01

객체지향 프로그래밍의 개념

학습내용

01 객체지향 프로그래밍의 개념

02 객체지향의 핵심 개념

03 프로그래밍 방법론의 비교

학습목표

- 객체지향 프로그래밍(OOP)의 핵심 원리에 대해 설명할 수 있다.
- 객체지향의 핵심 개념(추상화, 캡슐화, 상속, 다형성)을 이해하고, 각 개념을 코드로 표현하고 차이를 구분할 수 있다.
- 절차지향, 객체지향, 함수형 프로그래밍 방법론의 특징, 장단점, 적용 사례를 분석하여 상황에 맞는 프로그래밍 방식을 선택할 수 있다.

지난 주차 복습

스코프(Scope)

- 변수나 함수가 유효한 범위(접근 가능한 범위)를 의미하며, 코드의 구조와 관련됨

스코프의 종류	전역 스코프, 지역 스코프(함수/블록)로 나뉘며, let, const, var에 따라 범위가 다름 - var : 함수 스코프 - let, const : 블록 스코프
스코프 체인 (Scope Chain)	내부 함수가 외부 스코프에 연쇄적으로 접근하는 구조로, 변수 탐색 시 위로 거슬러 올라감
렉시컬 스코프 (Lexical Scope)	함수가 정의된 문법적 위치에 따라 스코프가 결정됨(실행 위치가 아니라 정의 위치 기준)

지난 주차 복습

고급 함수

즉시 실행 함수 (IIFE)	• 정의와 동시에 한 번만 실행되는 함수 • 전역 변수 오염 방지에 활용
중첩 함수 (Nested Function)	• 함수 안에 또 다른 함수를 정의하여 은닉화 • 클로저 생성 등에 사용
콜백 함수 (Callback Function)	• 다른 함수에 인자로 전달되어 나중에 호출되는 함수 • 주로 이벤트 처리, 비동기 로직에 사용

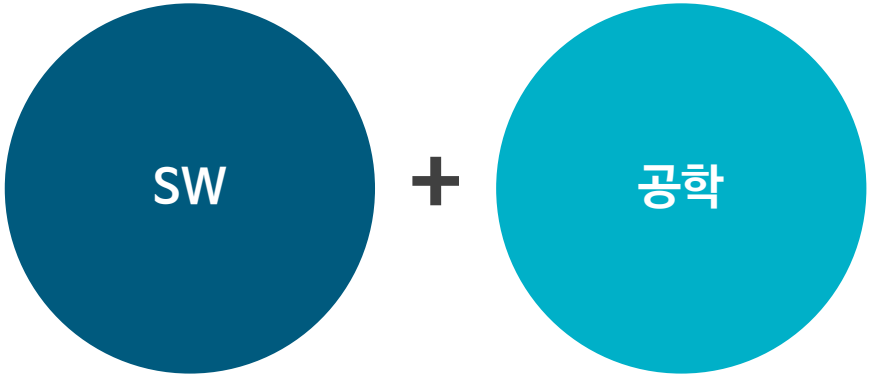
생각 해보기

Q 함수형 애플리케이션은 “데이터를 변형하지 않고”, “순수 함수들을 조합하여”, “예측 가능한 흐름”을 만드는 데 중점을 둡니다. 이때 고차함수를 사용하면 얻을 수 있는 장점은 어떤 것들이 있을지 생각해 보고, 게시판에 올려주세요.

