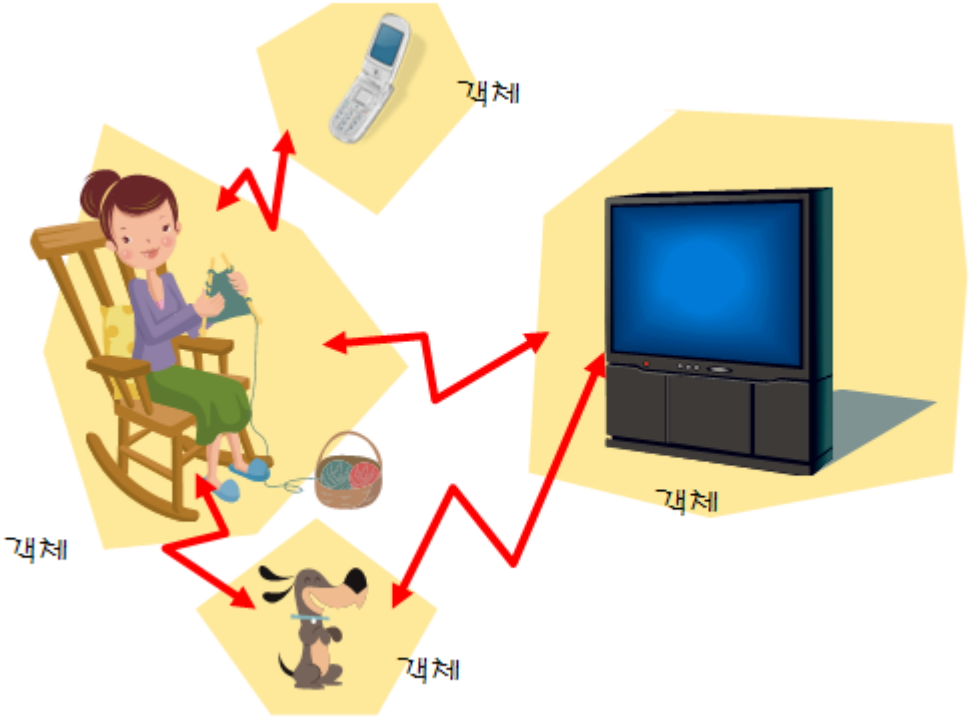
C++ Espresso

제8장 상속

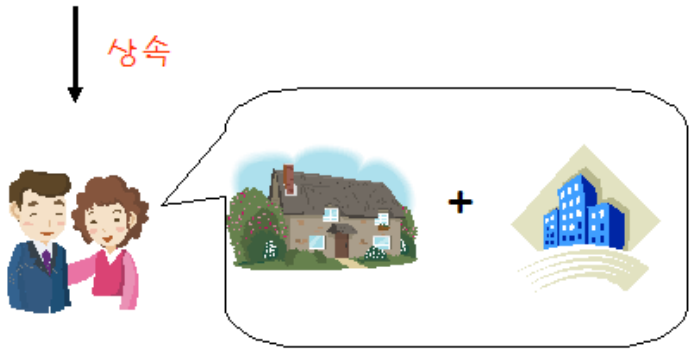


이번 장에서 학습할 내용

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

상속이란?

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



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

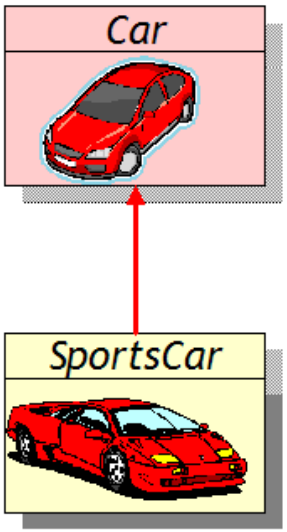
상속의 장점

- 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

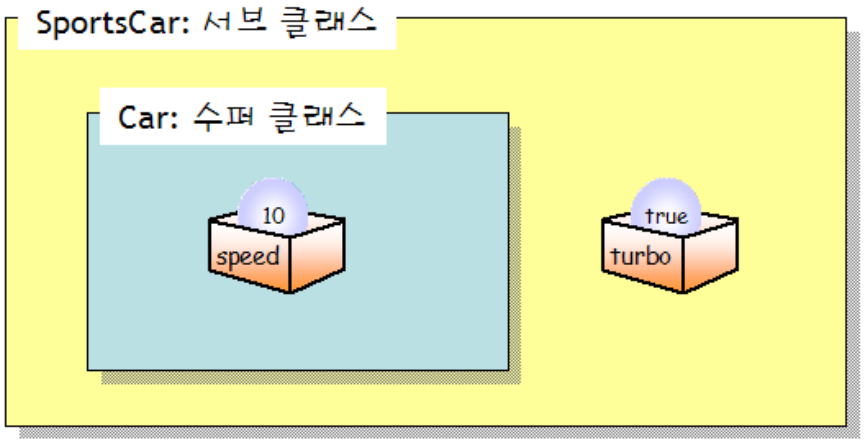
```
class Car
{
        int speed;
}
class SportsCar : public Car
{
        bool turbo;
}
```

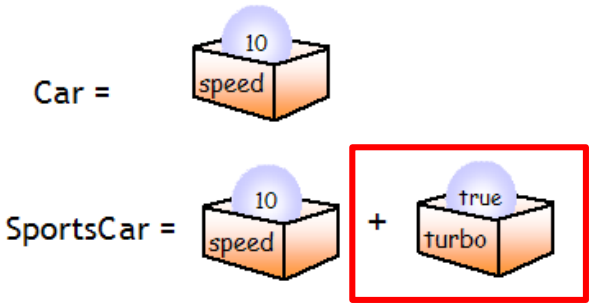


수퍼 클래스(superclass)

서브클래스(subclass)

