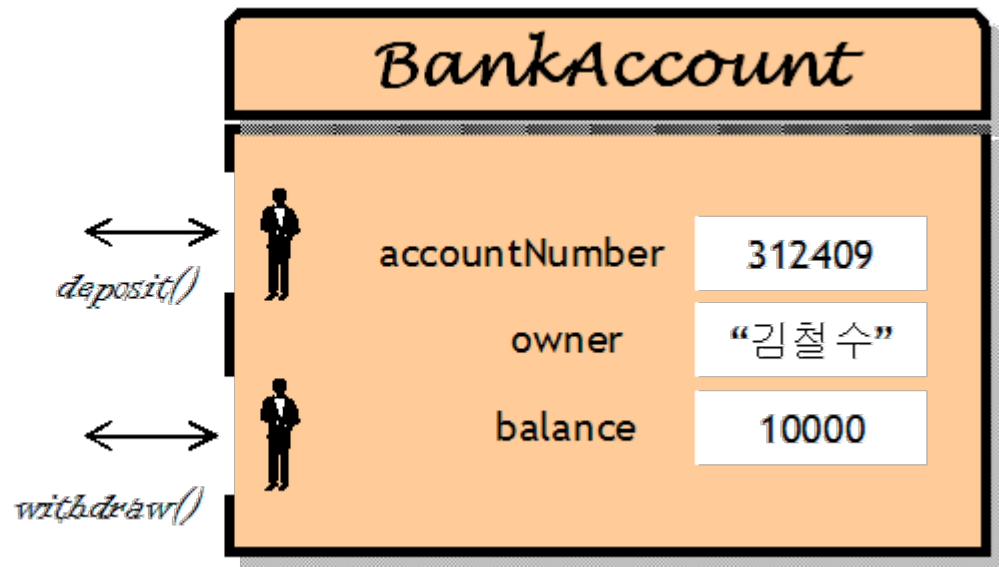


램프가 켜짐  
램프가 꺼짐



# 예제

- 은행 계좌





# 예제



```
#include <iostream>
#include <string>
using namespace std;

class BankAccount {           // 은행 계좌
private:
    int accountNumber;        // 계좌 번호
    string owner;              // 예금주
    int balance;               // 잔액을 표시하는 변수

public:
    void setBalance(int amount); // balance에 대한 설정자
    int getBalance();            // balance에 대한 접근자
    void deposit(int amount);    // 저금 함수
    void withdraw(int amount);   // 인출 함수
    void print();                // 현재 상태 출력
};

void BankAccount::setBalance(int amount)
{
    balance = amount;
}
```



# 예제



```
int BankAccount::getBalance()
{
    return balance;
}
void BankAccount::deposit(int amount)
{
    balance += amount;
}
void BankAccount::withdraw(int amount)
{
    balance -= amount;
}
void BankAccount::print()
{
    cout << "잔액은 " << balance << "입니다." << endl;
}
int main() {
    BankAccount account;
    account.setBalance(0);
    account.deposit(10000);
    account.print();
    account.withdraw(8000);
    account.print();
    return 0;
}
```



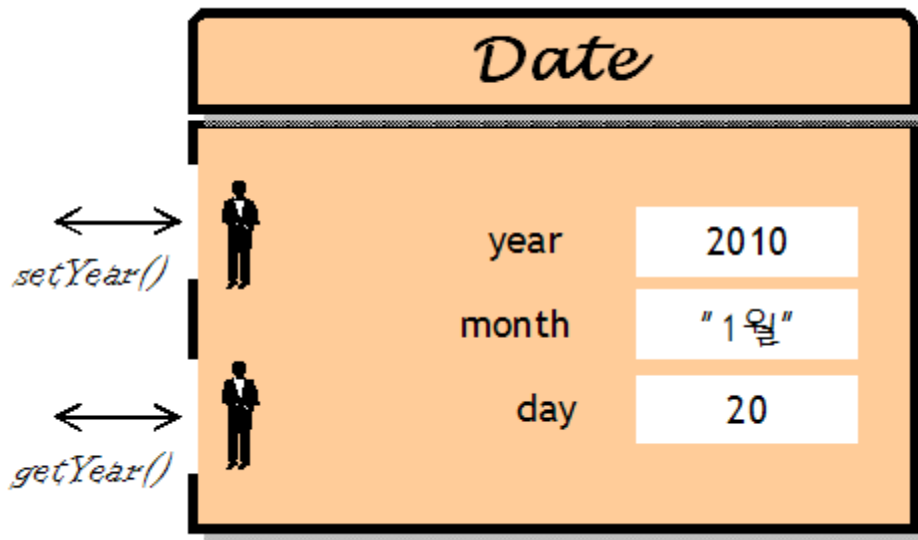
잔액은 10000입니다.  
잔액은 2000입니다.