01

객체지향 프로그래밍의 개념

1) SW공학

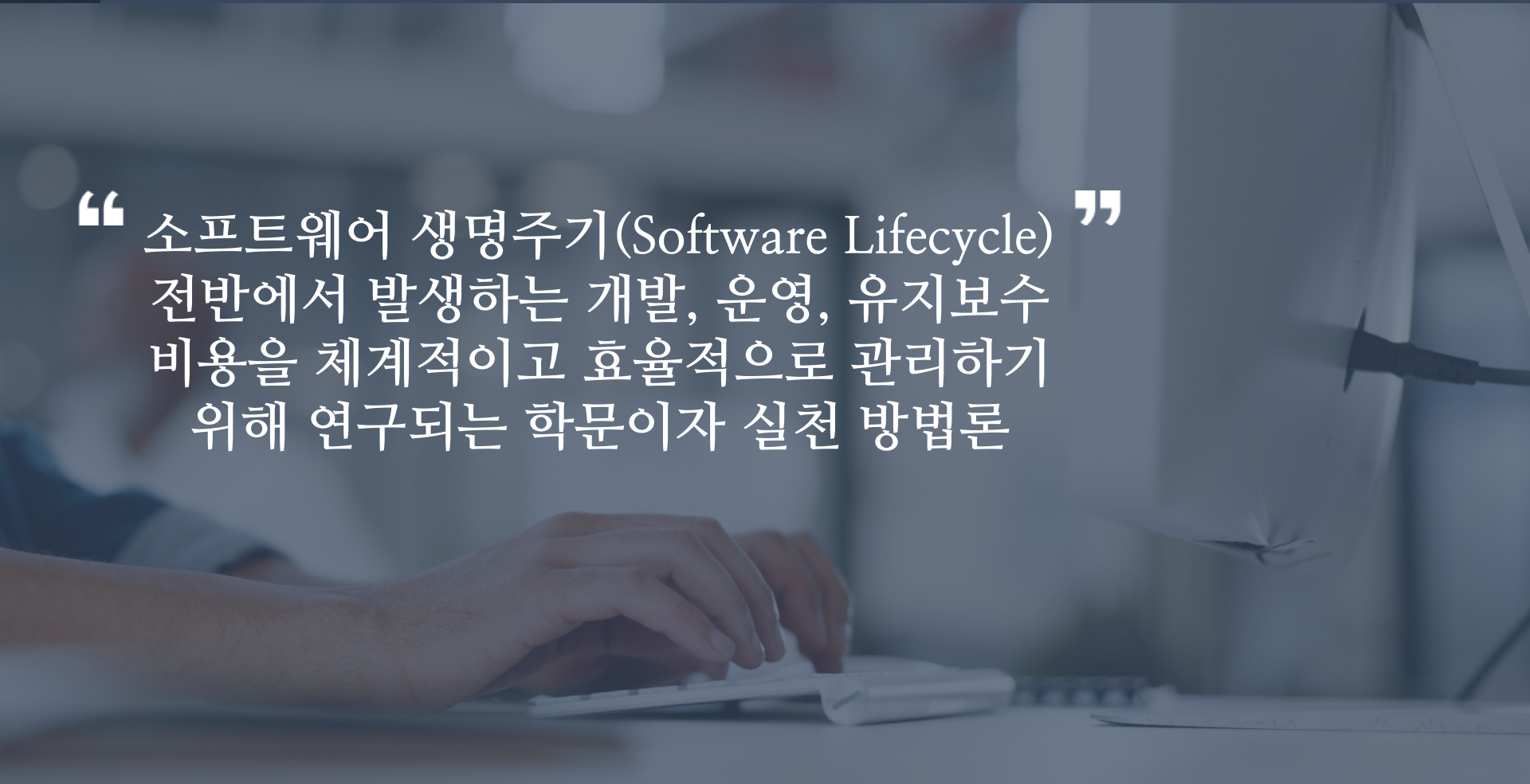


“ 소프트웨어 개발을 공학적으로 접근 ”

(자연과학을 바탕으로)
최소 비용(비용, 시간, 인력)으로
최대 효과(성능, 안정성, 효율성)

1) SW공학

세종사이버대학교



“ 소프트웨어 생명주기(Software Lifecycle) 전반에서 발생하는 개발, 운영, 유지보수 비용을 체계적이고 효율적으로 관리하기 위해 연구되는 학문이자 실천 방법론 ”