자식 클래스는 부모 클래스를 포함



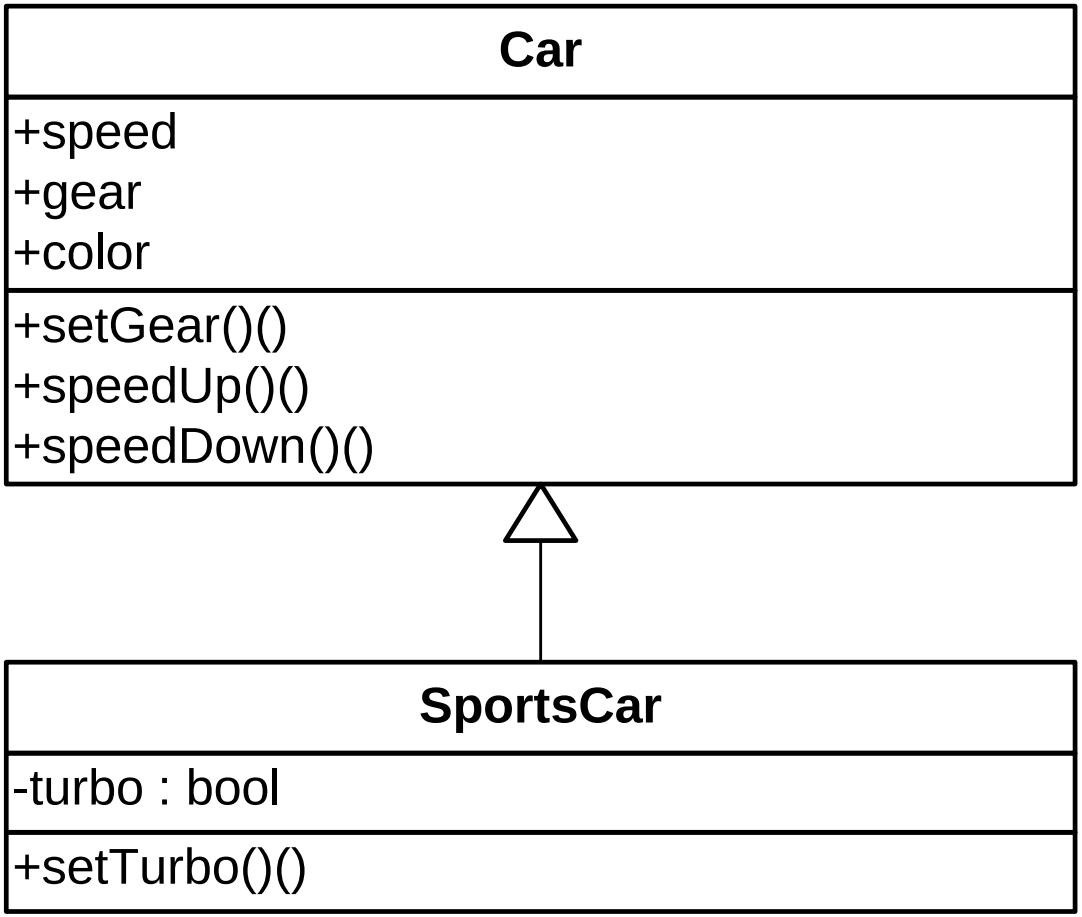


상속의 예

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), Motocycle(오토바이), Bicycle(자전거)
Student(학생)	GraduateStudent(대학원생), UnderGraduate(학부생)
Employee(직원)	Manager(관리자)
Shape(도형)	Rectangle(사각형), Triangle(삼각형), Circle(원)

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

상속의 예제

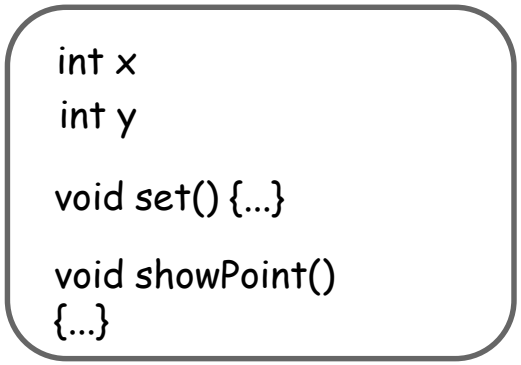


파생 클래스의 객체 구성

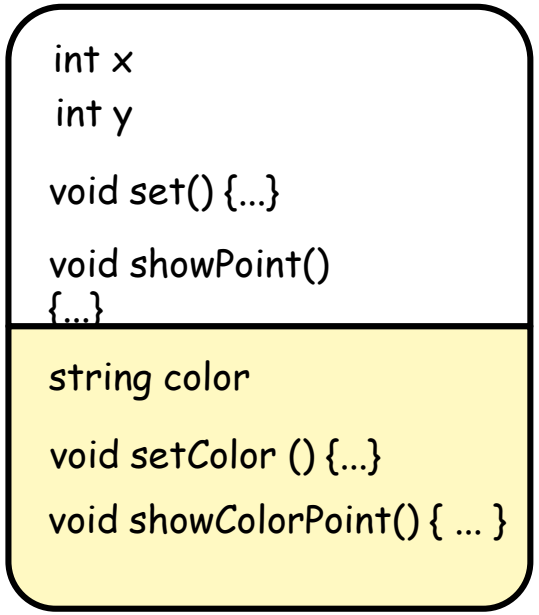
```
class Point {
   int x, y; // 한 점 (x,y) 좌표 값
public:
   void set(int x, int y);
   void showPoint();
};
```

```
class ColorPoint : public Point {    // Point를 상속받음
      string color;                  // 점의 색 표현
public:
      void setColor(string color);
      void showColorPoint();
};
```

Point p;



ColorPoint cp;

