

10주차 수업 정리노트

note #5

2023. 11. 6 (월)
16조 이수영, 원준서

목차

❖Pair Programming(3p~18p)

- 상속관계인 클래스 계층화
- 예제 8-1 ColorPoint 클래스
- 업캐스팅&다운캐스팅 논의
- 예제 8-2 오류 발생 원인 논의
- 예제 8-3 답&해석
- 예제 8-5 오류 발생 원인 해석
- 예제 8-7 다중 상속 구현

❖수업 내용 정리(19p~27p)

- C++에서의 상속
- 상속의 목적&장점
- protected 접근 지정
- 상속의 생성자, 소멸자
- 소멸자의 실행 순서
- 상속의 종류-상속 지정
- 다중 상속
- 가상 상속

Pair Programming

[1차시] ppt p.7 상속 관계인 클래스 간결화 답 안 보고 계층 구조로 표현해보기

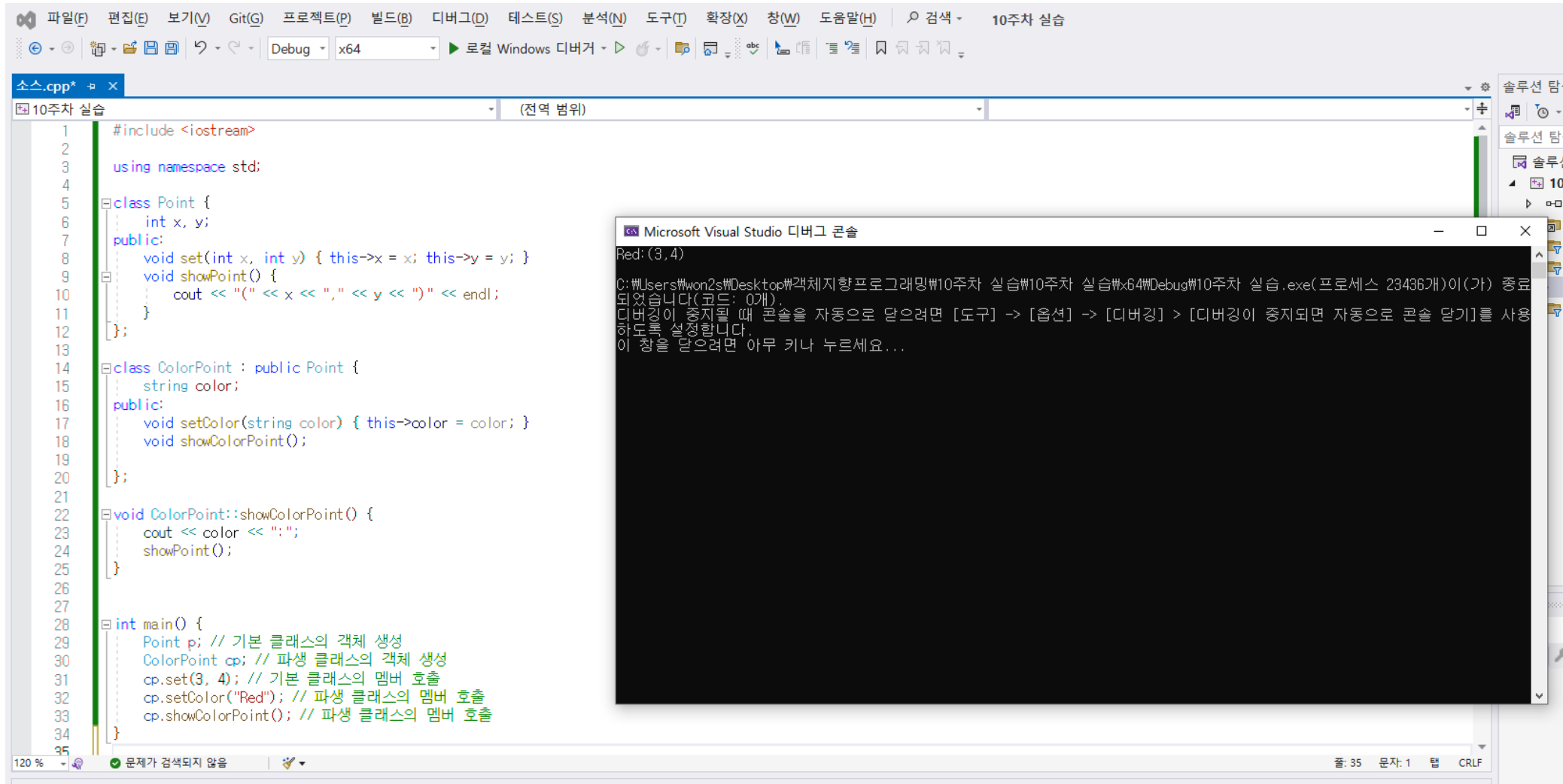
- 원준서
- class Student, class StudentWorker, class Researcher, class Professor
- 4개의 클래스는 공통적으로 말하기, 먹기, 걷기, 잠자기 기능이 포함되어 있습니다. 때문에 말하기, 먹기, 걷기, 잠자기가 포함된 공통 기능을 Person클래스로 작성하고, 공부하기, 일하기, 연구하기, 가르치기를 가지고 있는 클래스들을 분류한 후, Person클래스에게 상속을 받은 이후 Student->StudentWorker 순을 상속을 받고, Researcher->Professor 순으로 상속을 받습니다.