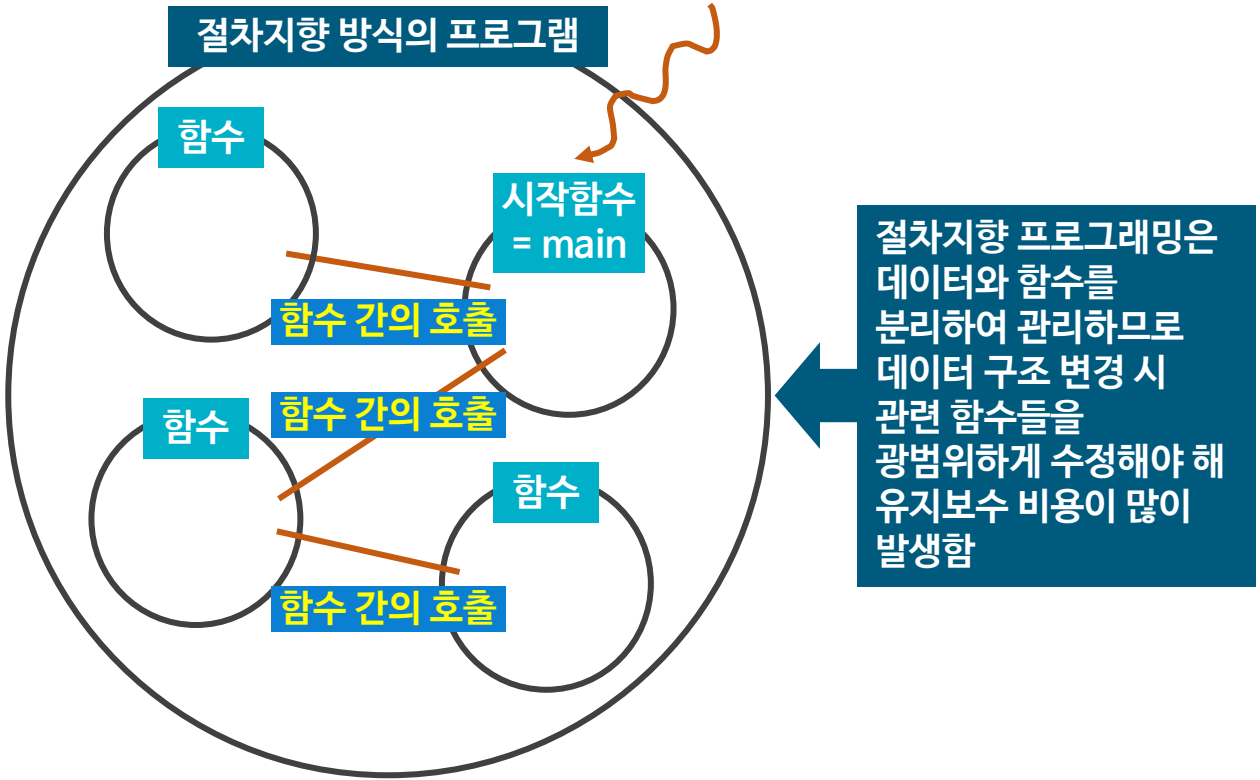
연구 영역

구분	프로젝트 진행 방법론	프로그래밍 방법론
관점	소프트웨어 개발 프로세스의 전체 흐름에 대한 관리 관점	코드 작성 방식에 대한 설계 및 구현 관점
목적	개발 과정을 효율적이고 예측 가능하게 관리	코드의 구조, 재사용성, 유지보수성 향상
적용 범위	요구 분석 ➡ 설계 ➡ 구현 ➡ 테스트 ➡ 배포까지의 전체 과정	주로 구현(코딩) 단계에서 적용
대표 예시	폭포수 모델, 애자일 방법, 스크럼, XP, DevOps	절차지향, 객체지향, 함수형, 선언형
중심 요소	일정 관리, 역할 분담, 반복 주기, 피드백	데이터 구조, 함수/객체 설계, 모듈화 방식