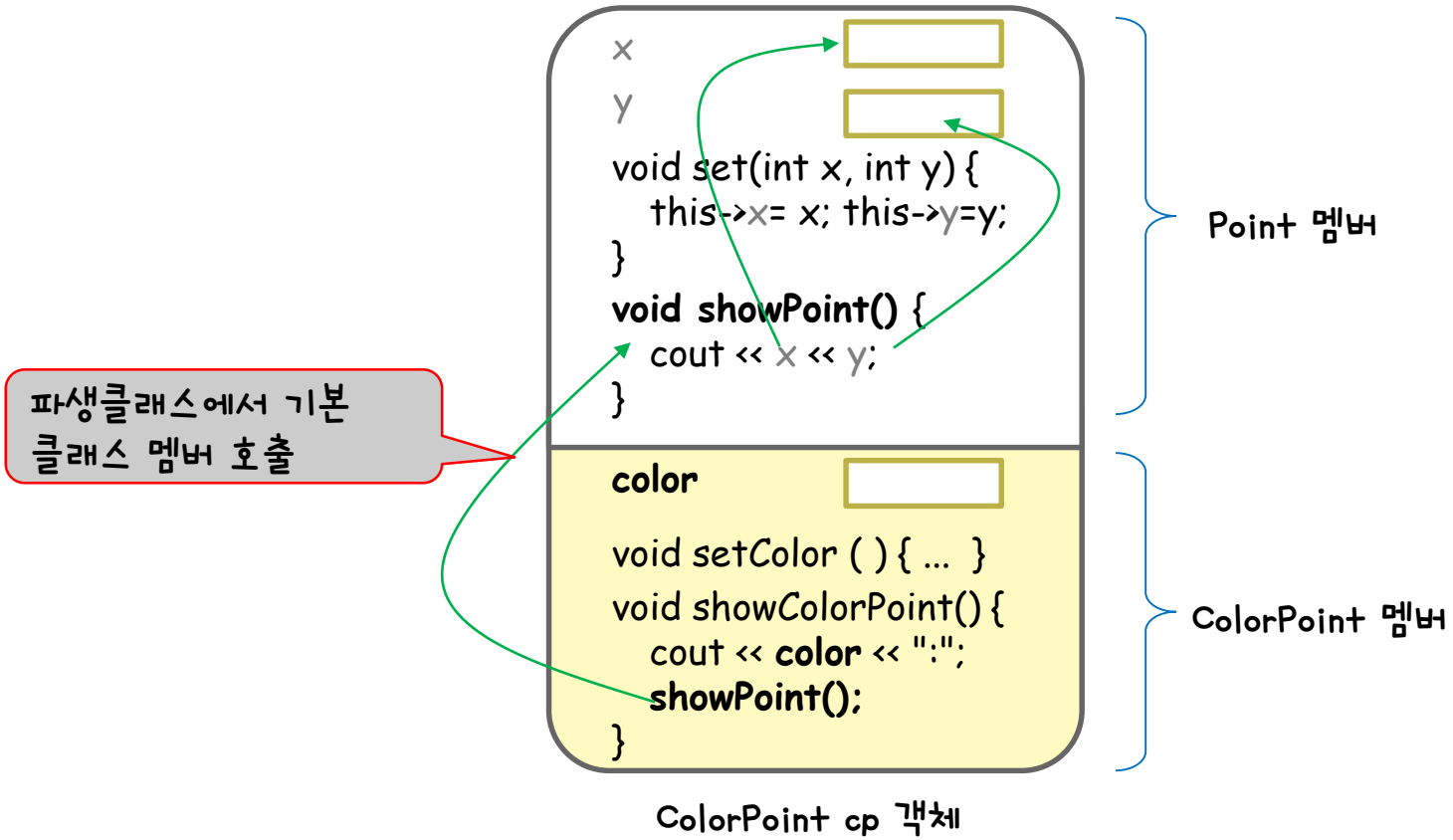


파생 클래스의 객체는 기본 클래스의 멤버 포함

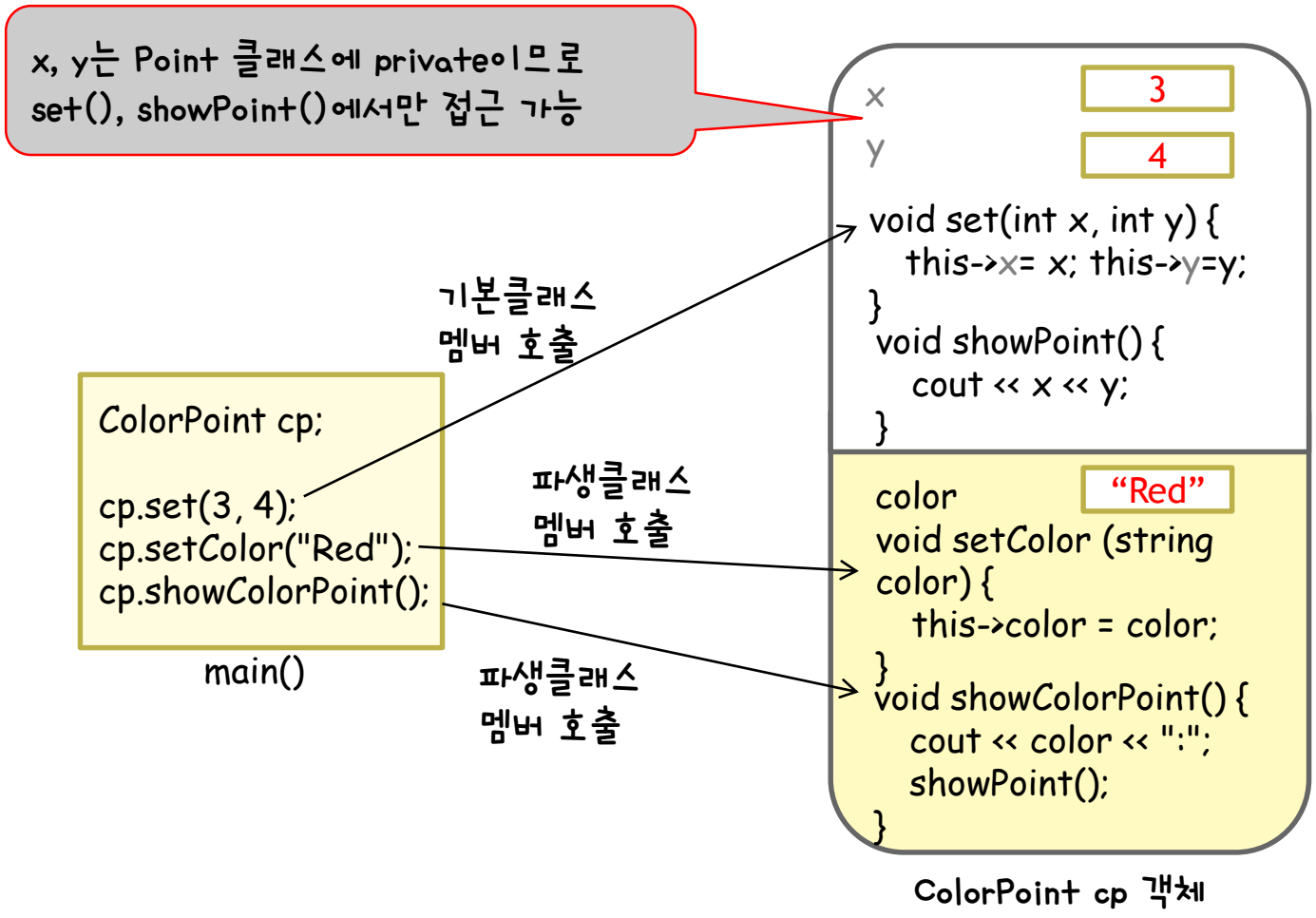
기본클래스 멤버

파생클래스 멤버

파생 클래스에서 기본 클래스 멤버 접근



외부에서 파생 클래스 객체에 대한 접근



#실습
(InheritanceTest1.cpp)