[1차시] ppt p.7 상속 관계인 클래스 간결화 답 안 보고 계층 구조로 표현해보기

- 이수영
- class Person { 말하기, 먹기, 걷기, 잠자기 }
- class Student : public Person { 공부하기 }
- class StudentWorker : public Student { 일하기 }
- class Researcher : public Person { 연구하기 }
- class Professor : public Researcher { 가르치기 }

[1차시] ppt p.9 예제 8-1

ColorPoint 클래스 안 보고 구현(원준서)



The image shows a Visual Studio IDE with a C++ project named '10주차 실습'. The main code file, '소스.cpp', contains the following code:

```
1 #include <iostream>
2
3 using namespace std;
4
5 class Point {
6     int x, y;
7 public:
8     void set(int x, int y) { this->x = x; this->y = y; }
9     void showPoint() {
10         cout << "(" << x << ", " << y << ")" << endl;
11     }
12 };
13
14 class ColorPoint : public Point {
15     string color;
16 public:
17     void setColor(string color) { this->color = color; }
18     void showColorPoint();
19 };
20
21
22 void ColorPoint::showColorPoint() {
23     cout << color << " ";
24     showPoint();
25 }
26
27
28 int main() {
29     Point p; // 기본 클래스의 객체 생성
30     ColorPoint cp; // 파생 클래스의 객체 생성
31     cp.set(3, 4); // 기본 클래스의 멤버 호출
32     cp.setColor("Red"); // 파생 클래스의 멤버 호출
33     cp.showColorPoint(); // 파생 클래스의 멤버 호출
34 }
35
```

