5장 연산자 다중정의

C++ ⇒ 연산자 재정의, 다중정의 지원 ⇒ class 에 한해서

↳ 함수화 · ⇒사용자 코드를 간결하게 ·

a + = 3; (+= 연산) vs a.Append(3, int);

⇒ 직관, + 추상성↑ 실수가능성↓ 유지보수 비용↓

• 기본연산자 - fail X ⇒실패가능성 없는 코드. (예외처리)

6장. 상속 기본.

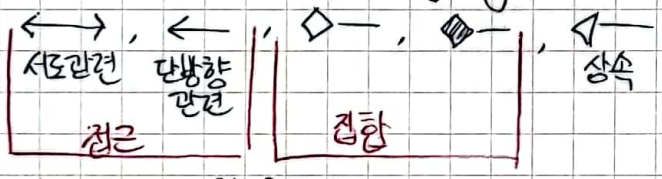
* 상속: ① 코드를 재사용하는 방법 中 1.

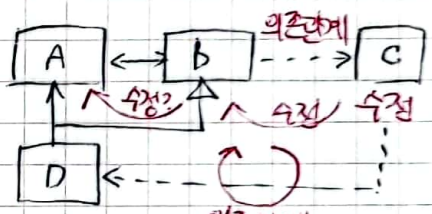
② 규모 확장, 개선 ·

③ 관계의 한 유형.

설계. ← 객체(대상) 규정 + 관계정의

↳ UML : unified modeling language.





과거 CMyData 부모클래스 Base 기본 (이미 되어있는것)

현재 CMyDataEx 자식클래스 Drived 파생 (구현)

CMyDataEx a;

CMyData 사람

의사.

시점 코드의 흐름 / 뮤텍의 전환

OOP ⇒ 설계.

모듈화 프로그래밍.

기존것에서 구조만 개선 ⇒ 리팩토링

설계적 수정.

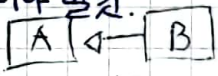
S/W → ① UI + ② Pg + ③ Data

↳ 나누어야 할것.

* 설계 원칙 (SOLID)

관계 : 의존관계 - 상속 : 존재의 의존 / 집합

문법 : class A {}; class B : public A {};



↳ 상속 ⇒ 누가? 언제? 파생 (미래) 초기개발자 ⇒ 확장 (타 개발자)

* 참조형식과 실형식.

★ 파생형식 ← 기본형식 (pointing, 사람) → 추상형식

파생 ← 기본 → 추상

기본형식 = 파생형식 ㅁ

↳ 반대 불가능.

메서드 ┌ 일반 ← 접근형식 따름

├ 가상 ← 실형식 따름.

└ static.

생성자 → '객체 자신 초기화' 이니

(상속) 파생형식 생성자 → 부모클래스 초기화 ⇒ XXXX