



상속 관계의 생성자와 소멸자 실행

2

□ 질문 1

- ▣ 파생 클래스의 객체가 생성될 때 파생 클래스의 생성자와 기본 클래스의 생성자가 모두 실행되는가? 아니면 파생 클래스의 생성자만 실행되는가?

■ 답 -

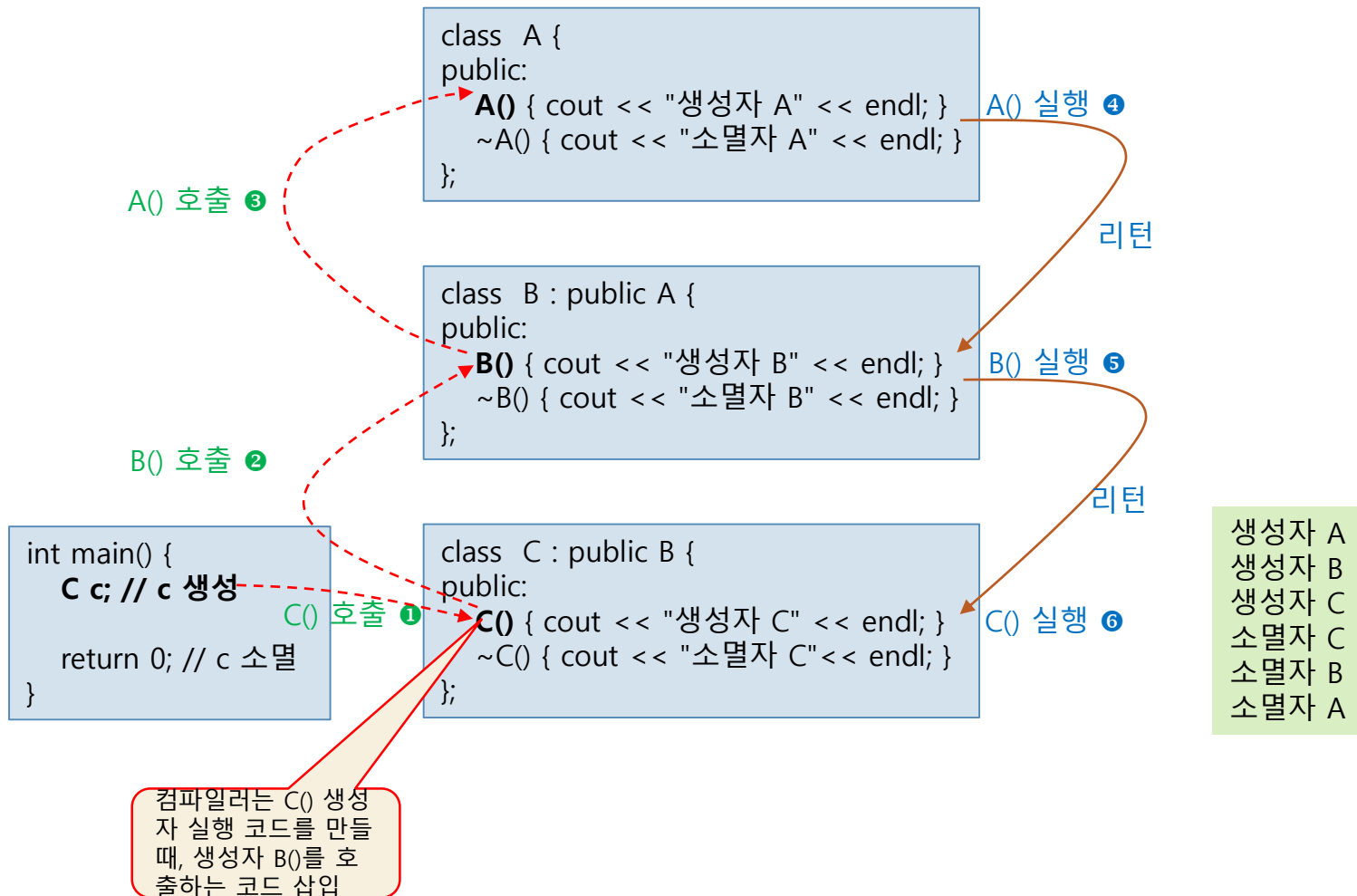
□ 질문 2

- ▣ 파생 클래스의 생성자와 기본 클래스의 생성자 중 어떤 생성자가 먼저 실행되는가?

