# 프론트 엔드 개발자가 알아야 하는 컴퓨터 공학 지식

## 소프트웨어 공학(SW Engineering)

소프트웨어 공학 | 프론드 엔드 개발자가 알아야 하는 CS 지식

강사 나동빈

# 프론트 엔드 개발자가 알아야 하는 컴퓨터 공학 지식

## 소프트웨어 공학(SW Engineering)

# 절차 지향 프로그래밍



- **절차 지향** 프로그램은 위에서부터 차례대로 순서에 맞게 코드를 작성하고 실행한다.
- 일반적인 컴퓨터의 작업 처리 방식과 유사하다는 점에서, 통상적으로 객체 지향 방식보다 빠르다.

# 객체 지향 프로그래밍

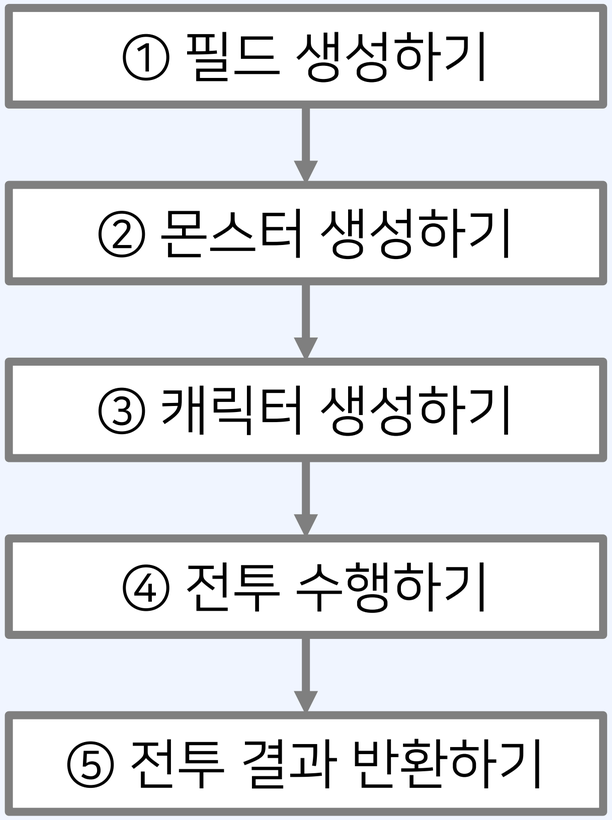
컴퓨터
공학 지식
소프트웨어
공학

- 현실 세계의 **사물(개체)**를 코드로 모델링하여 프로그램을 작성하는 방식이다.
- 다양한 부품을 조합하여 하나의 완성된 프로그램을 조립하는 것에 비유할 수 있다.
- 코드의 재 사용성이 높지만, 일반적으로 절차 지향 방식에 비해 느리다.

# 게임 프로그램 동작 예시(절차 지향)



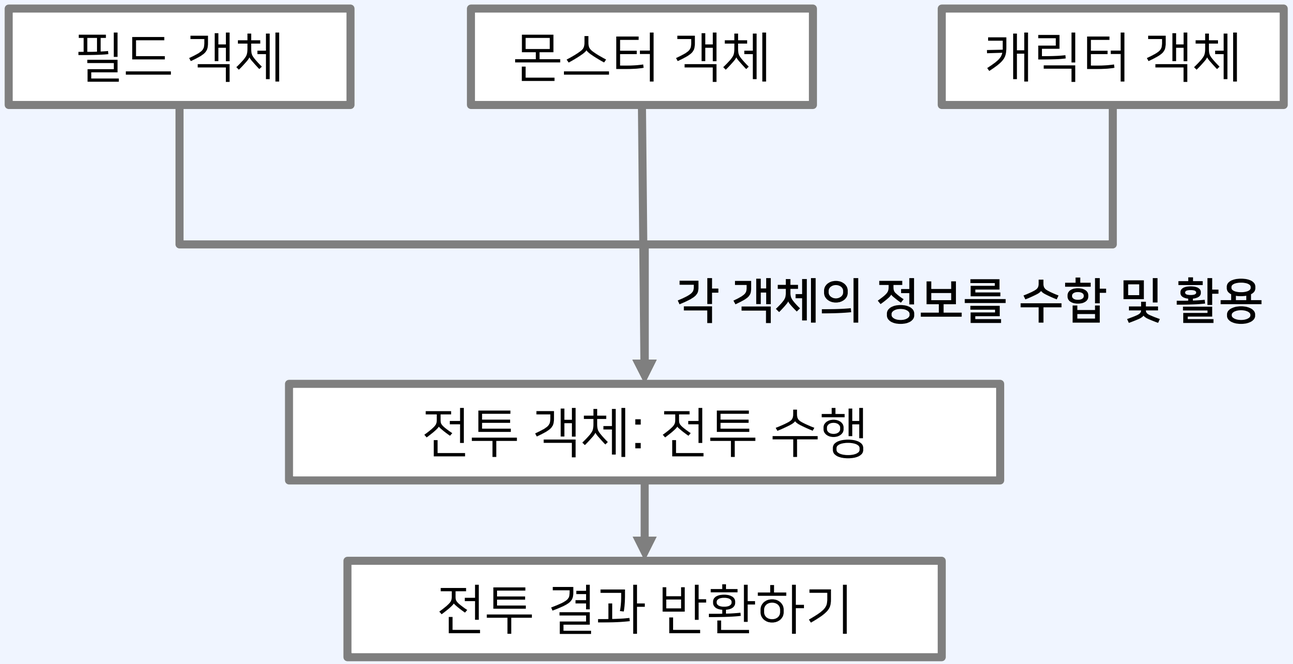
- 절차 지향 프로그램은 위에서부터 차례대로 순서에 맞게 코드를 작성하고 실행한다.



