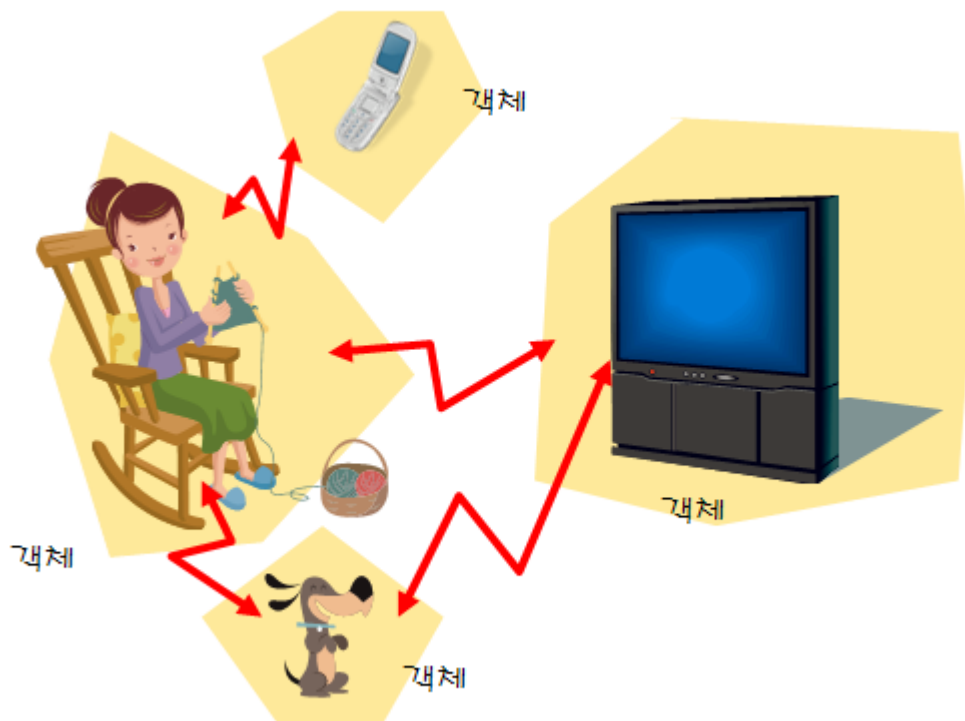




*C++ Espresso*

## 제8장 상속





# 이번 장에서 학습할 내용



- 상속이란?
- 접근 제어 지정자
- 상속에서의 생성자와 소멸자
- 재정의 (오버라이딩)
- 다중 상속

상속을 코드를  
재사용하기  
위한 중요한  
기법입니다.





# 상속이란?

- 상속: 기존에 존재하는 유사한 클래스로부터 속성과 동작을 이어받고 자신이 필요한 기능을 추가하는 기법



상속



상속을 이용하면 쉽게  
재산을 모을 수 있는  
것처럼 소프트웨어도  
쉽게 개발할 수 있다.



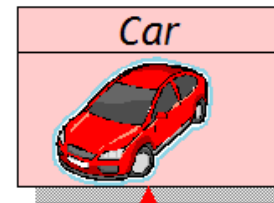
# 상속의 장점

- 상속의 장점
  - 상속을 통하여 기존 클래스의 필드와 메소드를 재사용
  - 기존 클래스의 일부 변경도 가능
  - 상속을 이용하게 되면 복잡한 **GUI** 프로그램을 순식간에 작성
- 상속은 이미 작성된 검증된 소프트웨어를 재사용
- 신뢰성 있는 소프트웨어를 손쉽게 개발, 유지 보수
- 코드의 중복을 줄일 수 있다.



# 상속

```
class Car
{
    int speed;
}
class SportsCar : public Car
{
    bool turbo;
}
```



수퍼 클래스(superclass)

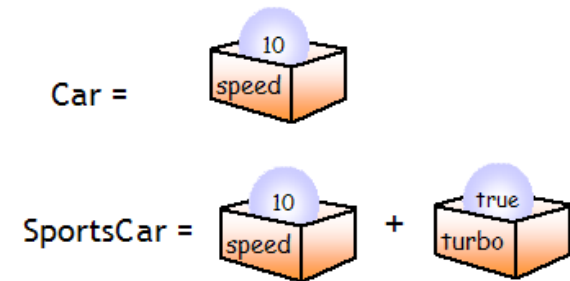
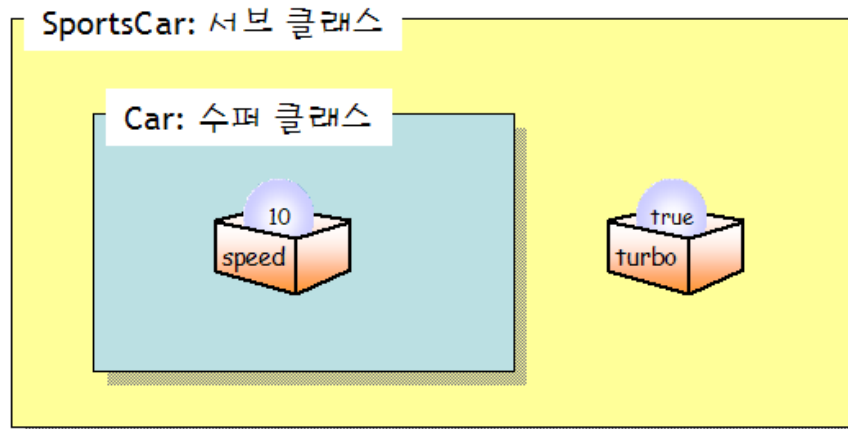


서브클래스(subclass)

상속한다는 의미



# 자식 클래스는 부모 클래스를 포함





# 상속의 예

수퍼 클래스	서브 클래스
Animal(동물)	Lion(사자), Dog(개), Cat(고양이)
Bike(자전거)	MountainBike(산악자전거)
Vehicle(탈것)	Car(자동차), Bus(버스), Truck(트럭), Boat(보트), Motorcycle(오토바이), Bicycle(자전거)
Student(학생)	GraduateStudent(대학원생), UnderGraduate(학부생)
Employee(직원)	Manager(관리자)
Shape(도형)	Rectangle(사각형), Triangle(삼각형), Circle(원)