■ 답 -

생성자 호출 관계 및 실행 순서

3



소멸자의 실행 순서

4

- 파생 클래스의 객체가 소멸될 때
 - ▣ 파생 클래스의 소멸자가 먼저 실행되고
 - ▣ 기본 클래스의 소멸자가 나중에 실행

컴파일러에 의해 묵시적으로 기본 클래스의 생성자를 선택하는 경우

5

파생 클래스의 생성자에서 기본 클래스의 기본 생성자 호출

컴파일러는 묵시적으로 기본 클래스의 기본 생성자를 호출하도록 컴파일함

```
class A {  
public:  
    ▶ A() { cout << "생성자 A" << endl; }  
    A(int x) {  
        cout << " 매개변수생성자 A" << x << endl;  
    }  
};
```

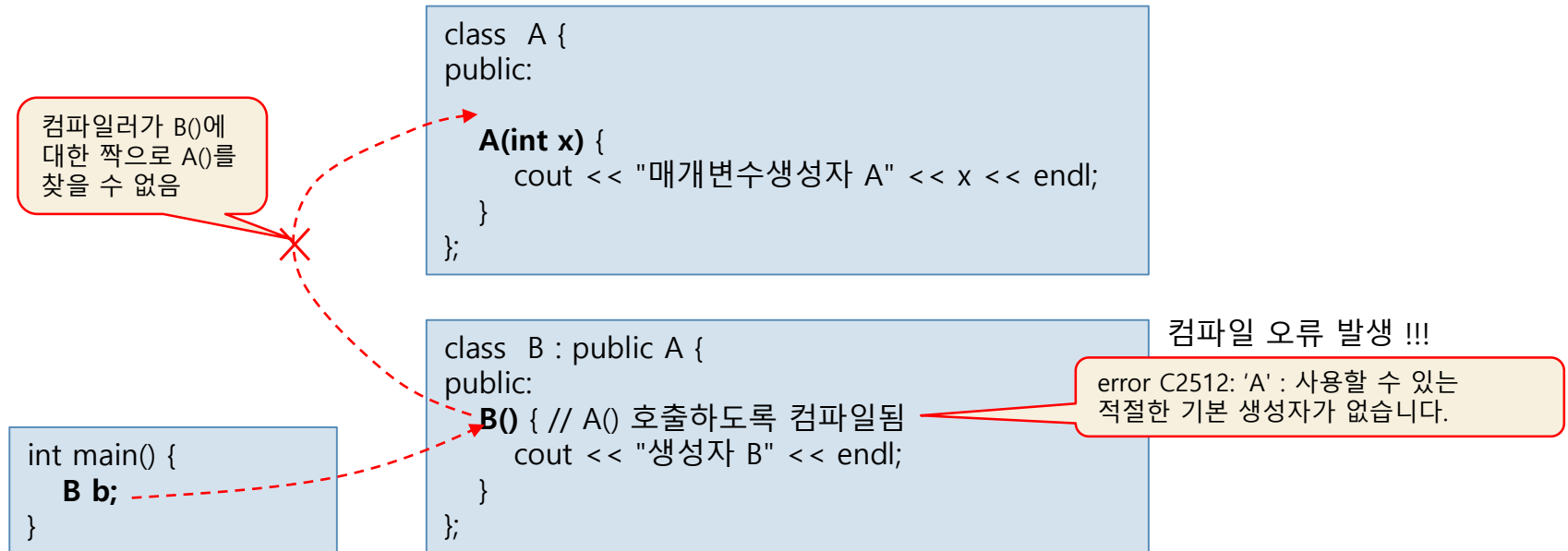
```
class B : public A {  
public:  
    ▶ B() { // A() 호출하도록 컴파일됨  
        cout << "생성자 B" << endl;  
    }  
};
```

```
int main() {  
    B b;  
}
```