Car 클래스

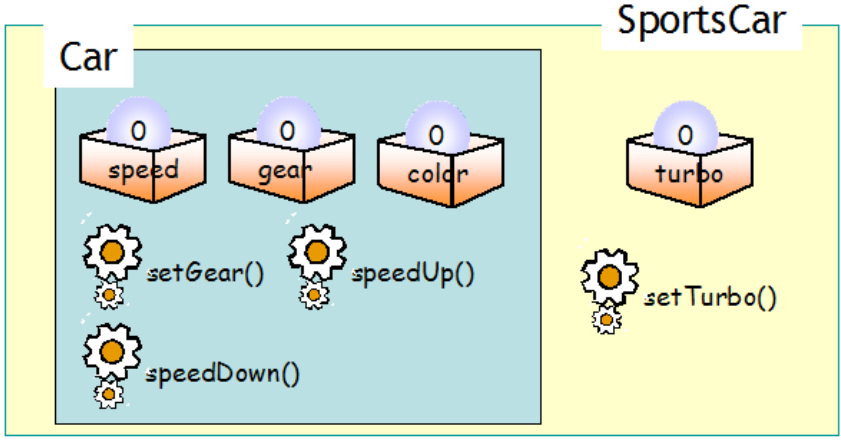
```
#include<iostream>
#include<string>
using namespace std;

class Car{
public:
    int speed;
    int gear;
    string color;
    void setGear(int newGear){
        gear = newGear;
    }
    void setSpeed(int newSpeed){
        speed = newSpeed;
    }
    void speedUp(int increment){
        speed += increment;
        cout << "speed : " << speed << endl;
    }
    void speedDown(int decrement){
        speed -= decrement;
        cout << "speed : " << speed << endl;
    }
};
```

```
class SportsCar :public Car{
    bool turbo;
public:
    void setTurbo(bool newValue){
        turbo = newValue;
    }
};

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