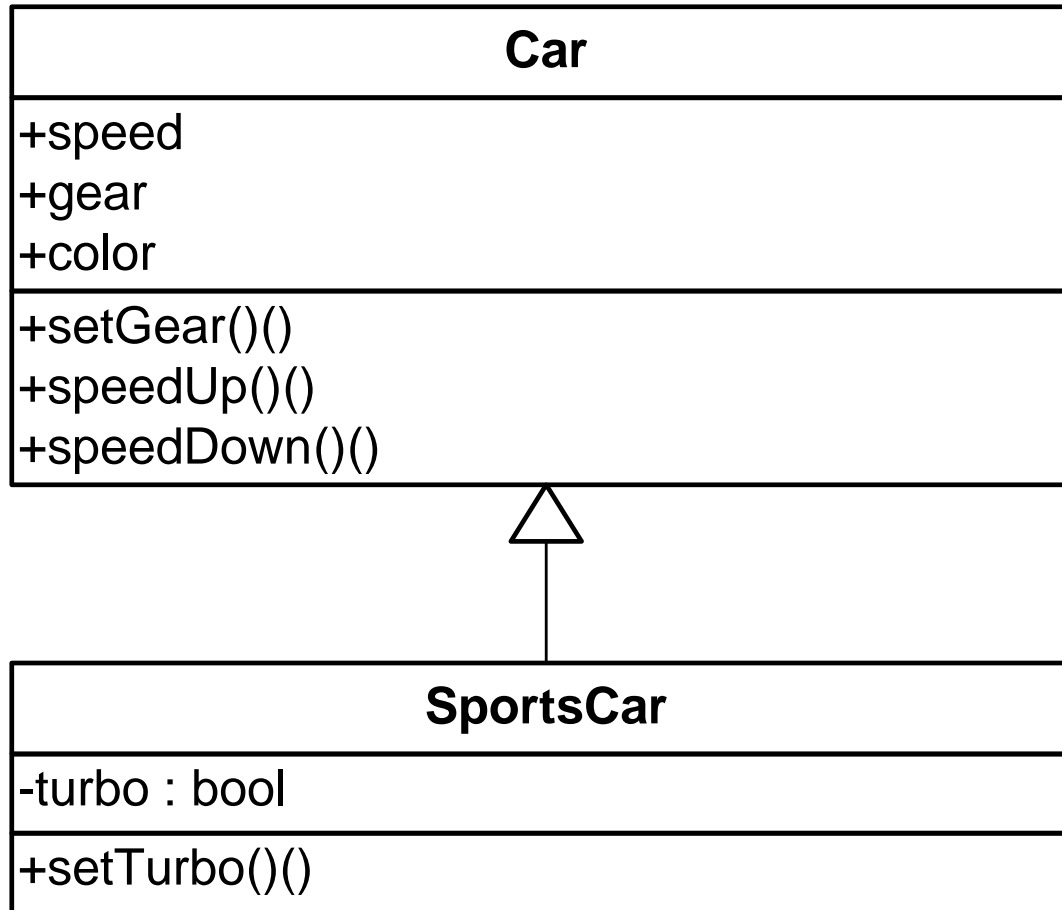


## 참고

- 수퍼 클래스 == 부모 클래스(parent class) == 베이스 클래스(base class)
- 서브 클래스 == 자식 클래스(child class) == 파생된 클래스(derived class)



# 상속의 예제







# Car 클래스

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

class Car {
public:
    // 3개의 멤버변수선언
    int speed; // 속도
    int gear; // 주행거리
    string color; // 색상

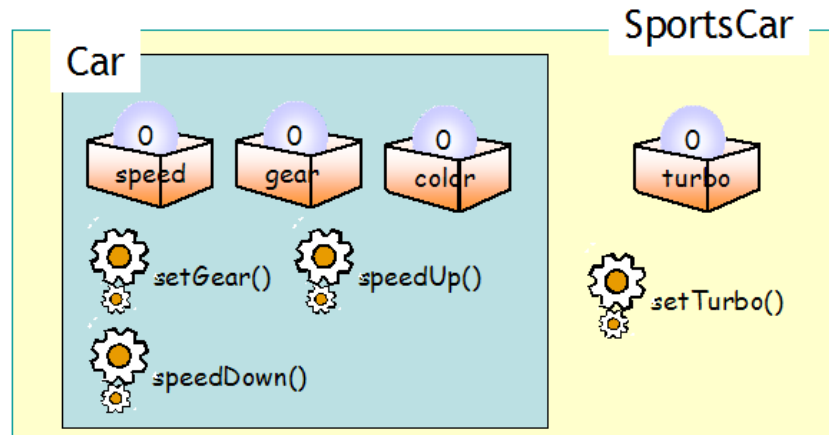
    // 3개의 멤버함수선언
    void setGear(int newGear) { // 기어설정멤버함수
        gear = newGear;
    }
    void speedUp(int increment) { // 속도증가멤버함수
        speed += increment;
    }
    void speedDown(int decrement) { // 속도감소멤버함수
        speed -= decrement;
    }
};
```



# SportsCar 클래스



```
// Car 클래스를 상속받아 다음과 같이 SportsCar 클래스를 작성하여보자.  
class SportsCar : public Car { // Car를 상속받는다.  
    // 1개의 멤버변수를 추가  
    bool turbo;  
  