생성자 A
생성자 B

기본 클래스에 기본 생성자가 없는 경우

6



매개 변수를 가진 파생 클래스의 생성자는 묵시적으로 기본 클래스의 기본 생성자 선택

7

파생 클래스의 매개 변수를 가진 생성자가 기본 클래스의 기본 생성자 호출

컴파일러는 묵시적으로 기본 클래스의 기본 생성자를 호출하도록 컴파일함

```
class A {  
public:  
    A() { cout << "생성자 A" << endl; }  
    A(int x) {  
        cout << "매개변수생성자 A" << x << endl;  
    }  
};
```

```
class B : public A {  
public:  
    B() { // A() 호출하도록 컴파일됨  
        cout << "생성자 B" << endl;  
    }  
    B(int x) { // A() 호출하도록 컴파일됨  
        cout << "매개변수생성자 B" << x << endl;  
    }  
};
```

```
int main() {  
    B b(5);  
}
```

생성자 A
매개변수생성자 B5

파생 클래스의 생성자에서 명시적으로 기본 클래스의 생성자 선택

8

파생 클래스의 생성자가 명시적으로 기본 클래스의 생성자를 선택 호출함

```
class A {  
public:  
    A() { cout << "생성자 A" << endl; }  
    A(int x) {  
        cout << "매개변수생성자 A" << x << endl;  
    }  
};
```

A(8) 호출

```
class B : public A {  
public:  
    B() { // A() 호출하도록 컴파일됨  
        cout << "생성자 B" << endl;  
    }  
    B(int x) : A(x+3) {  
        cout << "매개변수생성자 B" << x << endl;  
    }  
};
```

B(5) 호출

```
int main() {  
    B b(5);  
}
```

매개변수생성자 A8
매개변수생성자 B5

컴파일러의 기본 생성자 호출 코드 삽입

9

```
class B {  
    B() : A() {  
        cout << "생성자 B" << endl;  
    }  
  
    B(int x) : A() {  
        cout << "매개변수생성자 B" << x << endl;  
    }  
};
```

컴파일러가 묵시적으로
삽입한 코드

컴파일러가 묵시적으로
삽입한 코드

예제 8-3 TV, WideTV, SmartTV 생성자 매개 변수 전달

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

class TV {

};

class WideTV : public TV { // TV를 상속받는 WideTV

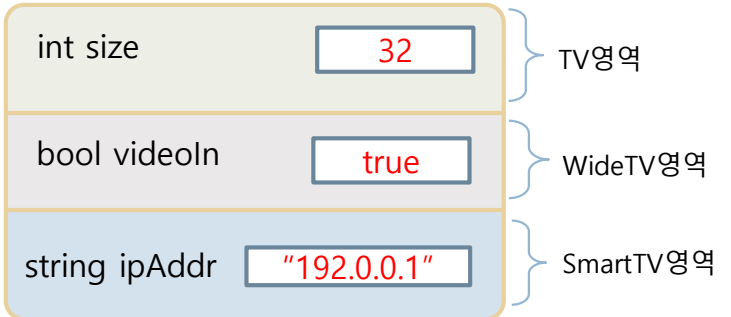
};