```
C:₩Windows₩system32₩cmd.exe
speed : 200
speed : 170
계속하려면 아무 키나 누르십시오 . . .
```



상속은 왜 필요한가?

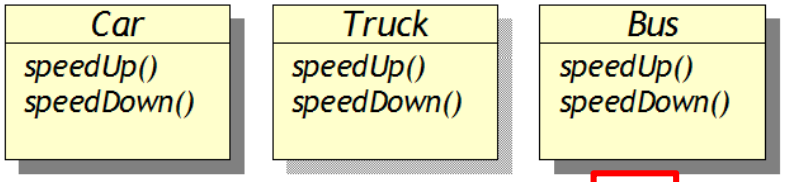


그림 11.6 각 클래스에 코드가 중복된다.

상속

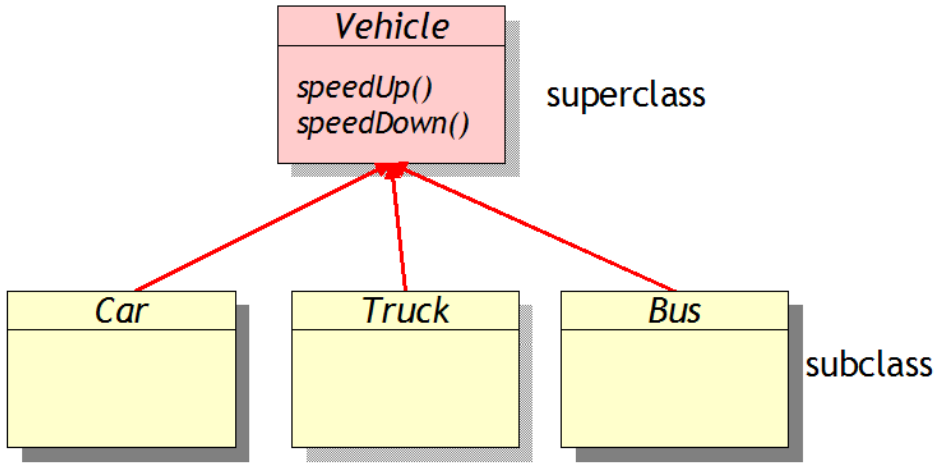
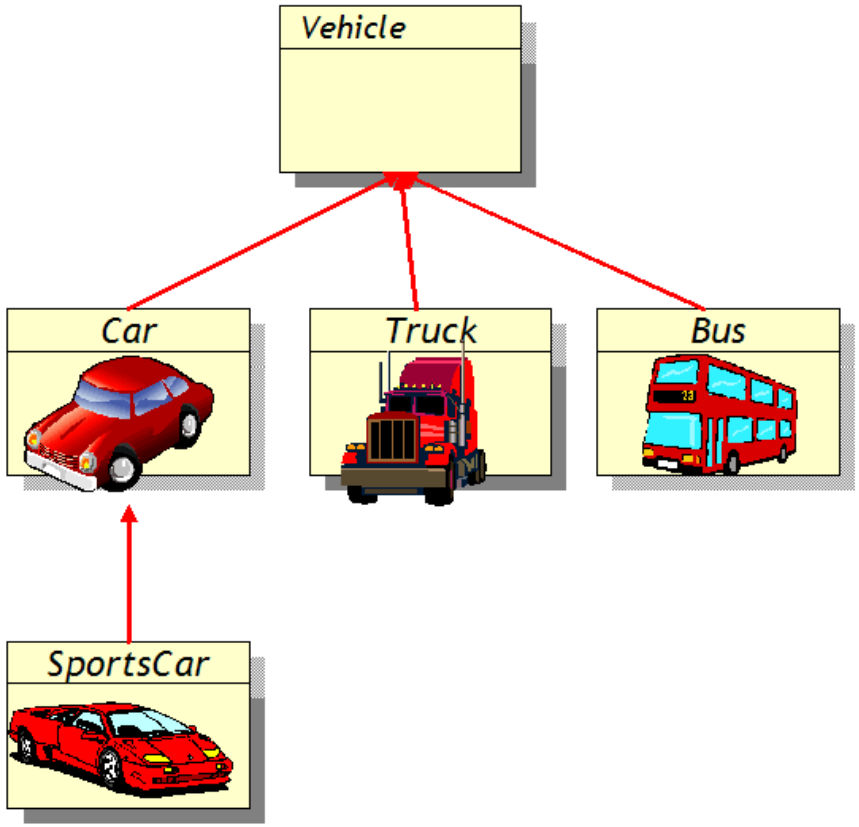


그림 11.7 중복되는 코드는 수퍼 클래스에 모은다.

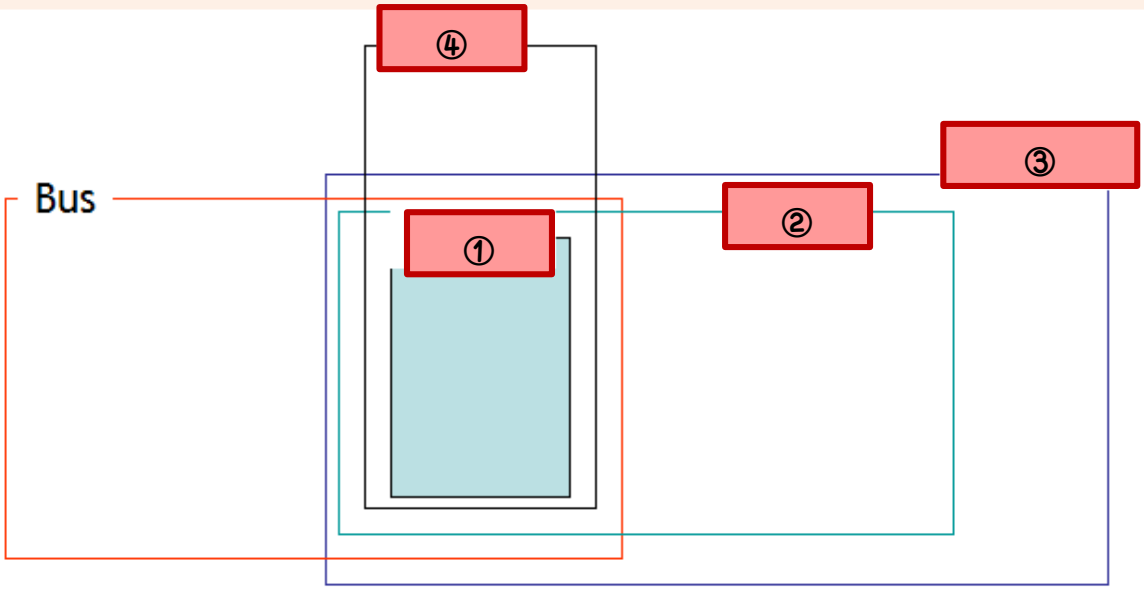
상속 계층도

- 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

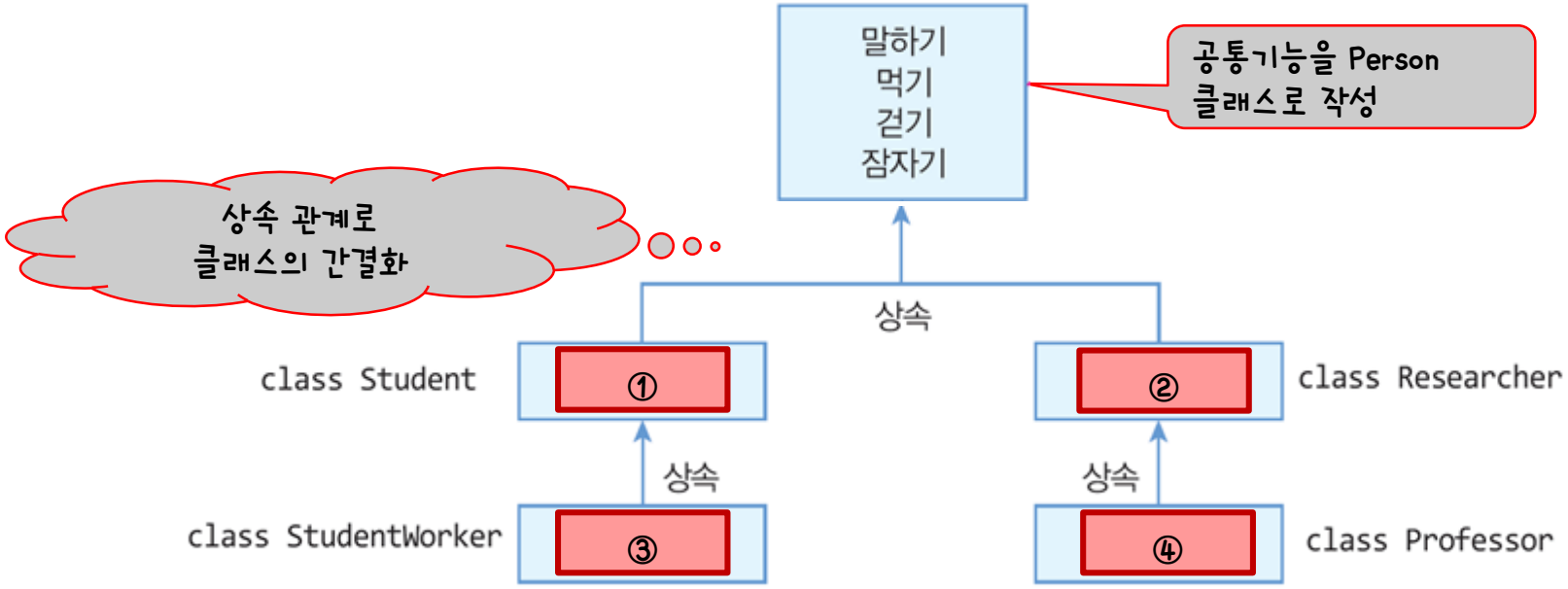
- 상속은 is-a 관계
  - 자동차는 탈것이다. (Car *is a* Vehicle).
  - 사자, 개, 고양이는 동물이다.

- has-a(포함) 관계는 상속으로 모델링을 하면 안 된다.
  - 도서관은 책을 가지고 있다(Library *has a* book).
  - 거실은 소파를 가지고 있다.

중간 점검 문제

기능이 중복된
4 개의 클래스

class Student	class StudentWorker	class Researcher	class Professor
말하기	말하기	말하기	말하기
먹기	먹기	먹기	먹기
걷기	걷기	걷기	걷기
잠자기	잠자기	잠자기	잠자기
공부하기	공부하기	연구하기	연구하기
	일하기		가르치기



#실습
(InheritanceTest2.cpp)

Point 클래스를 상속받는 ColorPoint 클래스 만들기

<Class>

- Point 클래스
  - 멤버변수 : private, int 형식의 x, y
  - 멤버함수 : public 형식
    - set(int, int) : 메인함수에서 x,y값을 받아 객체의 멤버변수를 초기화
    - showPoint() :
```
void showPoint(){
    cout << "(" << x << "," << y << ")" << endl;
}
```

- ColorPoint 클래스 : Point클래스를 상속받는다.
  - 멤버변수 : private, string 객체 color
  - 멤버함수
    - Color의 설정자 : setColor
    - showColorPoint() :
```
void ColorPoint::showColorPoint(){
    cout << color << ":";
    showPoint();
}
```