public:  
    // 1개의 멤버함수를 추가  
    void setTurbo(bool newValue) { // 터보모드 설정 멤버함수  
        turbo = newValue;  
    }  
};
```





# SportsCar 클래스



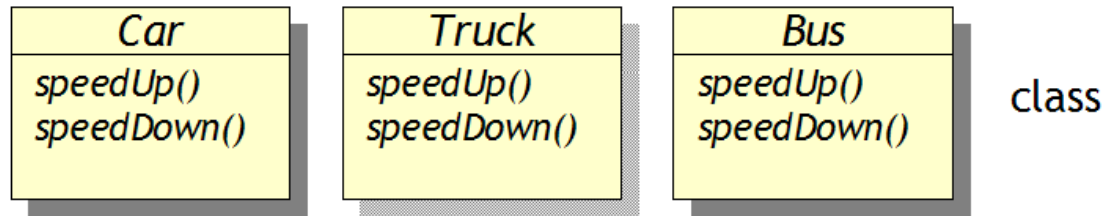
```
int main()
{
    SportsCar c;
    c.color = "Red";
    c.setGear(3);
    c.speedUp(100);
    c.speedDown(30);
    c.setTurbo(true);
    return 0;
}
```

// 부모클래스멤버변수접근  
// 부모클래스멤버함수호출  
// 부모클래스멤버함수호출  
// 부모클래스멤버함수호출  
// 자체멤버함수호출

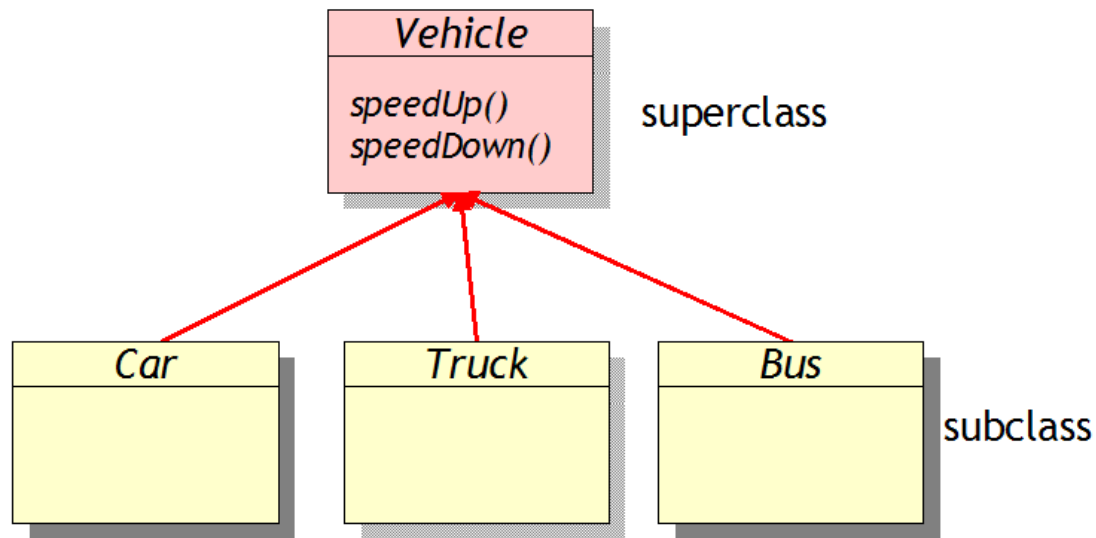
자식 클래스는 부모 클래스의 변수와 함수를 마치 자기 것처럼 사용할 수 있다.



# 상속은 왜 필요한가?



각 클래스에 코드가 중복된다.

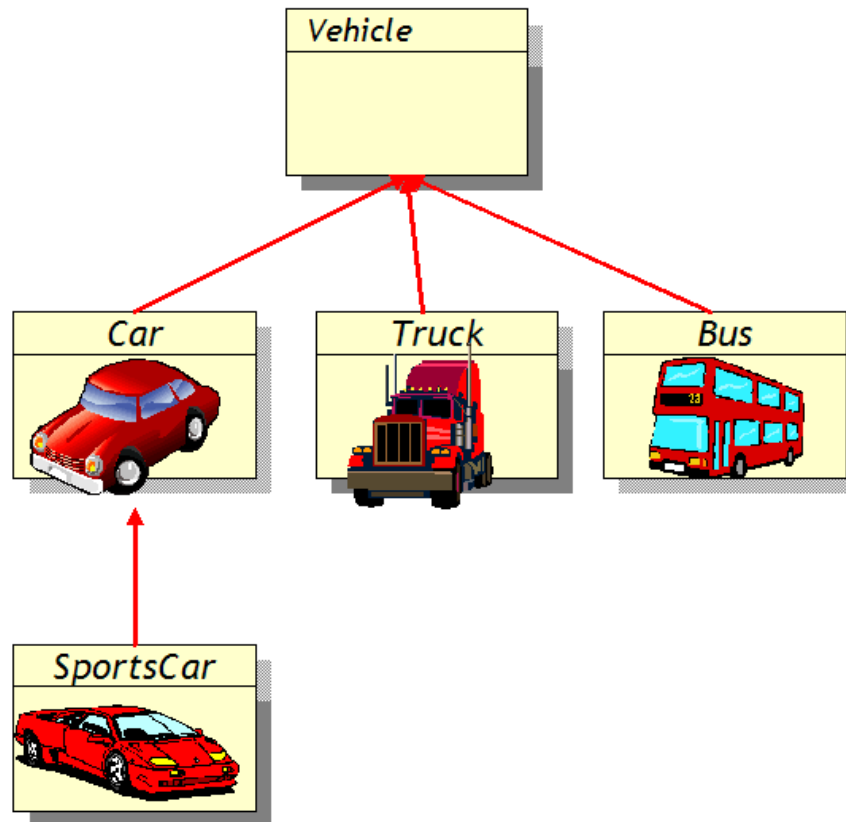


중복되는 코드는 수퍼 클래스에 모은다.



# 상속 계층도

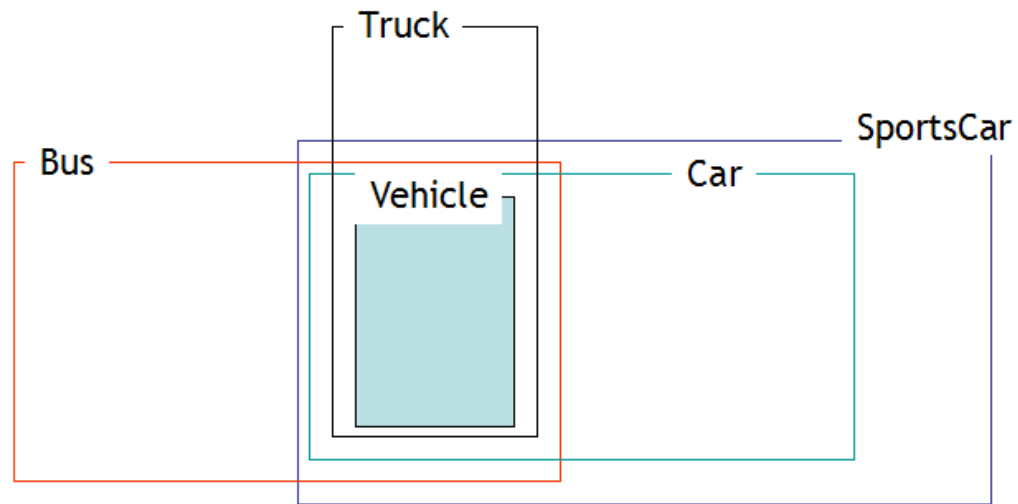
- 상속은 여러 단계로 이루어질 수 있다.





# 상속 계층도

```
class Vehicle { ... }  
class Car : public Vehicle { ... }  
class Truck : public Vehicle { ... }  
class Bus : public Vehicle { ... }  
class SportsCar : public Car { ... }
```





# 상속은 is-a 관계

- 상속은 is-a 관계
  - 자동차는 탈것이다. (*Car is a Vehicle*).
  - 사자, 개, 고양이는 동물이다.
- has-a(포함) 관계는 상속으로 모델링을 하면 안 된다.
  - 도서관은 책을 가지고 있다(*Library has a book*).
  - 거실은 소파를 가지고 있다.



# 접근 제어 지정자

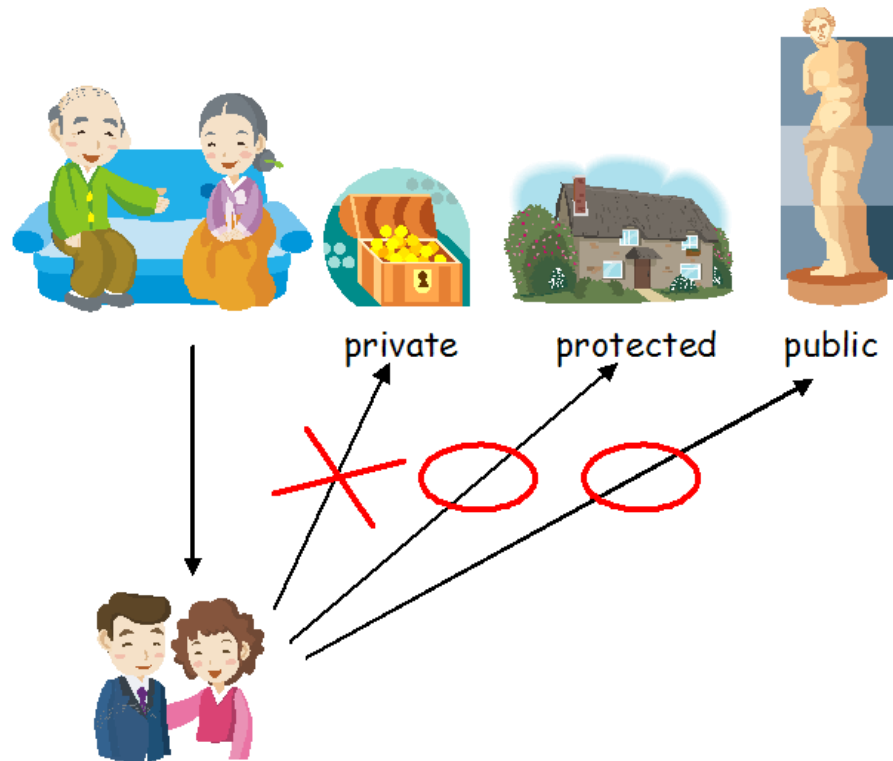


그림 13.9 상속에서의 접근 지정자





# 접근 제어 지정자

접근 지정자	현재 클래스	자식 클래스	외부
private	○	×	×
protected	○	○	×
public	○	○	○



# 예제