class SmartTV : public WideTV { // WideTV를 상속받는 SmartTV
```

```
int main() {
    // 32 인치 크기에 "192.0.0.1"의 인터넷 주소를 가지는 스마트 TV 객체 생성
    SmartTV htv("192.0.0.1", 32);
    cout << "size=" << htv.getSize() << endl;
    cout << "videoIn=" << boolalpha << htv.getVideoIn() << endl;
    cout << "IP=" << htv.getIpAddr() << endl;
}
```

boolalpha는 불린 값을 true, false로 출력되게 하는 조작자

size=32
videoIn=true
IP=192.0.0.1



htv

상속 지정

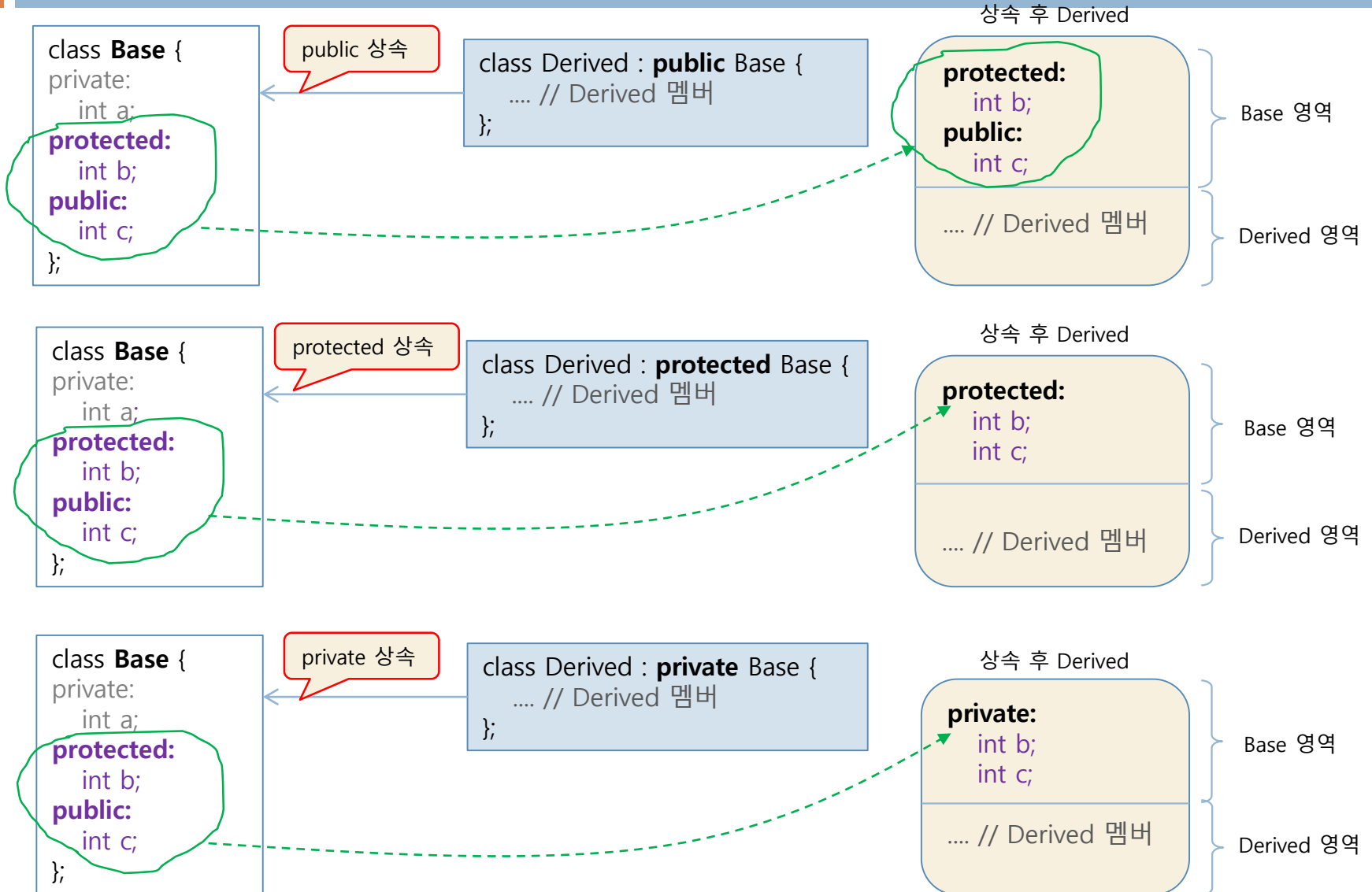
11

▣ 상속 지정

- 상속 선언 시 `public`, `private`, `protected`의 3가지 중 하나 지정
- 기본 클래스의 멤버의 접근 속성을 어떻게 계승할지 지정
 - `public` – 기본 클래스의 `protected`, `public` 멤버 속성을 그대로 계승
 - `private` – 기본 클래스의 `protected`, `public` 멤버를 `private`으로 계승
 - `protected` – 기본 클래스의 `protected`, `public` 멤버를 `protected`로 계승

상속 시 접근 지정에 따른 멤버의 접근 지정 속성 변화

12



예제 8-4 private 상속 사례

13

다음에서 컴파일 오류가 발생하는 부분을 찾아라.

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

class Base {
    int a;
protected:
    void setA(int a) { this->a = a; }
public:
    void showA() { cout << a; }
};

class Derived : private Base {
    int b;
protected:
    void setB(int b) { this->b = b; }
public:
    void showB() { cout << b; }
};
```

```
int main() {
    Derived x;
    x.a = 5;           // ①
    x.setA(10);        // ②
    x.showA();         // ③
    x.b = 10;          // ④
    x.setB(10);        // ⑤
    x.showB();         // ⑥
}
```