# 게임 프로그램 동작 예시(객체 지향)



- 현실 세계의 **사물(개체)**를 코드로 모델링하여 프로그램을 작성하는 방식이다.



# 객체 지향 프로그래밍의 특징



1. **추상화(abstraction)**: 객체들의 공통적인 속성과 메서드를 정의한다.
- 클래스(class)를 정의하는 과정 자체를 일종의 추상화로 볼 수 있다.
2. **상속(inheritance)**: 하나의 클래스가 가진 정보(속성, 메서드)를 다른 클래스가 물려 받아 사용한다.
- 코드의 재사용성을 높여 생산성을 높일 수 있다.
3. **캡슐화(encapsulation)**: 실제 구현부를 외부에 드러나지 않도록 은닉한다.

→ 각 객체가 독립적으로 기능하며, 데이터와 기능(메서드)을 하나로 묶어 관리할 수 있다.

4. **다형성(polymorphism)**: 다른 방법으로 동작하는 함수를, 동일한 함수명으로 호출할 수 있다.

1) **오버라이딩**: 부모 클래스의 메소드와 같은 이름 및 매개변수를 갖는 코드를 재정의한다.

2) **오버로딩**: 함수 이름은 같지만 매개변수가 다른 함수를 정의한 뒤에, 필요에 따라 호출해 사용한다.

# 객체 지향 프로그래밍의 특징



1. 객체 지향 프로그래밍은 **상속**, **다형성** 등을 통해 프로그램의 재사용성을 높인다.
2. 프로그램의 업그레이드, 디버깅 과정을 효율적으로 하여 **생산성**을 높인다.
3. 현실 세계의 객체를 기준으로 코드를 작성하여 모델링한다.

→ 학생에 대한 기능을 처리하는 코드는, **학생(Student)** 클래스로 정의하여 사용할 수 있다.

4. 캡슐화 덕분에 실제 코드 구현이 외부에 직접적으로 드러나지 않아, 상대적으로 보안성이 높은 편이다.

→ 객체의 멤버 변수에 접근하기 위해 일반적으로 getter, setter와 같은 함수를 사용해야 한다.

5. 반면에 객체 지향 프로그래밍은 객체를 관리하는 측면에서 추가적인 메모리가 요구될 수 있으며, 상대적으로 속도가 느리다.

# 절차 지향 vs. 객체 지향



- 일반적으로 절차지향 방식의 속도(C언어)가 더 빠르다.
- 하지만 유지보수 및 생산성 측면에서 한계가 존재한다.
- 반면에 객체지향 방식은 유지보수가 편리하여 대형 프로젝트에 적합하다.
- 일반적으로 절차지향 방식보다 처리 속도가 느리고, 다수의 설계 코드 조각 및 설계로 구성된다.

| | 절차 지향 | 객체 지향 |
|---|---|---|
| **대표 언어** | C언어 | Python, C++, Java |
| **속도** | 빠른 편 | 비교적 느린 편 |
| **추상성** | 낮은 편 | 높은 편 |