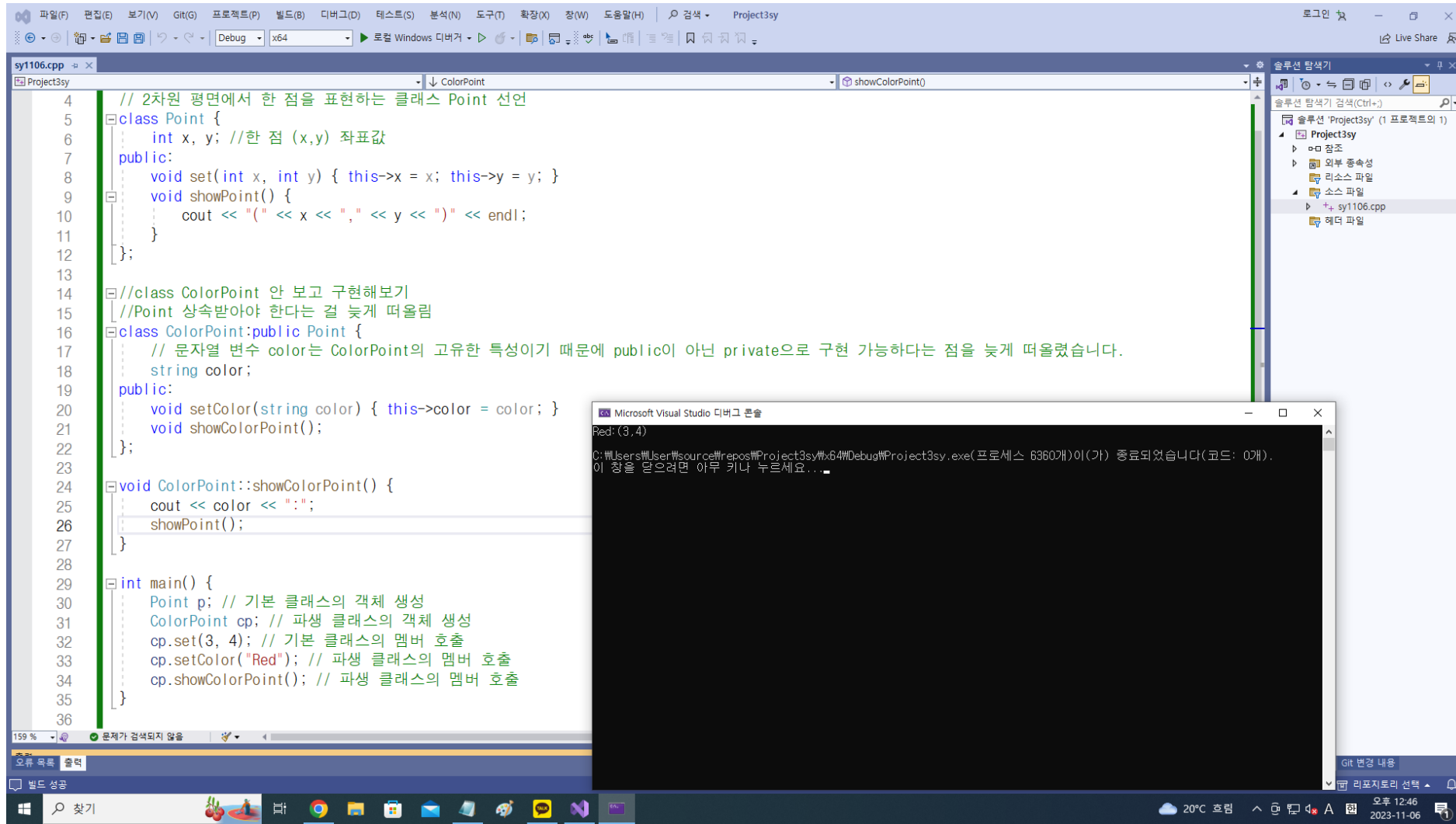
The debug console window, titled 'Microsoft Visual Studio 디버그 콘솔', shows the output of the program:

```
Red: (3,4)
C:\Users\won2s\Desktop\백체지향프로그래밍\10주차 실습\10주차 실습\Debug\10주차 실습.exe (프로세스 23436개)이(가) 종료되었습니다(코드: 0x0).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

The status bar at the bottom indicates '120 %' zoom, '문제가 검색되지 않음' (no problems found), and '줄: 35 문자: 1 탭 CRLF' (line 35, character 1, tab CRLF).

[1차시] ppt p.9 예제 8-1

ColorPoint 클래스 안 보고 구현(이수영)



[2차시] 업캐스팅, 다운캐스팅 설명(원준서)

(1) 업캐스팅

- 파생 클래스 포인터가 기본 클래스 포인터에 치환되는 것입니다.
- ex) 사람을 동물로 봄.

[2차시] 업캐스팅, 다운캐스팅 설명(원준서)

(2) 다운캐스팅

- 기본 클래스의 포인터가 파생 클래스의 포인터에 치환되는 것입니다.

[2차시] 업캐스팅, 다운캐스팅 설명(이수영)

(1) 업캐스팅

- 업캐스팅의 정의는 파생 클래스 포인터가 기본 클래스 포인터에 치환되는 것입니다. 치환된다는 것의 의미가 어려웠습니다. 저만의 언어로 바꿔보면, 파생 클래스 포인터가 이제 기본 클래스 포인터가 가리키는 곳을 똑같이 가리키게 됐다. 로 이해했습니다.
- 따라서 파생 클래스는 기본 클래스 멤버에 접근할 수 있다는 것, 기본 클래스가 파생 클래스 멤버에 접근하려고 하면 컴파일 오류가 난다는 것으로 이해했습니다. 파생 클래스가 기본 클래스에서 private 으로 선언된 멤버에 접근할 수 없었는데, 업캐스팅을 통해 기본 클래스의 private 멤버에도 접근할 수 있게 됩니다.

[2차시] 업캐스팅, 다운캐스팅 설명(이수영)

(2) 다운캐스팅

- 반면 다운캐스팅의 정의는 기본 클래스 포인터가 파생 클래스 포인터에 치환되는 것입니다. 업캐스팅에서 기본 클래스가 파생 클래스의 private 멤버에 접근할 수 없었는데, 이제 가능합니다. 기본 클래스 포인터가 파생 클래스 포인터가 가리키는 곳을 똑같이 가리키게 됐기 때문입니다.
- 따라서 기본 클래스는 파생 클래스 멤버에 접근할 수 있다는 것, 대신 강제 타입 변환을 반드시 해야 기본 클래스가 파생 클래스 멤버에 컴파일 오류 없이 접근할 수 있습니다.

[2차시] 업캐스팅, 다운캐스팅 설명(이수영)

(3) 한 줄 정리

- 한 마디로 정리하면,
- 업캐스팅은 파생 클래스가 기본 클래스의 (private) 멤버에 접근할 수 있게 된 것이고,
- 다운캐스팅은 기본 클래스가 파생 클래스 멤버에 접근해도 컴파일 오류 없이 접근할 수 있게 해주는 것입니다. 대신 이 때는 강제 타입변환이 필수적입니다.

8-2 1~6번 오류가 발생한 원인을 해석해봐라.

>예제8-2에서 발생했던 문제, 해결 과정 정리