```
int main()
{
    Point p;
    ColorPoint cp;
    cp.set(3, 4);
    cp.setColor("Red");
    cp.showColorPoint();
}
```

#실습
(InheritanceTest2.cpp)

Point 클래스를 상속받는 ColorPoint 클래스 만들기

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

class Point{
    int x, y;
public:
    void set(int x, int y){
        this->x = x;
        this->y = y;
    }
    void showPoint(){
        cout << "(" << x << "," << y << ")" << endl;
    }
};
```

```
class ColorPoint :public Point{
    string color;
public:
    void setColor(string color){
        this->color = color;
    }
    void showColorPoint();
};
void ColorPoint::showColorPoint(){
    cout << color << ":";
    showPoint();
}

int main()
{
    Point p;
    ColorPoint cp;
    cp.set(3, 4);
    cp.setColor("Red");
    cp.showColorPoint();
}
```

```
C:₩Windows₩system32₩cmd.exe
Red:(3,4)
계속하려면 아무 키나 누르십시오 . . .
```

접근 제어 지정자

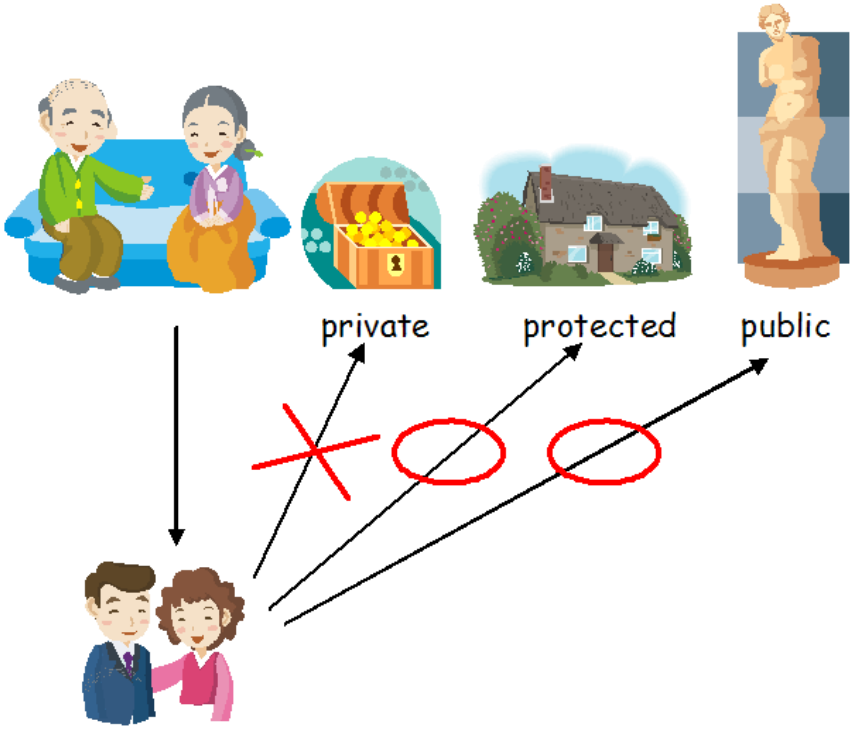
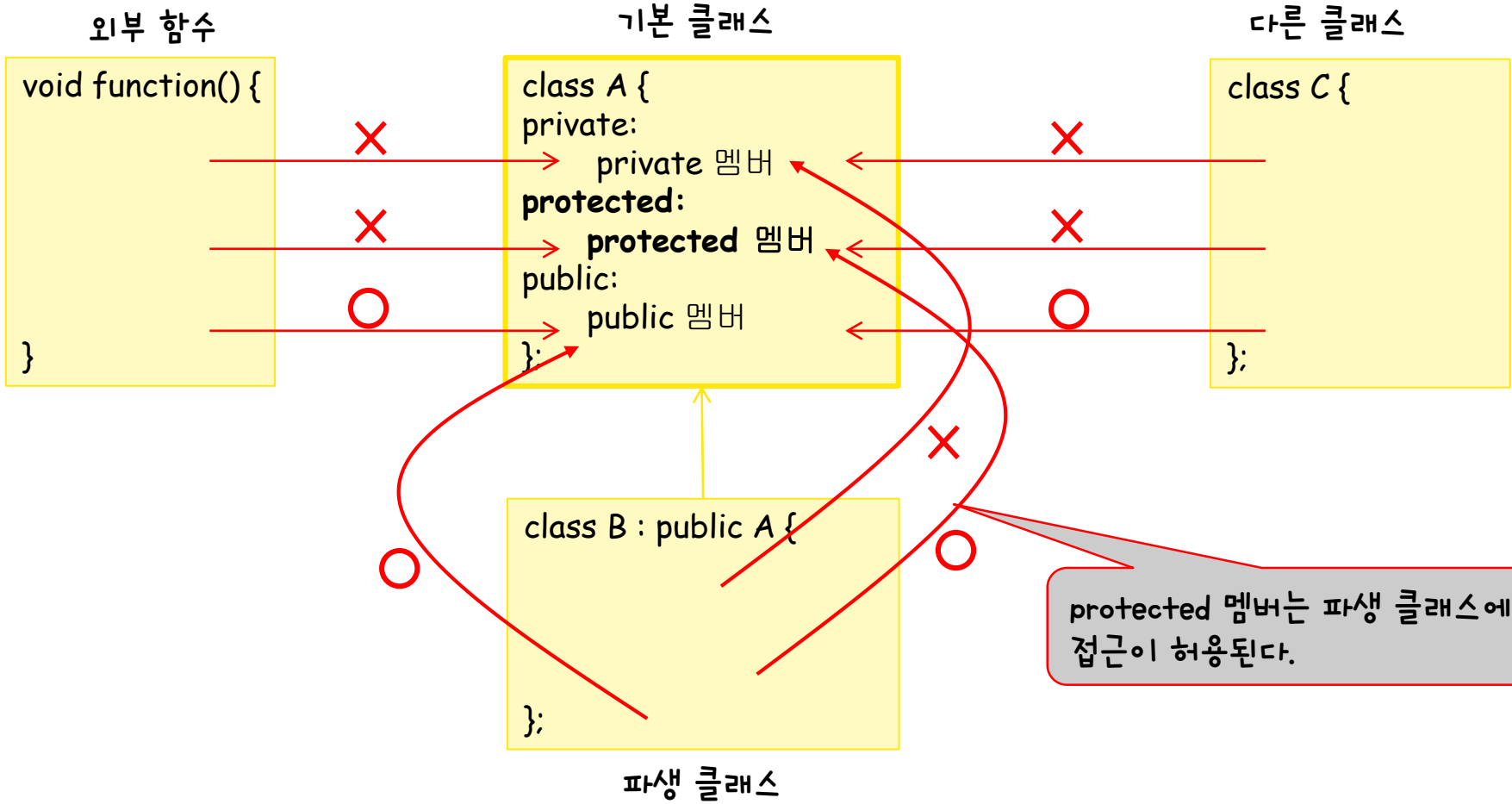