컴파일 오류

예제 8-5 protected 상속 사례

14

다음에서 컴파일 오류가 발생하는 부분을 찾아라.

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

class Base {
    int a;
protected:
    void setA(int a) { this->a = a; }
public:
    void showA() { cout << a; }
};

class Derived : protected Base {
    int b;
protected:
    void setB(int b) { this->b = b; }
public:
    void showB() { cout << b; }
};
```

```
int main() {
    Derived x;
    x.a = 5;           // ①
    x.setA(10);        // ②
    x.showA();          // ③
    x.b = 10;          // ④
    x.setB(10);         // ⑤
    x.showB();          // ⑥
}
```

컴파일 오류

예제 8-6 상속이 중첩될 때 접근 지정 사례

15

다음에서 컴파일 오류가 발생하는 부분을 찾아라.

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

class Base {
    int a;
protected:
    void setA(int a) { this->a = a; }
public:
    void showA() { cout << a; }
};

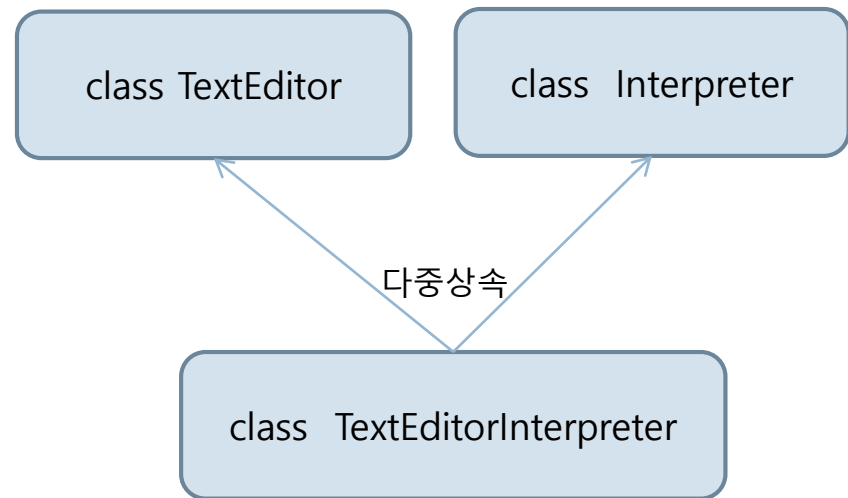
class Derived : private Base {
    int b;
protected:
    void setB(int b) { this->b = b; }
public:
    void showB() {
        setA(5);           // ①
        showA();           // ②
        cout << b;
    }
};
```

```
class GrandDerived : private Derived {
    int c;
protected:
    void setAB(int x) {
        setA(x);           // ③
        showA();           // ④
        setB(x);           // ⑤
    }
};
```

컴파일 오류

기기의 컨버전스와 C++의 다중 상속

16



다중 상속 선언 및 멤버 호출

17

```
class MP3 {  
public:  
    void play();  
    void stop();  
};
```

```
class MobilePhone {  
public:  
    bool sendCall();  
    bool receiveCall();  
    bool sendSMS();  
    bool receiveSMS();  
};
```

상속받고자 하는 기본
클래스를 나열한다.