- 3, 4는 기본 클래스가 '자식 클래스만 접근 가능한' protected로 지정된 변수 x, y에 접근하려고 해서 오류가 났나?
- 5, 6은 파생 클래스가 protected로 지정된 변수 x, y에 접근하려 하는데 왜 오류가 날까? 오류가 나면 안 되는데. 라고 생각했습니다.
- 그런데 이후에 예제 8-5에 대한 교수님의 해설을 듣고 예제 8-2에서 저희가 생각 못 했던 부분을 알게 됐고, 고민이 해결됐습니다.
- 예제 8-2는 기본 클래스 객체 p가, 파생 클래스 객체 cp가 protected 변수 x, y에 접근해서 오류가 발생하느냐 안 하느냐가 아니다. main()이라는 (자식 클래스가 아닌) 외부 함수에서 protected 변수에 접근하려고 하니까 2, 3, 4 그리고 5, 6에서 컴파일 오류가 난 것입니다. protected 접근 지정자로 선언된 변수는 외부에서 접근이 불가능하기 때문입니다.

예제8-3 답&해석

- SmartTV(string ipAddr, int size)에 따라 htv는 ipAddr가 192.0.0.1, size는 32가 됩니다. 이 값은 SmartTV가 WideTV를 상속받기 때문에, WideTV의 size에도 32가 적용되고 WideTV의 ipAddr도 192.0.0.1이 적용되며, WideTV 클래스가 TV 클래스를 상속받기 때문에, TV 클래스의 size도 32가 됩니다.
- 명시적 생성자 SmartTV(...)는 WideTV의 size라는 매개변수를 가진 생성자를 호출하라고 했기 때문에, 부모 클래스들의 변수를 읽어올 수 있는 것입니다. 또한 매개변수가 없는 TV()의 size인 20이 아니라 매개변수 int size가 있는 TV(int size)가 호출될 것입니다.

예제8-5 답&왜 컴파일오류 나는지 해석

2, 3은 8-5 해설을 듣고 난 후에 작성했습니다.

- 1: a는 기본 클래스(Base)의 private 멤버에 접근했는데, private 접근 지정자로 선언된 변수는 Base 클래스 외부에서 접근할 수 없어서 컴파일 오류가 발생합니다.
- 2, 3은 예제 8-2를 풀지 못해서, 오류가 왜 발생하는지 몰랐습니다. 그런데 교수님 해설을 듣고 2, 3을 해석할 수 있었습니다.
- 2: setA()는 자식 클래스인 Derived()에서 접근할 수 있는데 왜 오류가 날까?
- 3: showA()는 public()인데 왜 오류가 난다고 할까? public이 아니라 protected 상속이므로 showA()는 protected 변수로 봅니다. 그런데 자식 클래스가 아닌 '외부 함수 main()'은 protected 멤버에 접근할 수 없는데, 그 멤버 함수를 사용하려고 하니 오류가 난 것입니다.

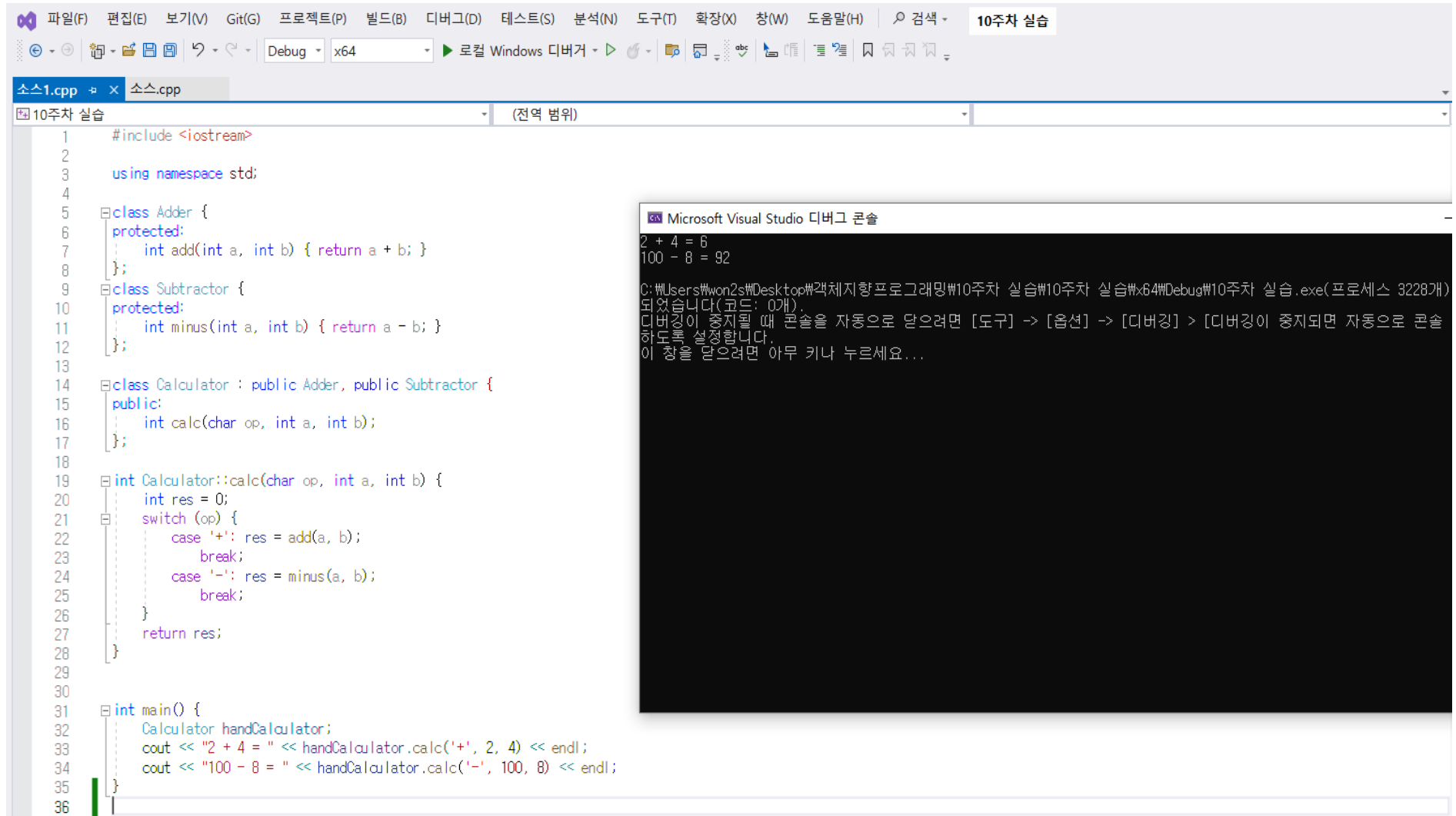
예제8-5 답&왜 컴파일오류 나는지 해석

2, 3은 8-5 해설을 듣고 난 후에 작성했습니다.

- 2: setA()는 protected로 지정된 멤버함수인데, 외부함수 main()은 protected로 지정된 멤버에 접근도 사용도 못 합니다.
- 4: b는 Derived 클래스에 선언된 private 멤버입니다. private으로 지정된 멤버는 그 클래스(Derived) 내에서만 효력이 있고, 접근할 수 있습니다. 외부(main)에서 private에 접근했기 때문에 오류가 납니다.