```
#include <iostream>
#include <string>

using namespace std;
class Employee {
    int rrn;           // Regident Resgistration Number: 주민등록번호

protected:
    int salary;       // 월급

public:
    string name;      // 이름
    void setSalary(int salary);
    int getSalary();
};

void Employee::setSalary(int salary) {
    this->salary = salary;
}

int Employee::getSalary() {
    return salary;
}
```



# 예제

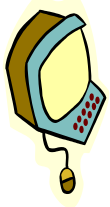
```
class Manager : public Employee {  
    int bonus;  
  
public:  
    Manager(int b=0) : bonus(b) { }  
    void modify(int s, int b);  
    void display();  
};  
  
void Manager::modify(int s, int b) {  
    salary = s;           // 부모클래스의 보호멤버 사용 가능!  
    bonus = b;  
}  
  
void Manager::display()  
{  
    cout << "봉급: " << salary << " 보너스: " << bonus << endl;  
    // cout << "주민등록번호: " << rrn << endl;    //  
}
```



# 예제



```
int main()
{
    Manager m;
    m.setSalary(2000);
    m.display();
    m.modify(1000, 500);
    m.display();
}
```



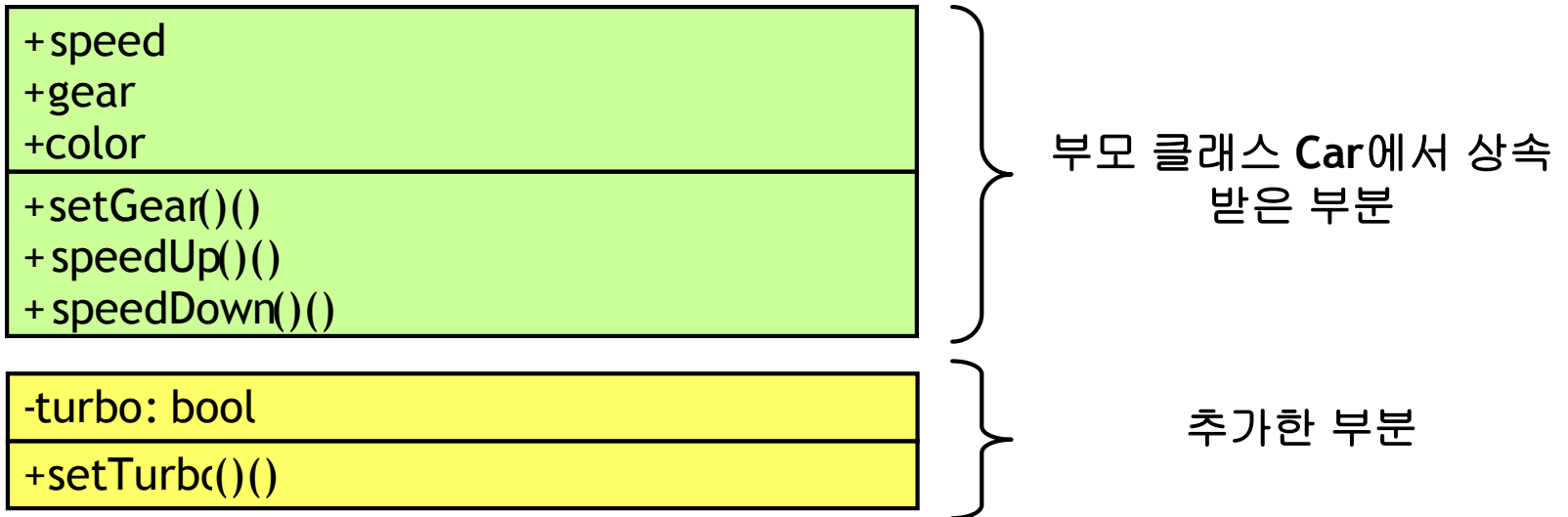
봉급: 1000 보너스: 500  
계속하려면 아무 키나 누르십시오 . . .



# 상속에서의 생성자와 소멸자

- 자식 클래스의 객체가 생성될 때에 당연히 자식 클래스의 생성자는 호출된다. 이때에 부모 클래스 생성자도 호출될까?

## SportsCar





# 상속에서의 생성자와 소멸자

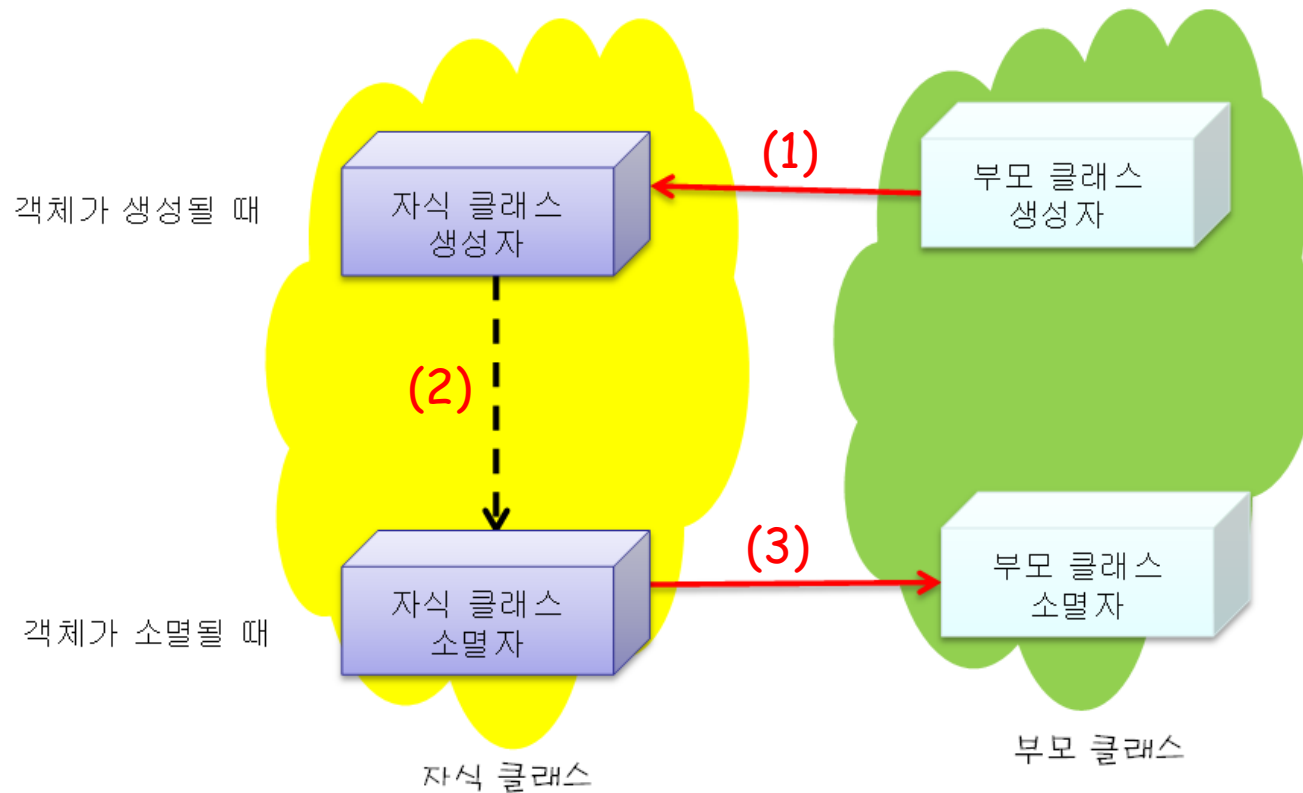


그림 8.10 상속에서 생성자와 소멸자의 호출



# 예제



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