다중 상속 선언

```
class MusicPhone : public MP3, public MobilePhone { // 다중 상속 선언  
public:  
    void dial();  
};
```

다중 상속 활용

```
void MusicPhone::dial() {  
    play(); // mp3 음악을 연주시키고  
    sendCall(); // 전화를 건다.  
}
```

MP3::play() 호출

MobilePhone::sendCall() 호출

다중 상속 활용

```
int main() {  
    MusicPhone hanPhone;  
    hanPhone.play(); // MP3의 멤버 play() 호출  
    hanPhone.sendSMS(); // MobilePhone의 멤버 sendSMS() 호출  
}
```

예제 8-7 Adder와 Subtractor를 다중 상속 받는 Calculator 클래스 작성

18

Adder와 Subtractor를 다중 상속받는 Calculator를 작성하라.

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

class Adder {
protected:
    int add(int a, int b) { return a+b; }
};

class Subtractor {
protected:
    int minus(int a, int b) { return a-b; }
};
```

```
// 다중 상속
class Calculator {

}
```

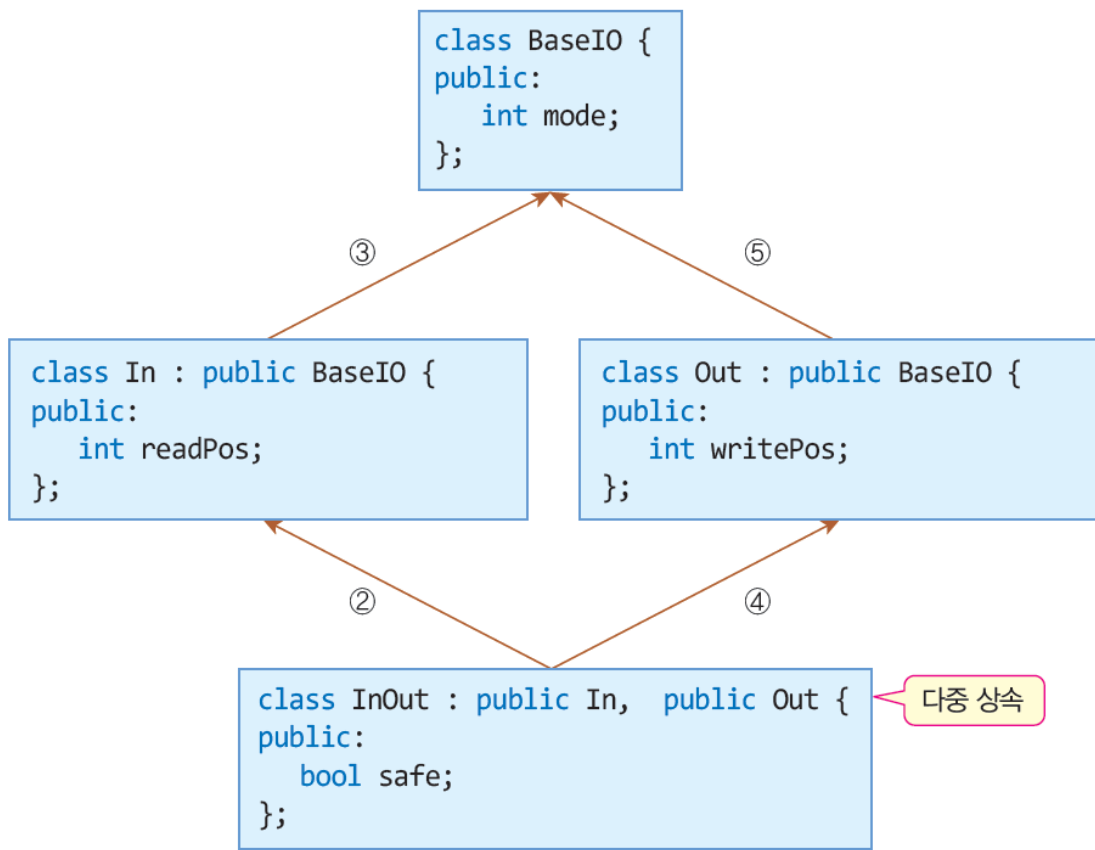
```
int main() {
    Calculator handCalculator;
    cout << "2 + 4 = "
         << handCalculator.calc('+', 2, 4) << endl;
    cout << "100 - 8 = "
         << handCalculator.calc('-', 100, 8) << endl;
}
```