- 5: setB()는 protected로 지정된 멤버이므로, 외부에서 접근할 수 없습니다.
- 6: showB()는 Derived 클래스 자기자신이고, public으로 지정돼 있어서 외부에서도 접근 가능하므로 오류가 나지 않습니다.

[3차시] ppt p.35 예제 8-7

다중상속 구현 부분 안 보고 구현(원준서)



The image shows a screenshot of the Microsoft Visual Studio IDE. The main window displays a C++ source file named '소스1.cpp'. The code implements a calculator using multiple inheritance. It defines two base classes, 'Adder' and 'Subtractor', which are inherited by a 'Calculator' class. The 'Calculator' class has a public method 'calc' that takes an operator and two integers, and returns the result. The 'main' function creates a 'Calculator' object and tests the 'calc' method with '+' and '-' operators.

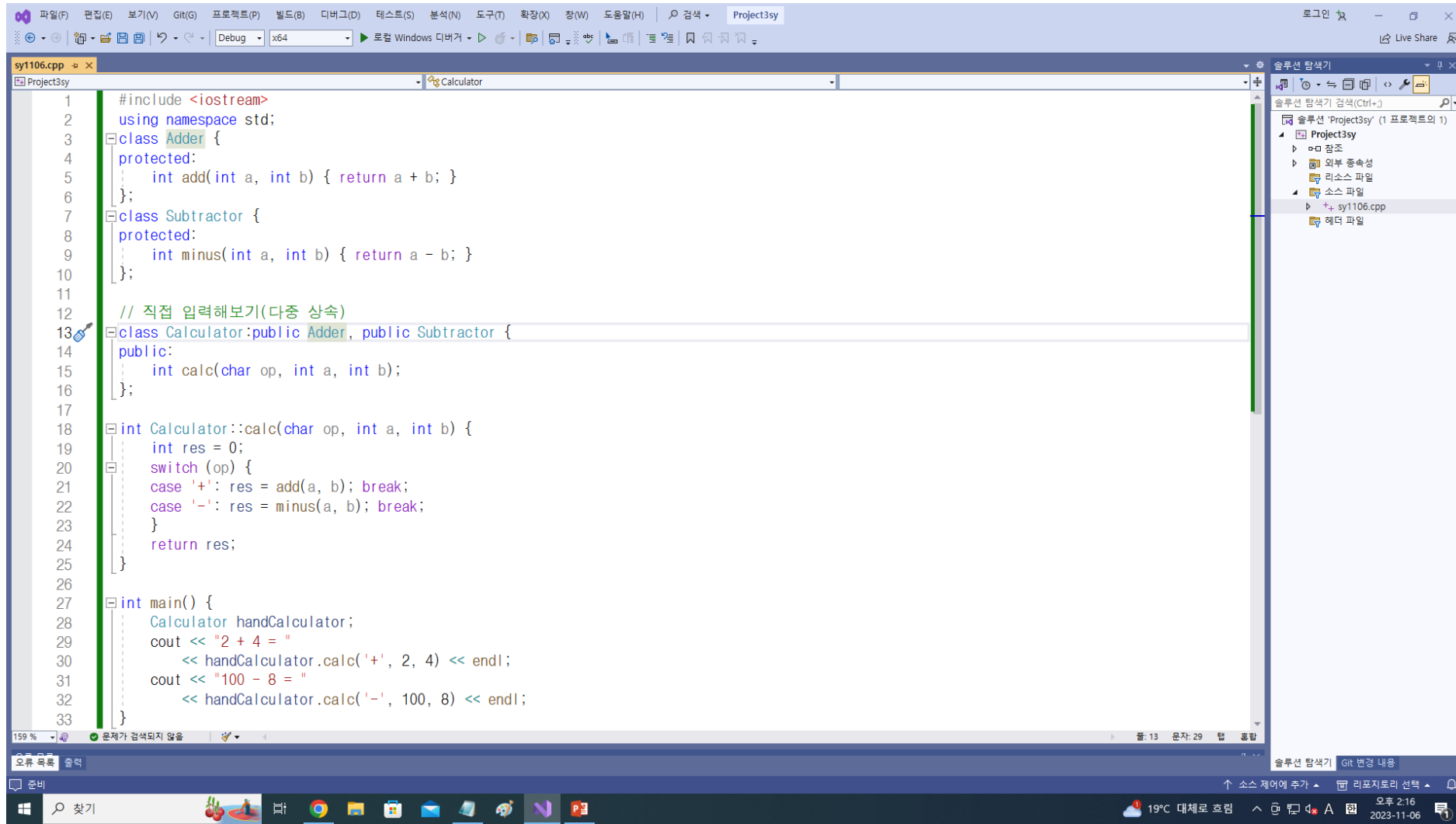
```
1  #include <iostream>
2
3  using namespace std;
4
5  class Adder {
6  protected:
7      int add(int a, int b) { return a + b; }
8  };
9
10 class Subtractor {
11 protected:
12     int minus(int a, int b) { return a - b; }
13 };
14
15 class Calculator : public Adder, public Subtractor {
16 public:
17     int calc(char op, int a, int b);
18 };
19
20 int Calculator::calc(char op, int a, int b) {
21     int res = 0;
22     switch (op) {
23     case '+': res = add(a, b);
24             break;
25     case '-': res = minus(a, b);
26             break;
27     }
28     return res;
29 }
30
31 int main() {
32     Calculator handCalculator;
33     cout << "2 + 4 = " << handCalculator.calc('+', 2, 4) << endl;
34     cout << "100 - 8 = " << handCalculator.calc('-', 100, 8) << endl;
35 }
36
```

The debug console window on the right shows the output of the program:

```
Microsoft Visual Studio 디버깅 콘솔
2 + 4 = 6
100 - 8 = 92
C:\Users\won2s\Desktop\객체지향프로그래밍\10주차 실습\10주차 실습\Debug\10주차 실습.exe (프로세스 3228개)
되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

[3차시] ppt p.35 예제 8-7

다중상속 구현 부분 안 보고 구현(이수영)



The screenshot shows a C++ IDE with a project named 'Project3sy'. The main file 'sy1106.cpp' is open, displaying a calculator program that demonstrates multiple inheritance. The program defines two base classes, 'Adder' and 'Subtractor', and a derived class 'Calculator' that inherits from both. The 'Calculator' class has a 'calc' method that uses the 'add' and 'minus' methods from the base classes. The 'main' function tests the calculator with two operations: '2 + 4' and '100 - 8'.

```
1 #include <iostream>
2 using namespace std;
3 class Adder {
4 protected:
5     int add(int a, int b) { return a + b; }
6 };
7 class Subtractor {
8 protected:
9     int minus(int a, int b) { return a - b; }
10 };
11
12 // 직접 입력해보기(다중 상속)
13 class Calculator:public Adder, public Subtractor {
14 public:
15     int calc(char op, int a, int b);
16 };
17
18 int Calculator::calc(char op, int a, int b) {
19     int res = 0;
20     switch (op) {
21     case '+': res = add(a, b); break;
22     case '-': res = minus(a, b); break;
23     }
24     return res;
25 }
26
27 int main() {
28     Calculator handCalculator;
29     cout << "2 + 4 = "
30         << handCalculator.calc('+', 2, 4) << endl;
31     cout << "100 - 8 = "
32         << handCalculator.calc('-', 100, 8) << endl;
33 }
```