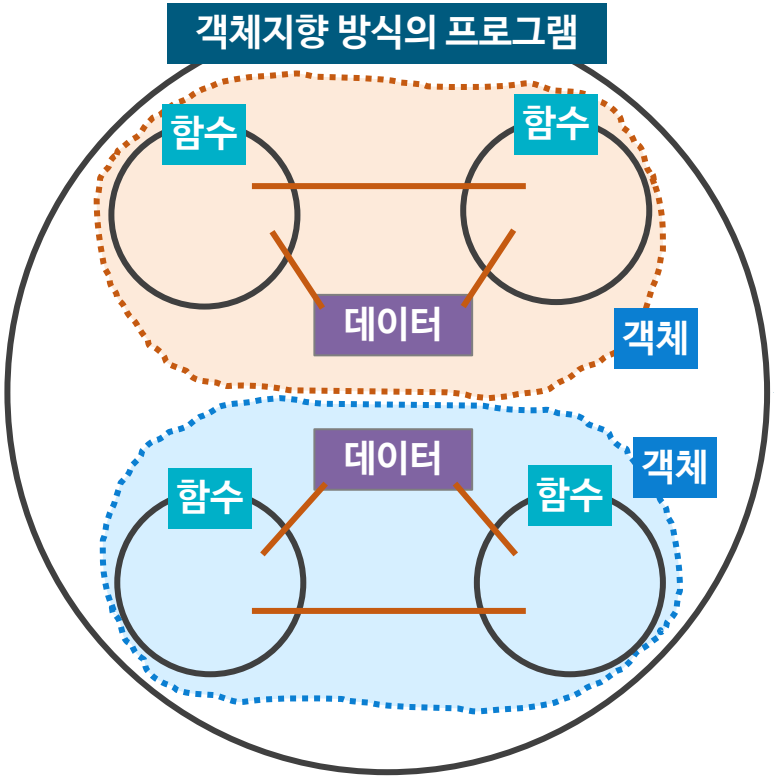
연구 영역

구분	프로젝트 진행 방법론	프로그래밍 방법론
등장 배경	프로젝트 규모 증가로 인한 관리 문제 해결	코드 복잡도 증가로 인한 설계 문제 해결
성과 지표	일정 준수, 팀 협업, 품질 보증	코드 재사용성, 유지보수성, 확장성
관계	'어떻게 만들 것인가?'에 대한 전략	'어떻게 짤 것인가'에 대한 전술

- 절차지향 vs. 객체지향


절차지향 방식의 프로그램
함수
시작함수
= main
함수 간의 호출
함수
함수 간의 호출
함수
함수 간의 호출
절차지향 프로그래밍은
데이터와 함수를
분리하여 관리하므로
데이터 구조 변경 시
관련 함수들을
광범위하게 수정해야 해
유지보수 비용이 많이
발생함