```
2 + 4 = 6
100 - 8 = 92
```

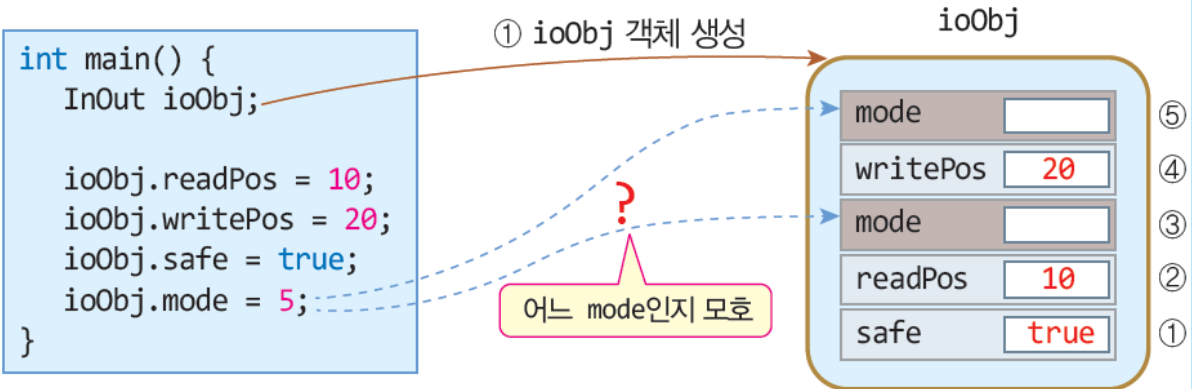
다중 상속의 문제점

- 기본 클래스 멤버의 중복 상속

- Base의 멤버가 이중으로 객체에 삽입되는 문제점.
- 동일한 x를 접근하는 프로그램이 서로 다른 x에 접근하는 결과를 낳게되어 잘못된 실행 오류가 발생된다.



(a) 클래스 상속 관계



(b) ioObj 객체 생성 과정 및 객체 내부

가상 상속

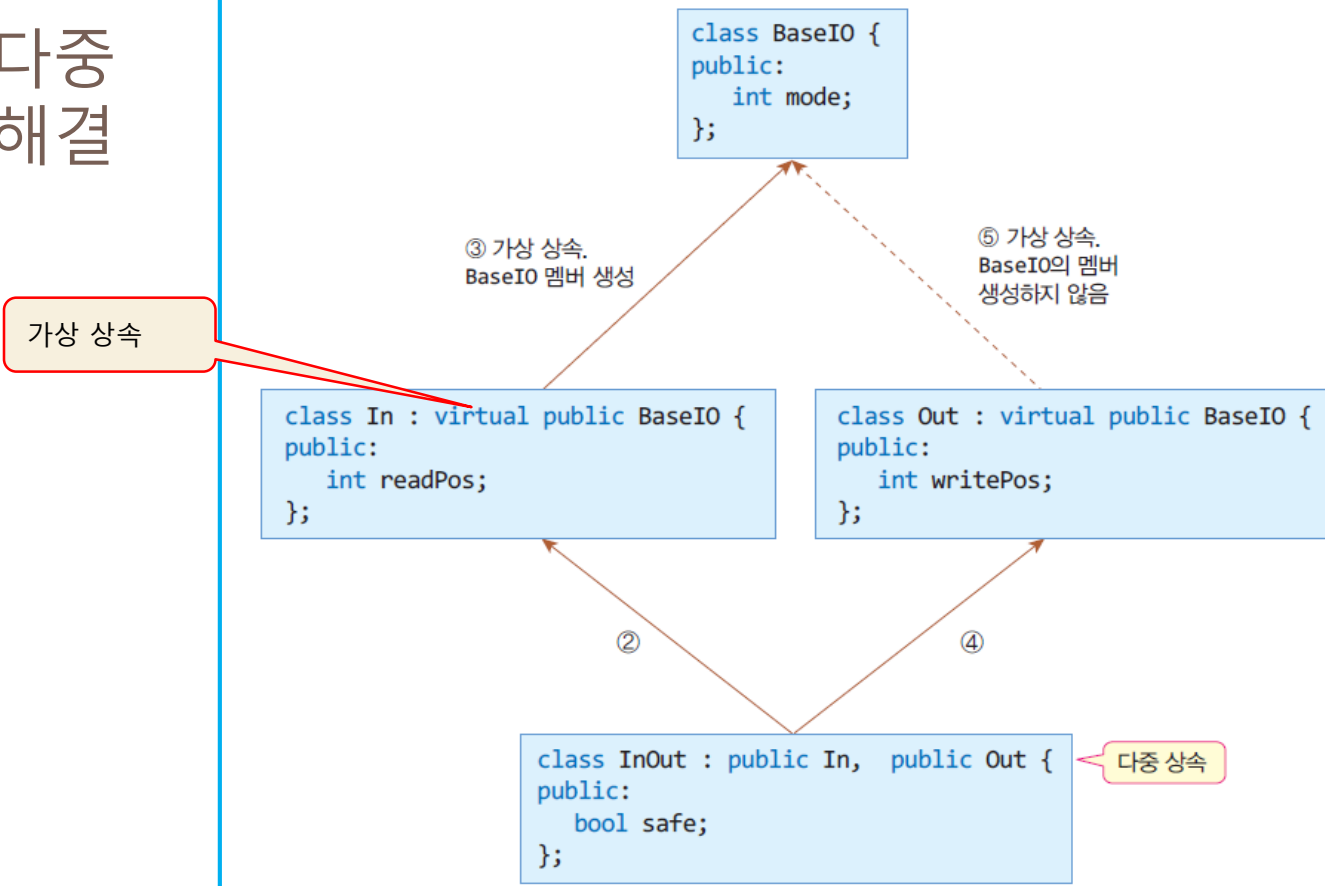
20

- 다중 상속으로 인한 기본 클래스 멤버의 중복 상속 해결
- 가상 상속
 - ▣ 파생 클래스의 선언문에서 기본 클래스 앞에 **virtual**로 선언