수업내용 정리

C++에서의 상속

- 원준서: C++에서의 상속이란 클래스 사이의 상속관계를 정의하는 것으로, 객체 사이에는 상속 관계가 없습니다. 기본 클래스의 속성과 기능을 파생 클래스에 물려주는 것입니다. 기본 클래스는 상속해주는 클래스로 부모 클래스이며, 파생 클래스는 상속받는 클래스로 자식 클래스입니다. 파생 클래스는 기본 클래스의 속성과 기능을 물려받고 자신만의 속성과 기능을 추가하여 작성합니다. 기본 클래스에서 파생 클래스로 갈수록 클래스의 개념의 구체화되며, 다중 상속을 통한 클래스의 재활용성을 높입니다.
- 이수영: 기본 클래스와 파생 클래스는 유전적 상속으로 이해하면 좋고, 코드를 다시 안 짜도 재사용할 수 있다는 장점이 있음을 복습했습니다. 특히 다중 상속도 있구나를 새로 배웠습니다. 어떻게 코딩할 지 궁금했습니다.

상속의 목적&장점

- 원준서: 1, 간결한 클래스 작성이 가능: 기본 클래스의 기능을 물려받아, 파생 클래스를 간결하게 작성할 수 있습니다. / 2, 클래스 간의 계층적 분류 및 관리의 용이함: 상속은 클래스들의 구조적 관계 파악이 용이합니다. / 3. 클래스 재사용과 확장을 통한 소프트웨어 생산성 향상: 빠른 소프트웨어의 생산이 필요할 때, 기존에 작성한 클래스의 재사용을 할 수 있으며, 상속받아 새로운 기능을 확장할 수도 있습니다. 때문에 상속에 대비한 클래스의 객체 지향적 설계가 필요합니다. / 상속을 통해 기존에 작성한 클래스를 재사용하고, 새로운 기능을 확장할 수도 있다는 점을 새롭게 알게 되었습니다.
- 이수영: C와 달리 객체 지향 언어인 C++은 상속의 개념이 있습니다. 폴드폰처럼 기존 핸드폰의 속성(수신/발신)이 있으면서 고유의 기능(듀얼 디스플레이)도 갖고, 기본 속성(수신/발신)에 문제가 있을 때, 이 기본 속성을 상속받는 자식 클래스 A, B, C, D의 코드를 일일이 수정하지 않고, 부모 클래스만 수정하면 됩니다.

protected 접근 지정

- 원준서: 접근 지정자는 private 멤버, public 멤버, protected 멤버가 존재합니다. private 멤버는 선언된 클래스 내에서만 접근이 가능하며, 파생 클래스에서도 기본 클래스의 private 멤버에 직접 접근이 불가능합니다. public 멤버는 선언된 클래스나 외부 어떤 클래스, 모든 외부 함수에 접근을 허용하며, 파생 클래스에서 기본 클래스의 public 멤버 접근이 가능합니다. protected 멤버는 선언된 클래스에서 접근이 가능하며, 파생 클래스에서만 접근이 허용됩니다. 파생 클래스가 아닌 다른 클래스나 외부 함수에서는 protected 멤버를 접근할 수 없습니다.
- 이수영: private은 선언된 클래스 내에서만, public은 외부에서도 가능한데, protected는 오직 파생(자식) 클래스에서만 가능합니다. 다만 앞선 예제 8-2, 8-5를 실습하며 놓쳤던 것처럼 main()은 외부 함수기 때문에, main()안에선 아무리 자식 클래스 객체(예를 들면, 예제 8-2의 cp, 예제 8-5의 Derived x)라도 protected로 선언된 멤버에 접근할 수 없습니다. 이 점은 두 예제에서 모두 고민하다가, 강의 후반부 3차시쯤 해결됐기 때문에 오래 기억될 것 같고, 확실히 습득한 개념이었습니다.

상속의 생성자, 소멸자

- 원준서: Q1, 파생 클래스의 객체가 생성될 때 파생 클래스의 생성자와 기본 클래스의 생성자가 모두 실행되는가? 아니면 파생 클래스의 생성자만 실행되는가? / 답 : 둘 다 실행됩니다. / Q2, 파생 클래스의 생성자와 기본 클래스의 생성자 중 어떤 생성자가 먼저 실행되는가? / 답 : 기본 클래스의 생성자가 먼저 실행된 후 파생 클래스의 생성자가 실행됩니다.
- 이수영: 기본(부모) 클래스에 기본 생성자가 없다면 컴파일러가 자동으로 매개변수가 없는 기본 생성자를 생성해주지 않는 점이 중간고사 이전에 배우던 내용과 다른 부분이었고, 컴파일 오류를 낸다는 점이었습니다. 그래서 기본 클래스에는 기본 생성자를 비롯한 생성자가 꼭 필요하다는 점을 배웠습니다. 특히 파생 클래스에 매개 변수가 있어도 컴파일러는 묵시적으로 기본 생성자를 호출되기 때문에, 매개 변수가 있는 생성자를 호출하고 싶다면 명시적으로 생성자를 선택해야 한다는 점을 새로 배웠습니다.