```
class Shape {
    int x, y;
```

```
public:
```

```
    Shape() {
        cout << "Shape 생성자()" << endl;
    }
```

```
    ~Shape() {
        cout << "Shape 소멸자()" << endl;
    }
```

```
};
```



# 예제

```
class Rectangle : public Shape {  
    int width, height;  
public:  
    Rectangle(){  
        cout << "Rectangle 생성자()" << endl;  
    }  
    ~Rectangle(){  
        cout << "Rectangle 소멸자()" << endl;  
    }  
};  
int main()  
{  
    Rectangle r;  
    return 0;  
}
```

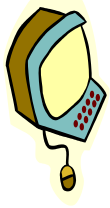
Shape 생성자()

Rectangle 생성자()

Rectangle 소멸자()

Shape 소멸자()

계속하려면 아무 키나 누르십시오 . . .







# 부모 생성자의 호출

```
Rectangle(int x=0, int y=0, int w=0, int h=0) : Shape(x, y)
```

```
{
```

```
    width = w;
```

```
    height = h;
```

```
}
```

부모클래스의  
생성자호출



# 예제



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

class Shape {
    int x, y;
public:
    Shape() {
        cout << "Shape 생성자()" << endl;
    }
    Shape(int xloc, int yloc) : x(xloc), y(yloc){
        cout << "Shape 생성자(xloc, yloc)" << endl;
    }
    ~Shape() {
        cout << "Shape 소멸자()" << endl;
    }
};
```



# 예제



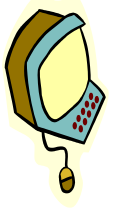
```
class Rectangle : public Shape {
    int width, height;
public:
    Rectangle(int x=0, int y=0, int w=0, int h=0);
    ~Rectangle(){
        cout << "Rectangle 소멸자()" << endl;
    }
};
Rectangle::Rectangle(int x, int y, int w, int h) : Shape(x, y) {
    width = w;
    height = h;
    cout << "Rectangle 생성자(x, y, w, h)" << endl;
}
```



# 예제



```
int main()
{
    Rectangle r(0, 0, 100, 100);
    return 0;
}
```

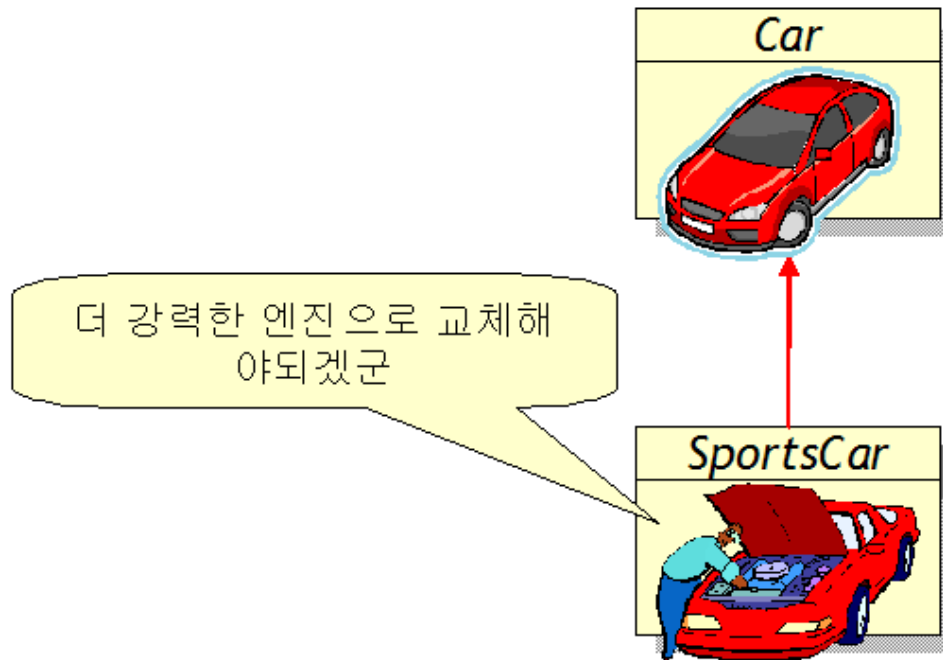


Shape 생성자(xloc, yloc)  
Rectangle 생성자(x, y, w, h)  
Rectangle 소멸자()  
Shape 소멸자()  
계속하려면 아무 키나 누르십시오 . . .



# 멤버함수 재정의

- 멤버함수 재정의(**overriding**): 자식 클래스가 필요에 따라 상속된 멤버 함수를 다시 정의하는 것





# 예제



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

```
class Car {
public:
```

```
    int getHP()
    {
        return 100;
    }
```

// 100마력반환

```
};
```

```
class SportsCar : public Car {
public:
```

```
    int getHP()
    {
        return 300;
    }
```

// 300마력반환

```
};
```

재정의



# 예제



```
int main()
{
    SportsCar sc;
    cout << "마력: " << sc.getHP() << endl;
    return 0;
}
```



마력: 300  
계속하려면 아무 키나 누르십시오 . . .



# 예제

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

class Animal {
public:
    void speak()
    {
        cout << "동물이 소리를 내고 있음" << endl;
    }
};

class Dog : public Animal {
public:
    void speak()
    {
        cout << "멍멍!" << endl;
    }
};
```





# 예제

```
int main()
{
    Dog obj;
    obj.speak();
    return 0;
}
```





# 부모클래스의 멤버함수 호출(1)

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

class ParentClass {
public:
    void print() {
        cout << "부모 클래스의 print() 멤버 함수" << endl;
    }
};

class ChildClass : public ParentClass {
    int data;
public:
    void print() { //멤버 함수 재정의
        ParentClass::print();
        cout << "자식 클래스의 print() 멤버 함수 " << endl;
    }
};
```



## 부모클래스의 멤버함수 호출(2)



```
int main()
{
    SportsCar sc;
    cout << "마력: " << sc.Car::getHP() << endl;    // 100이 출력된다.
    return 0;
}
```



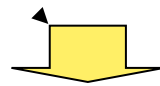
마력: 100  
계속하려면 아무 키나 누르십시오 . . .