절차지향 vs. 객체지향

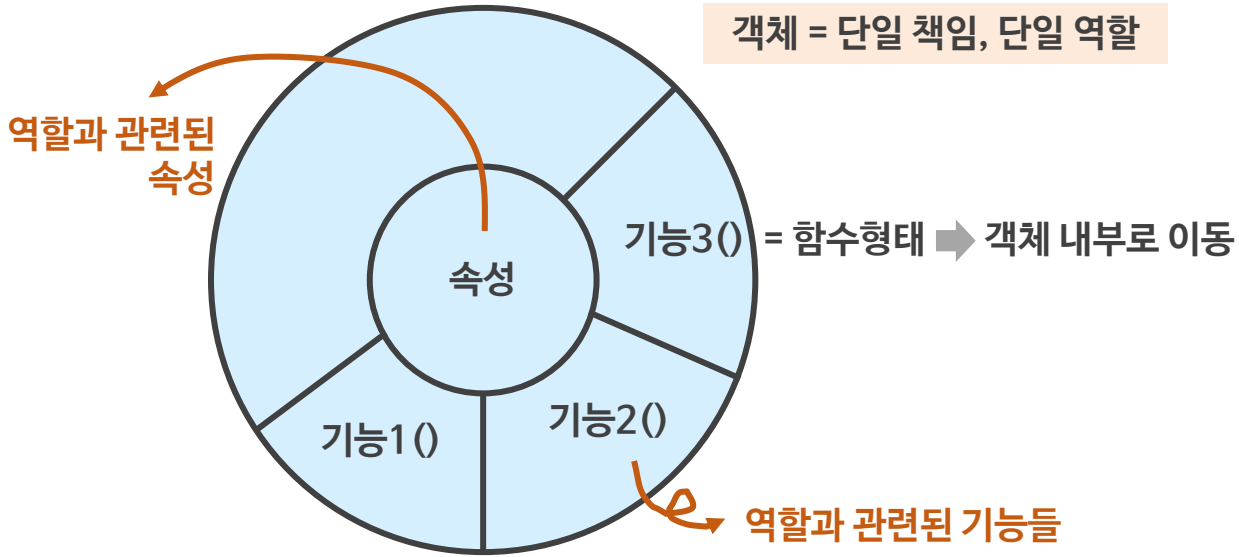


객체지향 프로그래밍은 데이터와 기능을 객체(단일 역할) 단위로 캡슐화하여 변경 범위를 최소화하고, 재사용성을 높이므로 유지보수 비용을 효과적으로 줄일 수 있음