소멸자의 실행 순서

- 원준서: 파생 클래스의 객체가 소멸될 때, 파생 클래스의 소멸자가 먼저 실행되고 기본 클래스의 소멸자가 나중에 실행됩니다.
- 이수영: 생성자 실행 순서의 역순입니다. 생성자는 기본 클래스가 먼저, 파생 클래스가 다음에 실행되므로, 소멸자는 파생 클래스가 먼저, 기본 클래스가 마지막으로 실행됩니다. 예를 들어 A-B-C와 같은 상속 관계로, A가 조상, B가 부모, C가 자식이라면, 생성자는 A, B, C, 소멸자는 C, B, A 순으로 실행될 것입니다.

상속의 종류: 상속 지정

- 원준서: 상속 지정은 상속 선언 시 public, private, protected의 3가지 중 하나를 지정하며, 기본 클래스의 멤버의 접근 속성을 어떻게 계승할지 지정합니다. /
public – 기본 클래스의 protected, public 멤버 속성을 그대로 계승합니다. /
private – 기본 클래스의 protected, public 멤버를 private으로 계승합니다. /
protected – 기본 클래스의 protected, public 멤버를 protected로 계승합니다.
- 이수영: "class 부모 클래스명:상속 지정 자식 클래스명"의 구조일 때, public이면 부모 클래스의 속성을 public으로 계승합니다. 이 자식 클래스는 public이므로 다른 클래스가 접근할 수 있습니다. / 반면 private이면 부모 클래스의 속성을 private으로 계승하는데, 이 자식 클래스의 멤버는 자식 클래스 내부에서만 접근 가능(private)합니다. / 그리고 protected이면 이 자식 클래스를 상속받는 클래스만 이 자식 클래스에 접근할 수 있기 때문에, public과 달리 외부 클래스에서 이 자식 클래스에 접근할 수 없다는 점이 특징입니다.

다중 상속

- 원준서: 하나의 파생 클래스가 여러 클래스를 동시에 상속받는 것으로, 다중 상속을 통해 여러 기능을 통합한 클래스를 만들 수 있습니다. / 다중 상속으로 파생 클래스를 선언할 때는 클래스 선언문에서 접근 지정과 함께 기본 클래스를 콤마(,)로 나열하면 됩니다.
- 이수영: 부모 클래스가 1개가 아닌 2개 이상인 경우를 말합니다. 1개일 땐 "class 자식 클래스명 : public 부모 클래스명"인데(일반적인 경우로 public으로 선언하겠습니다), 다중 상속일 땐 "class 자식 클래스명 : public 부모 클래스명, public 부모 클래스명"과 같이 부모 클래스를 콤마로 연결해서 표현합니다. 그런데 이럴 때 기본 클래스의 멤버가 중복 상속되어, 이 멤버를 main()에서 접근하려고 할 때, 여러 자식 클래스와 부모 클래스에 모두 이 멤버가 있기 때문에 어느 멤버에 접근해야 할 지 컴파일러가 모호한 상황에 처하고 실행 오류가 납니다. 이 문제를 해결한 것이 가상 상속입니다. 1차시엔 다중 상속이 있구나! 새롭다. 라고 생각했는데 이런 문제가 있다는 걸 알고 나니 그래서 이런 코드를 많이 못 봤던 거구나.를 깨달았고, 이것을 어떻게 해결할지 궁금한 마음이 커져 가상 상속에 대한 흥미가 더 높아졌습니다.

가상 상속

- 원준서: 다중 상속으로 인한 기본 클래스 멤버의 중복 상속을 해결합니다. 가상 상속은 파생 클래스의 선언문에서 기본 클래스 앞에 virtual로 선언하며, 파생 클래스의 객체가 생성될 때 기본 클래스의 멤버는 오직 한번만 생성되는데, 이는 기본 클래스의 멤버가 중복하여 생성되는 것을 방지합니다.
- 이수영: 자식 클래스+콜론+public 부모 클래스의 형태에서 콜론 뒤, public 앞에 virtual 키워드를 선언하면 됩니다. 가상 상속은 부모 클래스의 멤버가 부모 클래스에 오직 한 번만 생성된단 걸 나타냅니다. 즉 아무리 In, Out같은 여러 자식 클래스가 부모 클래스를 상속해도, 강의자료의 mode라는 멤버변수는 기본 클래스에만 존재하기 때문에 main()에서 mode 변수에 값을 지정할 때, 컴파일러는 어느 클래스(부모/자식)에 있는 mode를 찾아야 하는지 모호하지 않고, 기본 클래스에 있는 mode에 접근합니다.