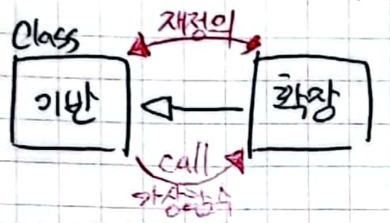
7강. 상속 심화.

* 가상함수 : 상속 + 재정의. (Frame work) ~ "미래"를 호출.

S/W = Data / GUI / Ctrl.



⇒ 문법
virtual int TestFunc();
↳ 가상화 선언.

일반메서드는 참조형식을 따르고, 가상함수는 실형식을 따른다.

함수 ┬ 일반.
└ 가상 [순수가상. ⇒ only 선언. 정의X.

⇒ 가상함수 : 이후 재정의시 virtual 선언 없이도 가상함수가 됨.

⇒ final 키워드 : 재정의 시 compile error.

*소멸자 가상화.

A에 대한 파생 B가 있을 때.

delete pdata;

↳ A의 소멸자 호출. B소멸자 호출X

⇒ memory leak

↳ A의 소멸자를 virtual로 선언.

접근형식 + 실형식.

접근형식 : Pointer*, Rep & ...

A *pdata = new B;
접근형식 ↳ 추상자료형. / 실형식 ↳ 파생.
↳ 기본
⇒ 접근형식 (기본) ← 실형식 (파생)

☆ ↳ B에서 A의 함수 재정의 가능.
⇒ pdata에서 호출 ⇒ 접근형식(A)의 코드 call.
⇒ 만일 virtual 선언시 ⇒ 실형식(B)의 코드 call

(소멸자 : call, 실행 순서 원치)

* 순수가상 클래스. (pure virtual) ⋯→ 선언o, 정의X

흐름을 설계적으로만 명시

virtual int GetData() const = 0 ← 순수가상함수.

↳ 파생클래스에서 순수가상함수 반드시 구현.

↳ 인터페이스 용도 - 파생클래스에 맞게 구현할 수 있도록.

S/W

(시간적 여유. O)

처리위해 준비 [수집(입력)] ← HID (Human Input Device)

↓

처리 ← 고속. (유휴X → 성능저하)

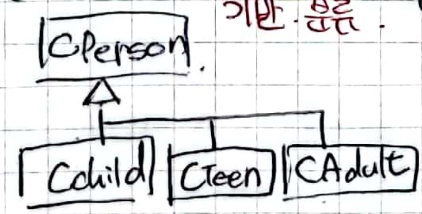
↓

결과

↳ Look-up table.

↳ 객체지향 → 입력에서 분류.

(요금계산 ex) 추상자료형 기반 분류.



CPerson pPerson.
switch (나이)
case
pPerson = new CChild.

[V] CalcFee() = 0

↳ child / Teen / Adult 각각의 함수 재정의.

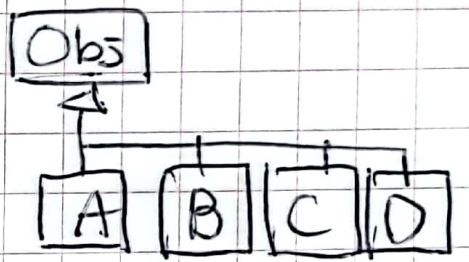
YEAR MONTH DAY

* Static Cast.

△ ① const_cast< >( ) const 제거

○ ② static_cast< > → Compile
× [③ dynamic_cast< >] 상·하향 → Runtime

× ④ Reinterpret_cast<> → C언어 강제형변환 : → 사용빈도↓
↳ C의 type cast 지양.



Obj * pobj = new A;
pobj → ?일 때 역계산 ③
(A*)pobj ⇒ static_cast
↳ 상·하향 이외의 강제적 캐스트는 컴파일 오류로 알림.

* 상속과 연산자 다중정의
↳ 파생형식의 클래스에서 부모클래스의 연산자를 정의하지 않은 경우.
부모클래스의 연산자를 그대로 사용할 때.
⇒ using 부모클래스 :: operator (+...등);

* 다중상속 ⇒ 지양 (모호성) (예외. 인터페이스 다중상속)
↳ 인터페이스 다중상속. → 한계의 강치에 활용법이 여러개.