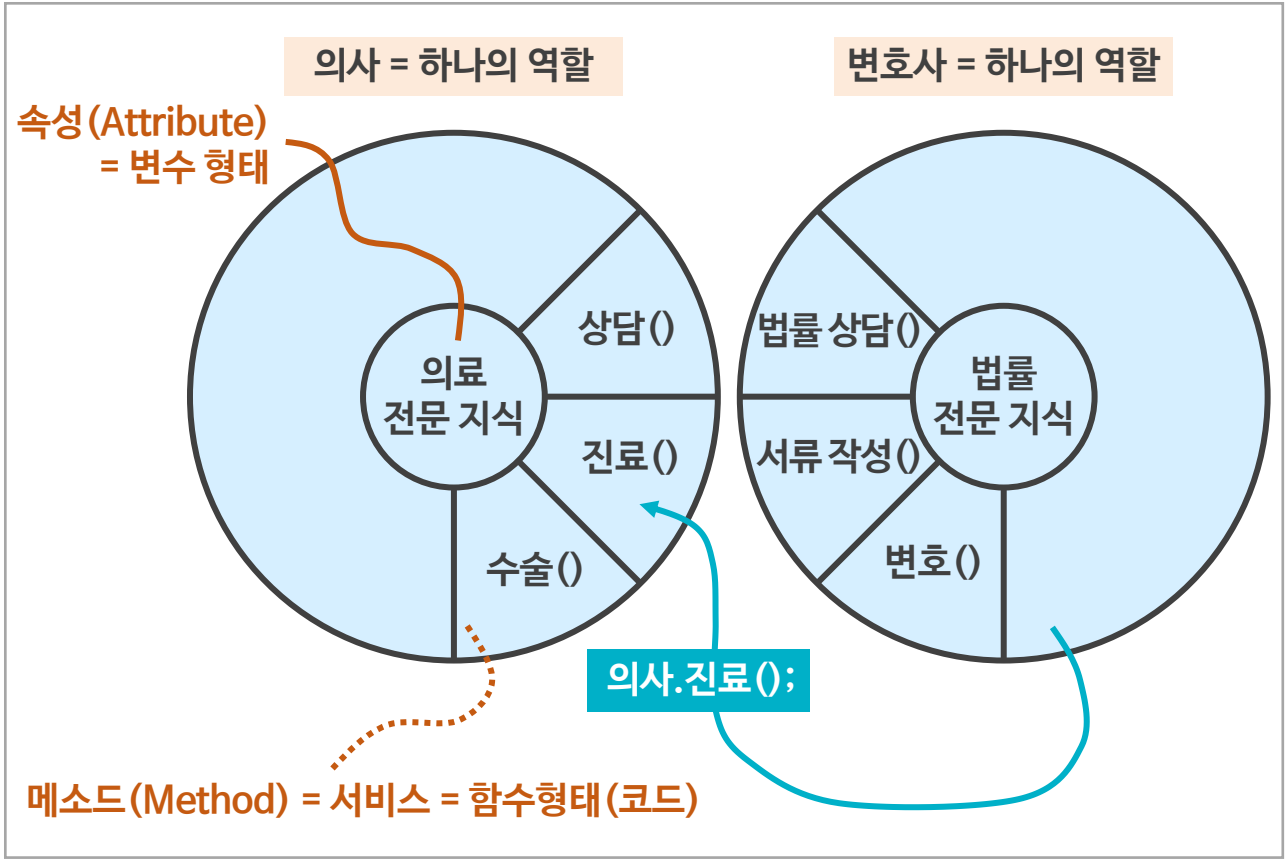
➡ 응집도 ↑, 결합도 ↓

3) 객체의 의미

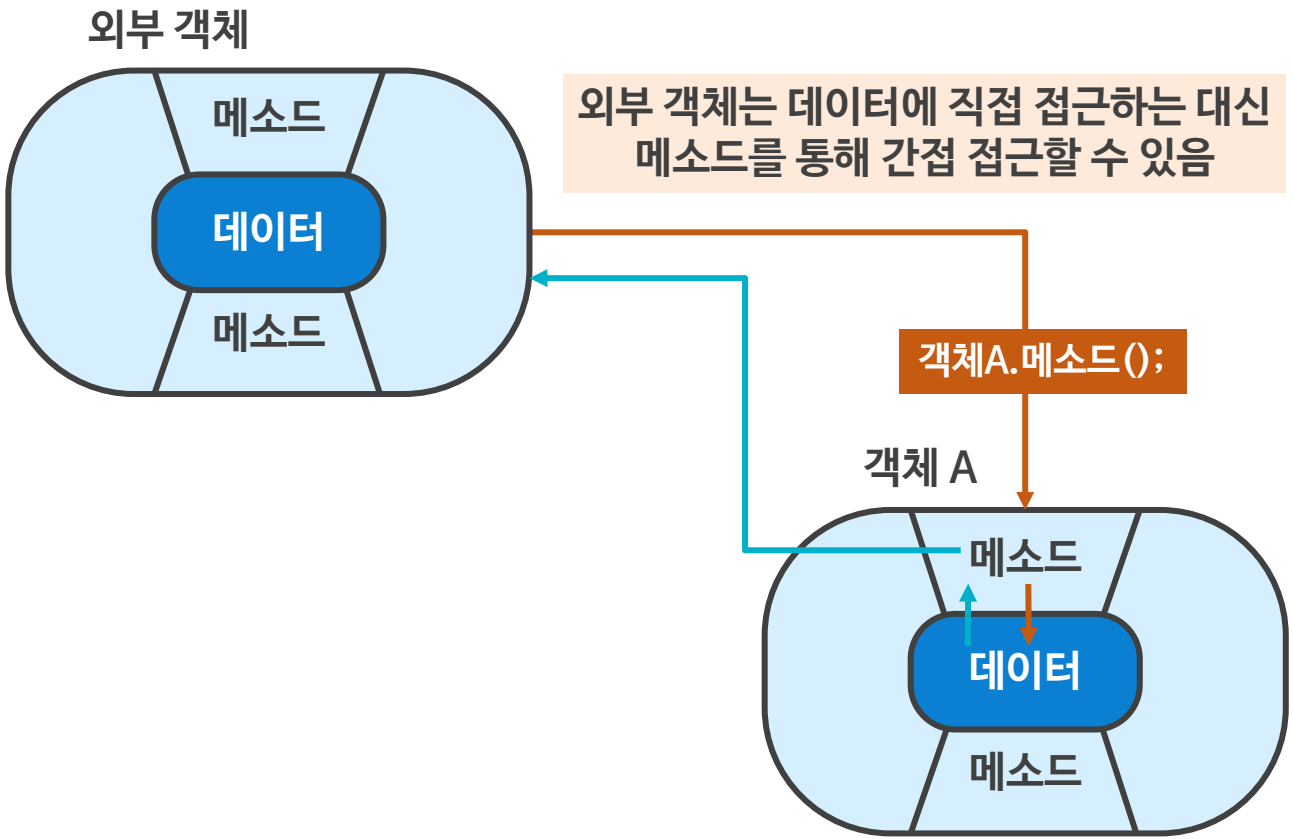
객체 구성 형태	다수의 데이터 + 다수의 기능(함수)
객체의 의미	단일 책임, 단일 역할 수행