# 재정의의 조건

- 부모 클래스의 멤버 함수와 동일한 시그니처를 가져야 한다.
- 즉 멤버 함수의 이름, 반환형, 매개 변수의 개수와 데이터 타입이 일치하여야 한다.

```
class Animal {  
    void makeSound()  
    {  
    }  
};
```



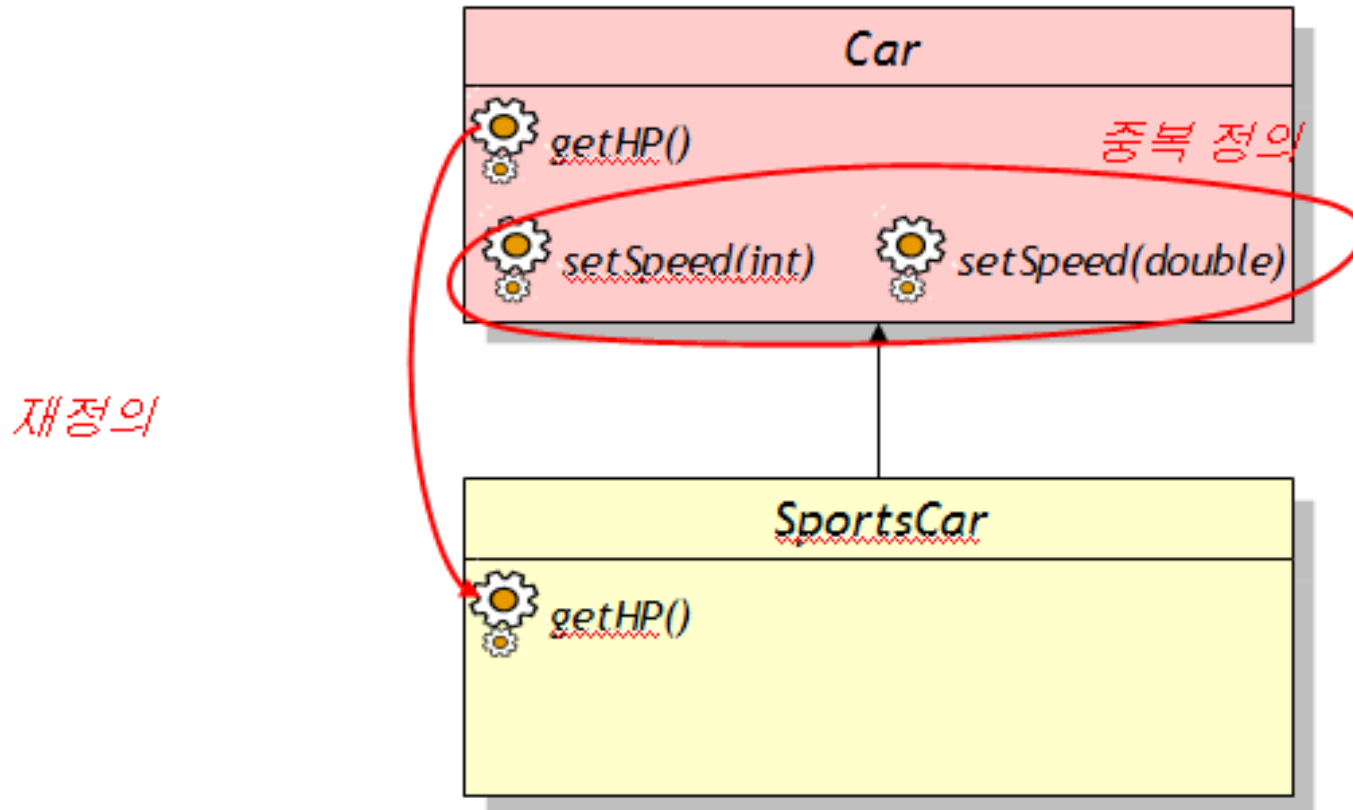
재정의가 아님

```
class Dog : public Animal {  
    int makeSound()  
    {  
    }  
};
```



# 재정의와 중복 정의

- 중복 정의: 같은 이름의 멤버 함수를 여러 개 정의하는 것  
시그니처 (매개변수의 개수, 자료형, 순서)가 다르다.
- 재정의: 부모 클래스에 있던 상속받은 멤버 함수를 다시 정의하는 것





# 멤버 변수 재정의



```
class Car {
```

```
public:
```

```
int speed;  
int gear;  
string color;
```

```
Car(): speed(0), gear(1), color("white") { }
```

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

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

```
};
```

재정의 -> 가능하지만 혼란을 일으킴!

```
class SportsCar : public Car {
```

```
public:
```

```
int speed;  
int gear;  
string color;
```

```
SportsCar(): speed(100), gear(3), color("blue") { }
```

```
};
```



# 비교



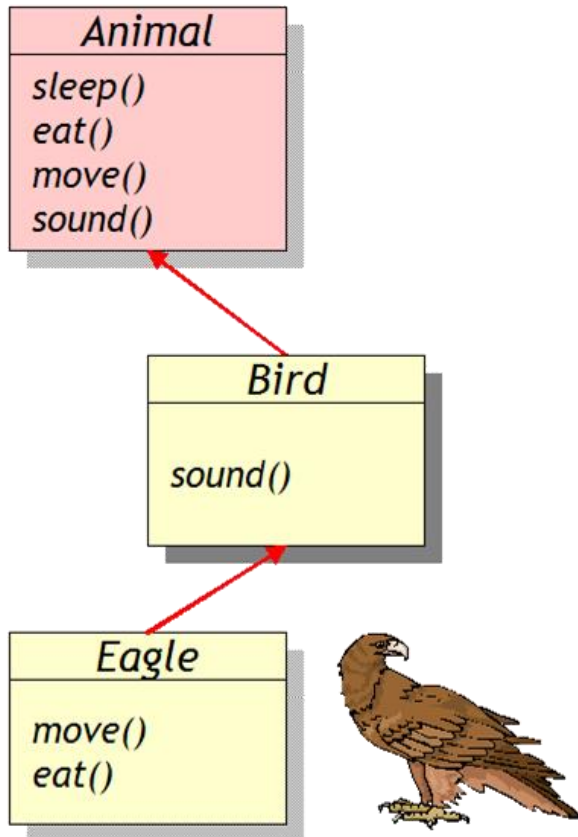
```
int main()
{
    SportsCar sc;
    cout << "스피드: " << sc.speed << endl;      // 자식 클래스의 speed
    cout << "스피드: " << sc.Car::speed << endl;   // 부모 클래스의 speed
    cout << "스피드: " << sc.getSpeed() << endl;   // 부모 클래스의 speed 반환
    return 0;
}
```



스피드: 100  
스피드: 0  
스피드: 0  
계속하려면 아무 키나 누르십시오 . . .