그림 13.9 상속에서의 접근 지정자

protected 접근 지정

- 접근 지정자
  - private 멤버
    - 선언된 클래스 내에서만 접근 가능
    - 파생 클래스에서도 기본 클래스의 private 멤버 직접 접근 불가
  - public 멤버
    - 선언된 클래스나 외부 어떤 클래스, 모든 외부 함수에 접근 허용
    - 파생 클래스에서 기본 클래스의 public 멤버 접근 가능
  - protected 멤버
    - 선언된 클래스에서 접근 가능
    - 파생 클래스에서만 접근 허용
      - 파생 클래스가 아닌 다른 클래스나 외부 함수에서는 protected 멤버를 접근할 수 없다.

멤버의 접근 지정에 따른 접근성



접근 제어 지정자

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

#실습
(InheritanceTest3.cpp)

예제

```
#include <iostream>
#include<string>
using namespace std;
class Employee{
    int rrn;
protected:
    int salary;
public:
    string name;
    void setSalary(int salary);
    int getSalary();
};

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

```
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;
    bonus = b;
}
void Manager::display(){
    cout << "봉급: " << salary << "보너스: " << bonus << endl;
}

int main()
{
    Manager m;
    m.setSalary(2000);
    m.display();
    m.modify(1000, 500);
    m.display();
    return 0;
}
```