# 예제

- 날짜





# 예제



```
#include <iostream>
#include <string>
using namespace std;

class Date {
private:
    int year;
    int month;
    int day;
public:
    int getYear();
    void setYear(int y);
    int getMonth();
    void setMonth(int m);
    void setDay(int d);
    int getDay();
    void print();
};

int Date::getYear()
{
    return year;
}
```



# 예제



```
void Date::setYear(int y)
{   year = y;   }
int Date::getMonth()
{   return month;   }
void Date::setMonth(int m)
{   month = m;   }
int Date::getDay()
{   return day;   }
void Date::setDay(int d)
{   day = d;   }
void Date::print()
{
    cout << year << "년 " << month << "월 " << day << "일" << endl;
}
int main()
{
    Date date;
    date.setYear(2010);
    date.setMonth(1);
    date.setDay(20);
    date.print();
    return 0;
}
```

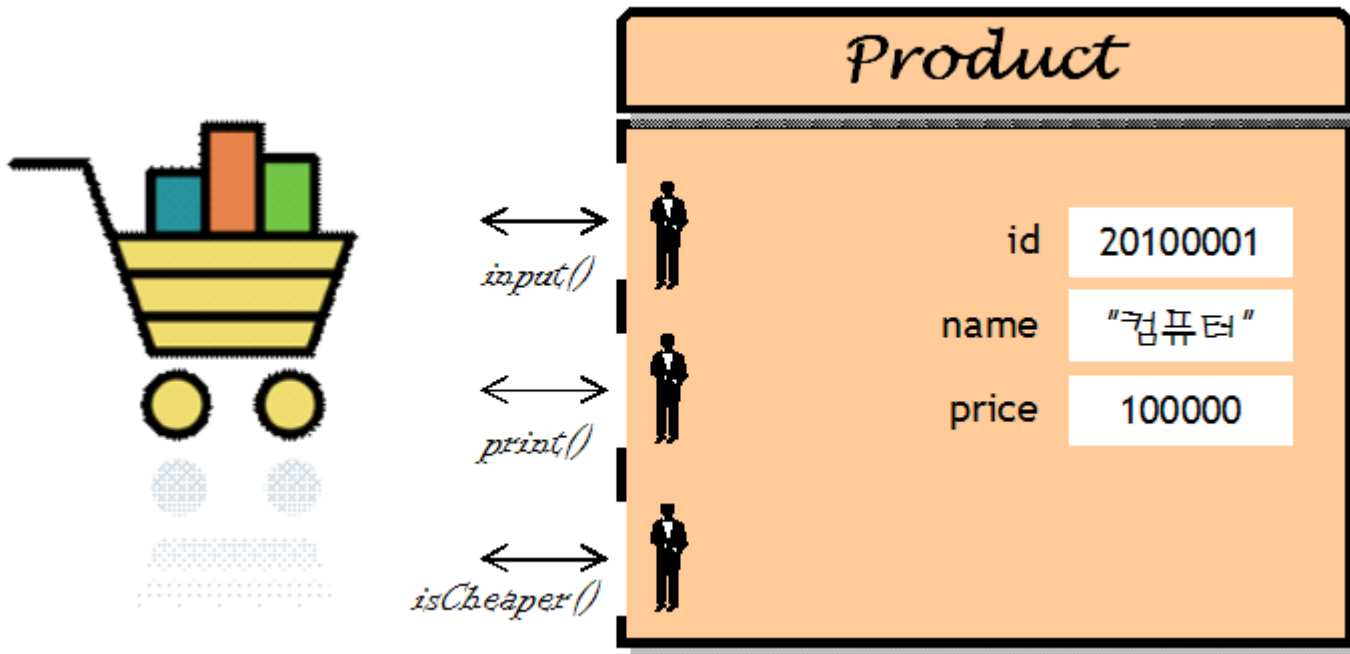


2010년 1월 20일



# 예제

- 상품





# 예제



```
#include <iostream>
#include <string>
using namespace std;

class Product {
private:
    int id;
    string name;
    int price;
public:
    void input();
    void print();
    bool isCheaper(Product other);
};

void Product::input()
{
    cout << "상품의 일련 번호: ";
    cin >> id;
    cout << "상품의 이름: ";
    cin >> name;
    cout << "상품의 가격: ";
    cin >> price;
}
```



# 예제

```
void Product::print()
{
    cout << " 상품 번호 " << id << endl
         << " 상품의 이름: " << name
         << " 상품의 가격: " << price << endl;
}

bool Product::isCheaper(Product other)
{
    if( price < other.price )
        return true;
    else
        return false;
}
```



# 예제



```
int main()
{
    Product p1, p2;
    p1.input();
    p2.input();
    if( p1.isCheaper(p2) ){
        p1.print();
        cout << "이 더 쌉니다\n";
    }
    else {
        p2.print();
        cout << "이 더 쌉니다\n";
    }
    return 0;
}
```



상품의 일련 번호: 1  
상품의 이름: 모니터  
상품의 가격: 100000  
상품의 일련 번호: 2  
상품의 이름: 컴퓨터  
상품의 가격: 20000000  
상품 번호 1  
상품의 이름: 모니터 상품의 가격: 100000  
이 더 쌉니다



# Q & A