# 재정의된 멤버 함수의 호출 순서



Eagle e;

```
e.sleep(); // Animal의 sleep() 호출
e.eat();   // Eagle의 eat() 호출
e.sound(); // Bird의 sound() 호출
```





# 부모 클래스의 멤버 호출



```
class ParentClass {  
public:  
    void print() {  
        cout << "부모클래스의 print() 멤버함수" << endl;  
    }  
};
```

```
class ChildClass : public ParentClass {  
    int data;
```

```
public:
```

부모 클래스의 함수 호출!

```
void print() { //멤버함수오버라이딩
```

```
    ParentClass::print();
```

```
    cout << "자식클래스의 print() 멤버함수" << endl;
```

```
}
```

```
};
```

```
int main()
```

```
{
```

```
    ChildClass obj;
```

```
    obj.print();
```

```
    return 0;
```

```
}
```



# 예제



부모 클래스의 `print()` 멤버 함수  
자식 클래스의 `print()` 멤버 함수  
계속하려면 아무 키나 누르십시오 . . .



# 상속의 3가지 유형

```
class 자식클래스 : public 부모클래스
{
    ...
}
```

public으로 상속



	public으로 상속	protected로 상속	private로 상속
부모 클래스의 public 멤버	->public	->protected	->private
부모 클래스의 protected 멤버	->protected	->protected	->private
부모 클래스의 private 멤버	접근 안됨	접근 안됨	접근 안됨



# 예제



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

class ParentClass {
private:
    int x;
protected:
    int y;
public:
    int z;
};

class ChildClass1 : public ParentClass
{
};

class ChildClass2 : protected ParentClass
{
};

class ChildClass3 : private ParentClass
{
};
```



# 예제



```
int main()
{
    ChildClass1 obj1;
    ChildClass2 obj2;
    ChildClass3 obj3;

    cout << obj1.x << endl;      // 불가능
    cout << obj1.y << endl;      // 불가능: y는protected로유지된다.
    cout << obj1.z << endl;      // 가능: z는public으로유지된다.

    cout << obj2.x << endl;      // 불가능
    cout << obj2.y << endl;      // 불가능: y는protected로유지된다.
    cout << obj2.z << endl;      // 불가능: z는public에서protected로변경되었다.

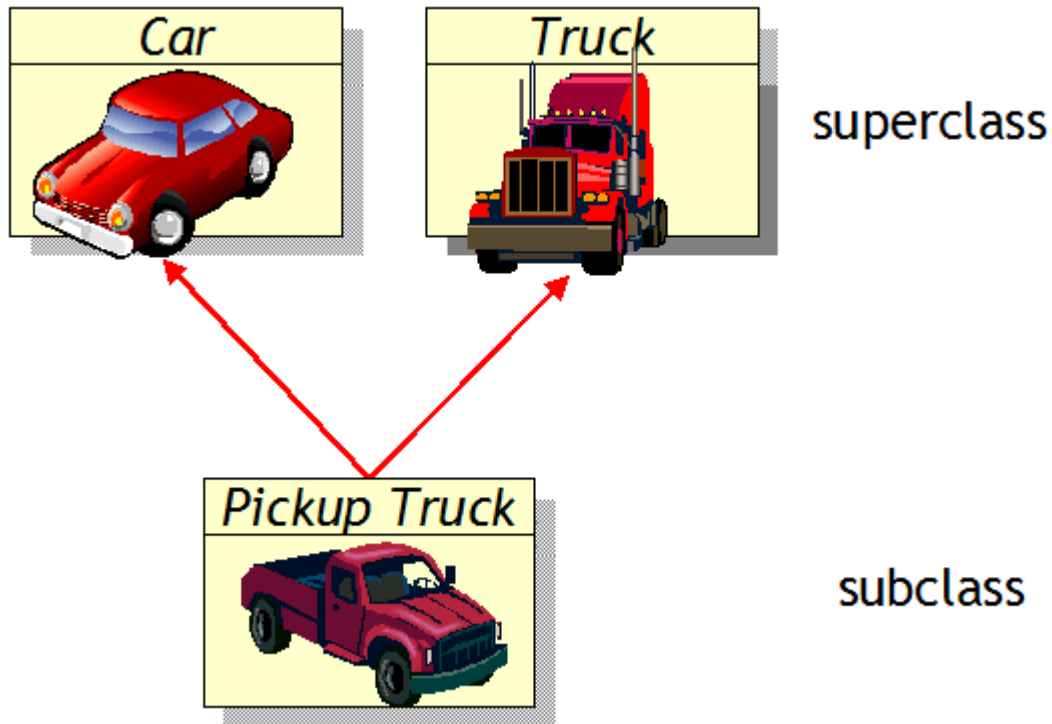
    cout << obj3.x << endl;      // 불가능
    cout << obj3.y << endl;      // 불가능: y는protected에서private로변경되었다.
    cout << obj3.z << endl;      // 불가능: z는public에서private로변경되었다.
    return 0;
}
```



# 다중 상속

- 하나의 자식 클래스가 두 개 이상의 부모 클래스로부터 상속받는 것

```
class Sub : public Sup1, public Sup2
{
    ...// 추가된 멤버
    ...// 재정의된 멤버
}
```





# 예제



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

class PassengerCar {
public:
    int seats; // 정원
    void set_seats(int n){ seats = n; }
};

class Truck {
public:
    int payload; // 적재하중
    void set_payload(int load){ payload = load; }
};

class Pickup : public PassengerCar, public Truck {
public:
    int tow_capability; // 견인능력
    void set_tow(int capa){ tow_capability = capa; }
};
```



# 비교



```
int main()
{
    Pickup my_car;
    my_car.set_seats(4);
    my_car.set_payload(10000);
    my_car.set_tow(30000);
    return 0;
}
```



계속하려면 아무 키나 누르십시오 . . .





# 다중 상속의 문제점



```
class SuperA
{
public:
    int x;
    void sub(){
        cout << "SuperA의 sub()" << endl;
    }
};
class SuperB
{
public:
    int x;
    void sub(){
        cout << "SuperB의 sub()" << endl;
    }
};
```



# 다중 상속의 문제점



```
class Sub : public SuperA, public SuperB
{
};
```

```
int main()
{
    Sub obj;
    obj.x = 10;
    return 0;
}
```

Obj.SuperA::x=10;