```
C:₩Windows₩system32₩cmd.exe
봉급: 2000보너스: 0
봉급: 1000보너스: 500
계속하려면 아무 키나 누르십시오 . . .
```

중간 점검 문제

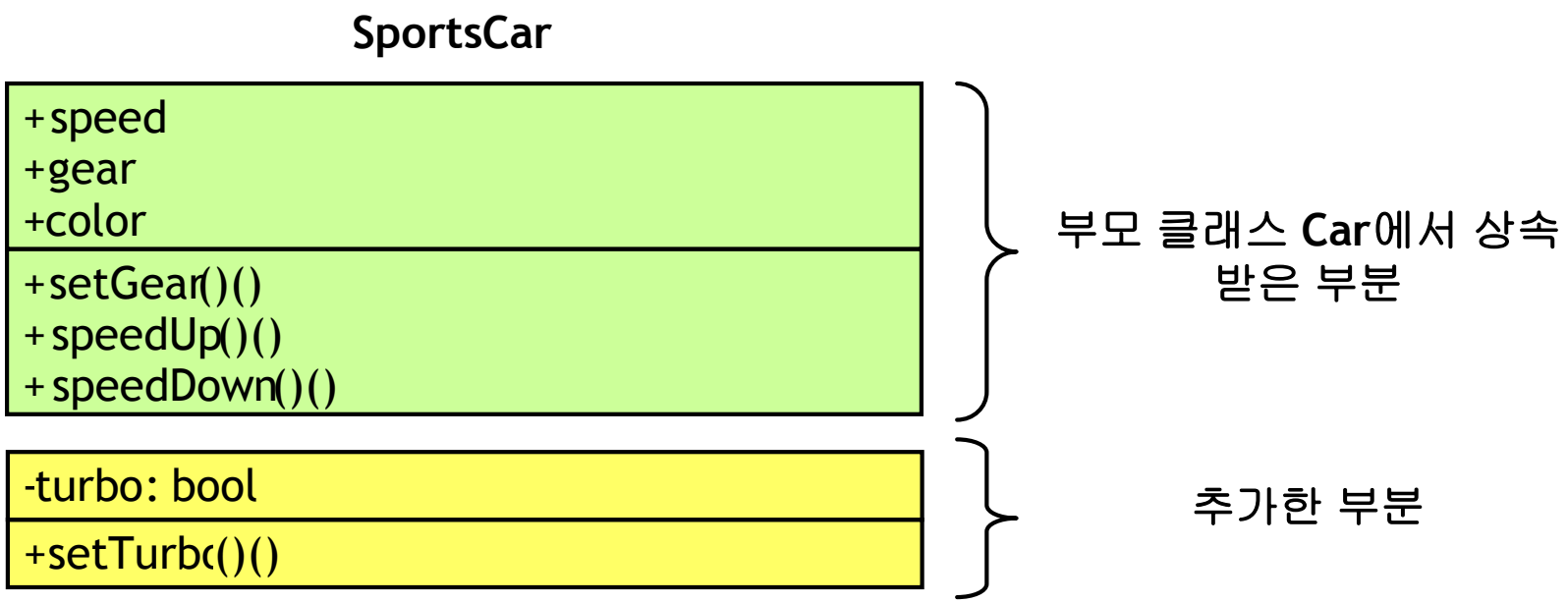
1. protected가 의미하는 바는 무엇인가?

2. 부모 클래스에서 private로 선언된 변수를 자식 클래스에서 사용할 수 있는가?

3. 자식 클래스의 객체 obj를 통하여 부모 클래스에서 protected로 선언된 변수 x를 사용할 수 있는가? 즉 obj.x = 10;과 같은 문장을 작성할 수 있는가?

상속에서의 생성자와 소멸자

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



상속에서의 생성자와 소멸자

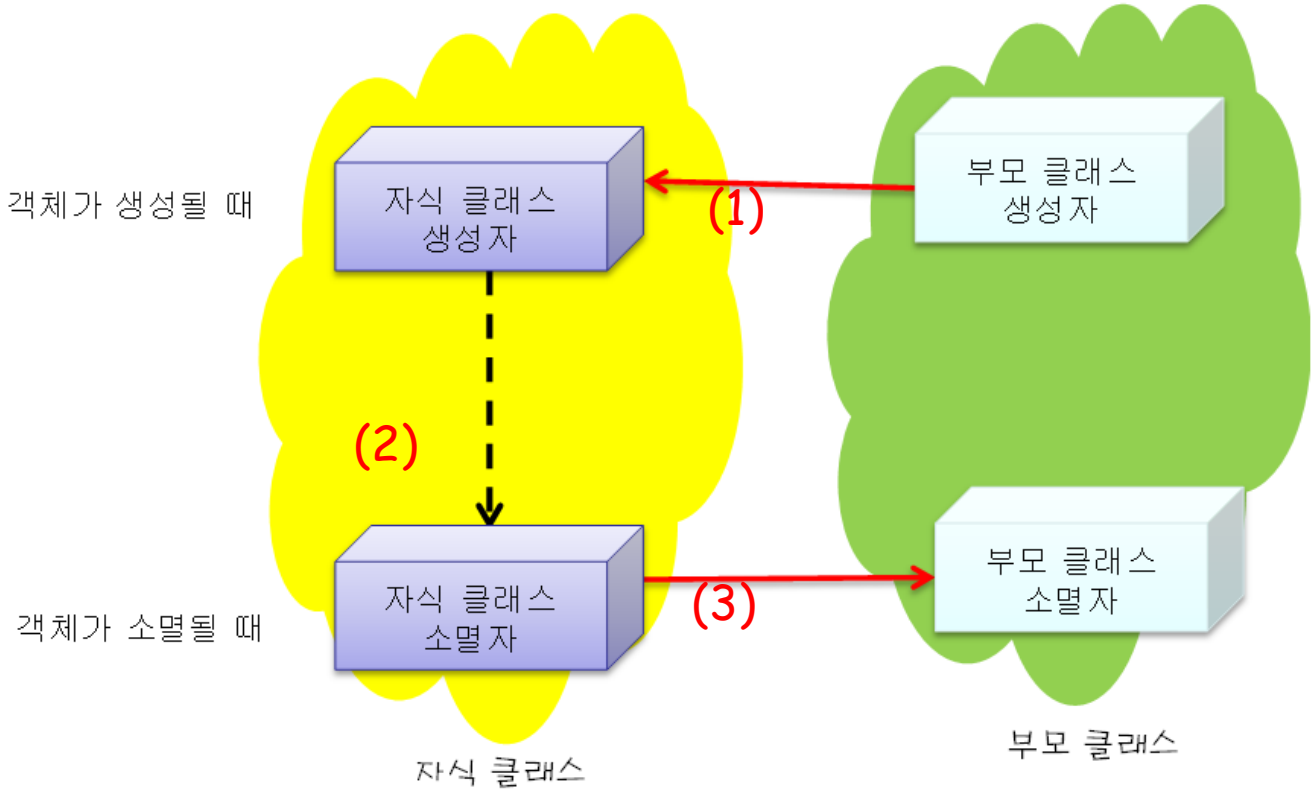
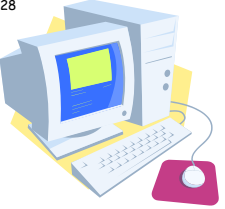


그림 8.10 상속에서 생성자와 소멸자의 호출



생성자 호출 관계 및 실행 순서

```
class A {
public:
   A() { cout << "생성자 A" << endl; }
   ~A() { cout << "소멸자 A" << endl; }
};
```

A() 실행 ❹

A() 호출 ❸

리턴

```
class B : public A {
public:
   B() { cout << "생성자 B" << endl; }
   ~B() { cout << "소멸자 B" << endl; }
};
```

B() 실행 ❺

B() 호출 ❷

리턴

```
int main() {
   C c; // c 생성

   return 0; // c 소멸
}
```

C() 호출 ❶

```
class C : public B {
public:
   C() { cout << "생성자 C" << endl; }
   ~C() { cout << "소멸자 C"<< endl; }
};
```

C() 실행 ❻

```
생성자 A
생성자 B
생성자 C
소멸자 C
소멸자 B
소멸자 A
```

컴파일러는 C() 생성자 실행 코드를 만들때,
생성자 B()를 호출하는 코드 삽입

#실습
(InheritanceTest4.cpp)

예제

```
#include<iostream>
#include<string>
using namespace std;
class Shape{
    int x, y;
public:
    Shape(){
        cout << "Shape 생성자()" << endl;
    }
    ~Shape(){
        cout << "Shape 소멸자()" << endl;
    }
};
```

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

```
C:₩Windows₩system32₩cmd.exe
Shape 생성자()
Rectangle 생성자()
Rectangle 소멸자()
Shape 소멸자()
계속하려면 아무 키나 누르십시오 . . .
```

부모 생성자의 명시적 호출

```
Rectangle(int x=0, int y=0, int w=0, int h=0) : Shape(x,y)
{
    width = w;
    height = h;
}
```

부모 클래스의 생성자 호출

#실습
(InheritanceTest5.cpp)

예제

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

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

```
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;
}

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

```
C:₩Windows₩system32₩cmd.exe
Shape 생성자(xloc,yloc)
Rectangle 생성자(x,y,w,h)
Rectangle 소멸자()
Shape 소멸자()
계속하려면 아무 키나 누르십시오 . . .
```

중간 점검 문제

1. 상속에서 자식 클래스의 생성자와 부모 클래스의 생성자 중에서 함수의 몸체가 먼저 실행되는것은?

2. 상속에서 자식 클래스의 소멸자와 부모 클래스의 소멸자 중에서 함수의 몸체가 먼저 실행되는것은?

Q & A