4) 객체지향 프로그램의 구성도 예시



4) 객체지향 프로그램의 구성도 예시



02

객체지향의 핵심 개념

Object

= 단일 책임 = 단일 역할 = 관련된 다수의 기능 + 관련된 다수의 데이터

= 독립적 실행 단위 = Code + Data

2) 객체지향 패러다임의 기본 개념

캡슐화	데이터와 메서드를 하나로 묶고 외부에서 직접 접근하지 못하도록 보호하는 것
추상화	복잡한 현실을 핵심적인 개념이나 동작만 드러내어 단순화하는 것
다형성	동일한 인터페이스로 서로 다른 동작을 실행할 수 있는 능력

[ES6]

```
class Account {
  #balance = 0;  // #으로 선언하면 private 필드
  deposit(amount) { //예금
    if (amount > 0) this.#balance += amount;
  }
  getBalance() {
    return this.#balance;
  }
}

const  acc = new Account(); //객체생성➔역할을 수행할 실체
acc.deposit(1000);
console.log(acc.getBalance());  // 1000
// console.log(acc.#balance);   // 에러: private 필드 접근 불가
```

[ES6]

```
class Car {
  drive() {
    console.log("자동차가 달립니다.");
  }
}

const myCar = new Car();
myCar.drive();  // 복잡한 엔진 작동은 숨기고 "달린다"는 행위만 드러냄
```

5) 다형성

[ES6]

```
class Animal {
  speak() {
    console.log("동물이 소리를 냅니다.");
  }
}
class Dog extends Animal {
  speak() {
    console.log("멍멍!");
  }
}
class Cat extends Animal {
  speak() {
    console.log("야옹~");
  }
}
```

```
const animals = [new Dog(),
                 new Cat(),
                 new Animal()];


//같은 메서드 이름으로 다른 클래스가 각기 다르게 동작함
animals.forEach(animal => animal.speak());

// 결과:
// 멍멍!
// 야옹~
// 동물이 소리를 냅니다.
```

03

프로그래밍 방법론의 비교

1) 절차지향 vs. 객체지향 vs. 함수지향

항목	절차지향 프로그래밍 (Procedural)	객체지향 프로그래밍 (OOP)	함수형 프로그래밍 (Functional)
핵심 단위	함수(절차, 알고리즘)	객체(데이터+행동)	순수 함수, 고차 함수
중심 개념	순차적 처리 흐름	역할, 캡슐화, 상속	불변성, 함수를 값으로 처리
데이터와 로직	분리	통합	완전 분리
재사용성	낮음(복사)	높음(상속, 캡슐화)	매우 높음(함수 조합)
변화 대응력	약함	보통	높음(데이터 독립)
대표 언어	C, Pascal	Java, Python, C++	Haskell, JS(ES6+)
SW공학 내 역할	초기 개발비용 절감	유지보수 효율화	복잡도 축소

06주. 유지보수 비용을 줄이자! 객체지향 이야기

02

클래스와 인스턴스 활용