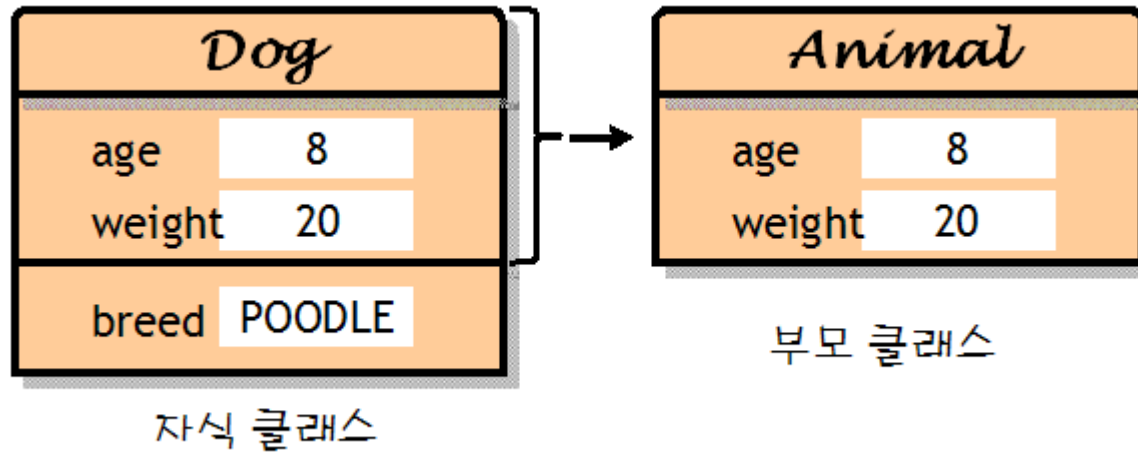
// obj.x는 어떤 부모클래스의 x를 참조하는가?



```
1>.\multi_inheri.cpp(27) : error C2385: 'x' 액세스가 모호합니다.
1>     기본 'SuperA'의 'x'일 수 있습니다.
1>     또는 기본 'SuperB'의 'x'일 수 있습니다.
```



# 예제





# 예제



```
#include <iostream>
using namespace std;
enum BREED { YORKIE, POODLE, BULLDOG };

class Animal
{
protected:
    int age;    // 나이
    int weight; // 몸무게

public:
    // 생성자와 소멸자
    Animal();
    ~Animal();

    // 멤버 함수들
    void speak() const;
    void sleep() const;
    void eat() const;
};
```



# 예제



```
Animal::Animal()
{
    cout << "Animal 생성자\n";
}
Animal::~~Animal()
{
    cout << "Animal 소멸자\n";
}

// 멤버함수들
void Animal::speak() const
{
    cout << "Animal speak()\n";
}
void Animal::sleep() const
{
    cout << "Animal sleep()\n";
}
void Animal::eat() const
{
    cout << "Animal eat()\n";
}
```



# 예제



```
class Dog : public Animal
{
private:
    BREED breed;
public:

    // 생성자와 소멸자
    Dog();
    ~Dog();

    // 멤버 함수들
    void wag();
    void bite();
    void speak() const;
};

Dog::Dog()
{
    cout << "Dog 생성자\n";
}
```



# 예제



```
Dog::~Dog()
{
    cout << "Dog 소멸자\n";
}
// 멤버함수들
void Dog::wag()
{
    cout << "Dog wag()\n";
}
void Dog::bite()
{
    cout << "Dog bite()\n";
}
void Dog::speak() const
{
    cout << "Dog speak()\n";
}
```



# 예제



```
int main()
{
    Dog dog;

    dog.eat();
    dog.sleep();
    dog.speak();
    dog.wag();
    return 0;
}
```

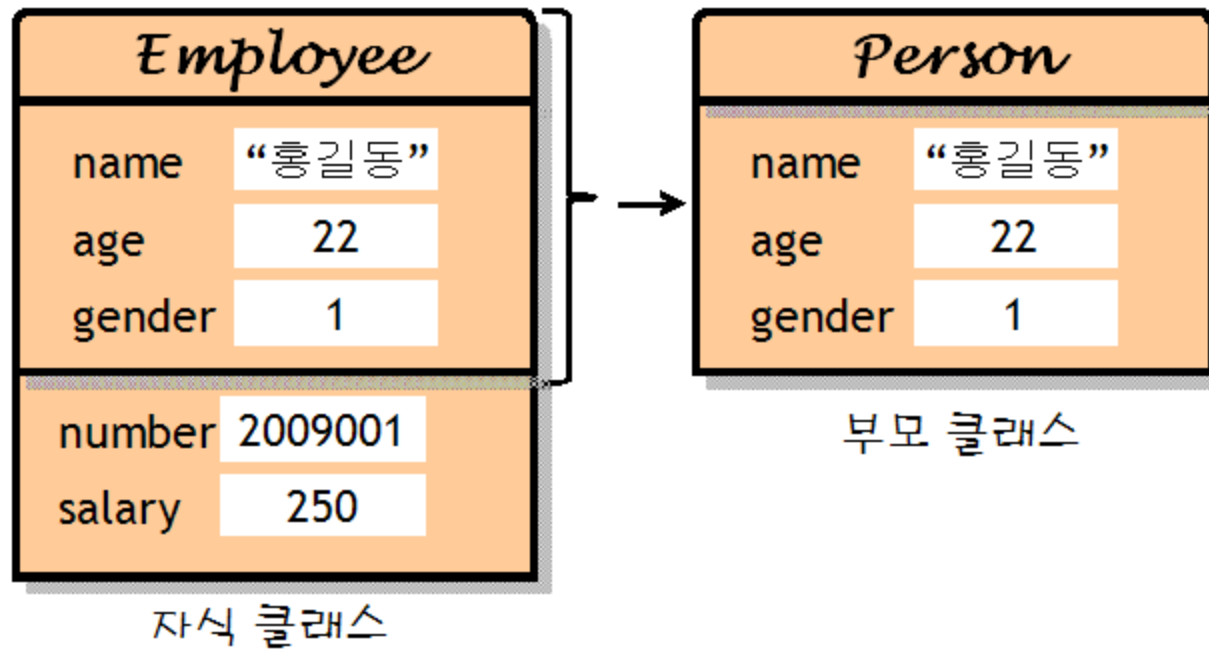


Animal 생성자  
Dog 생성자  
Animal eat()  
Animal sleep()  
Dog speak()  
Dog wag()  
Dog 소멸자  
Animal 소멸자





## 예제 #2





# 예제



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

class Person {
    string name;
    int age;
    bool gender;
public:
    Person(string n="", int a=0, bool g=true): name(n), age(a), gender(g) {
    }

    void setName(string s) { name = s; }
    string getName() const { return name; }
    void setAge (int a) { age = a; }
    int getAge() const { return age; }
    void setGender (bool g) { gender = g; }
    bool getGender() const { return gender; }
};
```



# 예제



```
class Employee : public Person {  
    int number;  
    int salary;  
  
public:  
    Employee(string n="", int a=0, bool g=true, int num=0, int s=0):  
    Person(n, a, g), number(num), salary(s) { }  
    void display() const;  
    void setNumber (int n) { number = n; }  
    int getNumber() const { return number; }  
    void setSalary (int s) { salary = s; }  
    int getSalary() const { return salary; }  
};  
  
void Employee::display() const  
{  
    cout << this->getName() << endl;  
    cout << this->getAge() << endl;  
    cout << this->getGender() << endl;  
    cout << this->getNumber() << endl;  
    cout << this->getSalary() << endl;  
}
```



# 예제



```
int main()
{
    Employee e("김철수", 26, true, 2010001, 2800);
    e.display();
    return 0;
}
```



김철수  
26  
1  
2010001  
2800  
계속하려면 아무 키나 누르십시오 . . .