 - ▣ 파생 클래스의 객체가 생성될 때 기본 클래스의 멤버는 오직 한 번만 생성
 - 기본 클래스의 멤버가 중복하여 생성되는 것을 방지

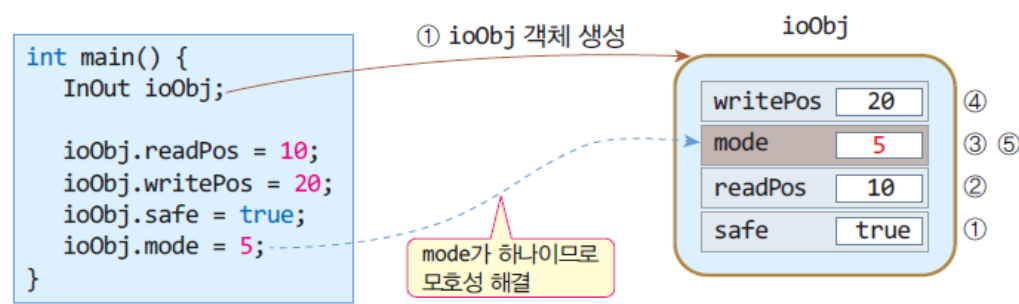
```
class In : virtual public BaseIO { // In 클래스는 BaseIO 클래스를 가상 상속함
...
};
```

```
class Out : virtual public BaseIO { // Out 클래스는 BaseIO 클래스를 가상 상속함
...
};
```

가상 상속으로 다중 상속의 모호성 해결



(a) 기본 클래스를 가상 상속 받는 클래스 상속 관계



(b) 가상 기본 클래스를 가진 경우, ioObj 객체 생성 과정 및 객체 내부