

**초격차 패키지 Online.**

# Start with Boot!

## **PART1** | 강사 소개, 사용 도구 소개

오리엔테이션 느낌으로 가볍게

## **PART2** | 프로젝트 셋업

자바 프로젝트 만들기, 깃헙에 올리기

## **PART3** | OOP 로 만든 정렬 구현체

Spring 공부를 위한 워밍업

## **PART4** | Spring의 적용

Spring 이 무엇을 도와주는가? 왜 써야 하는가?

## **PART5** | Spring Boot의 적용

왜 Spring Boot 로 시작해야 하는가?

# Start with Boot!

**3** OOP로 만든 정렬 구현체 3종

## OOP 프로그래밍

### 3. OOP로 만든 정렬 구현체 3종

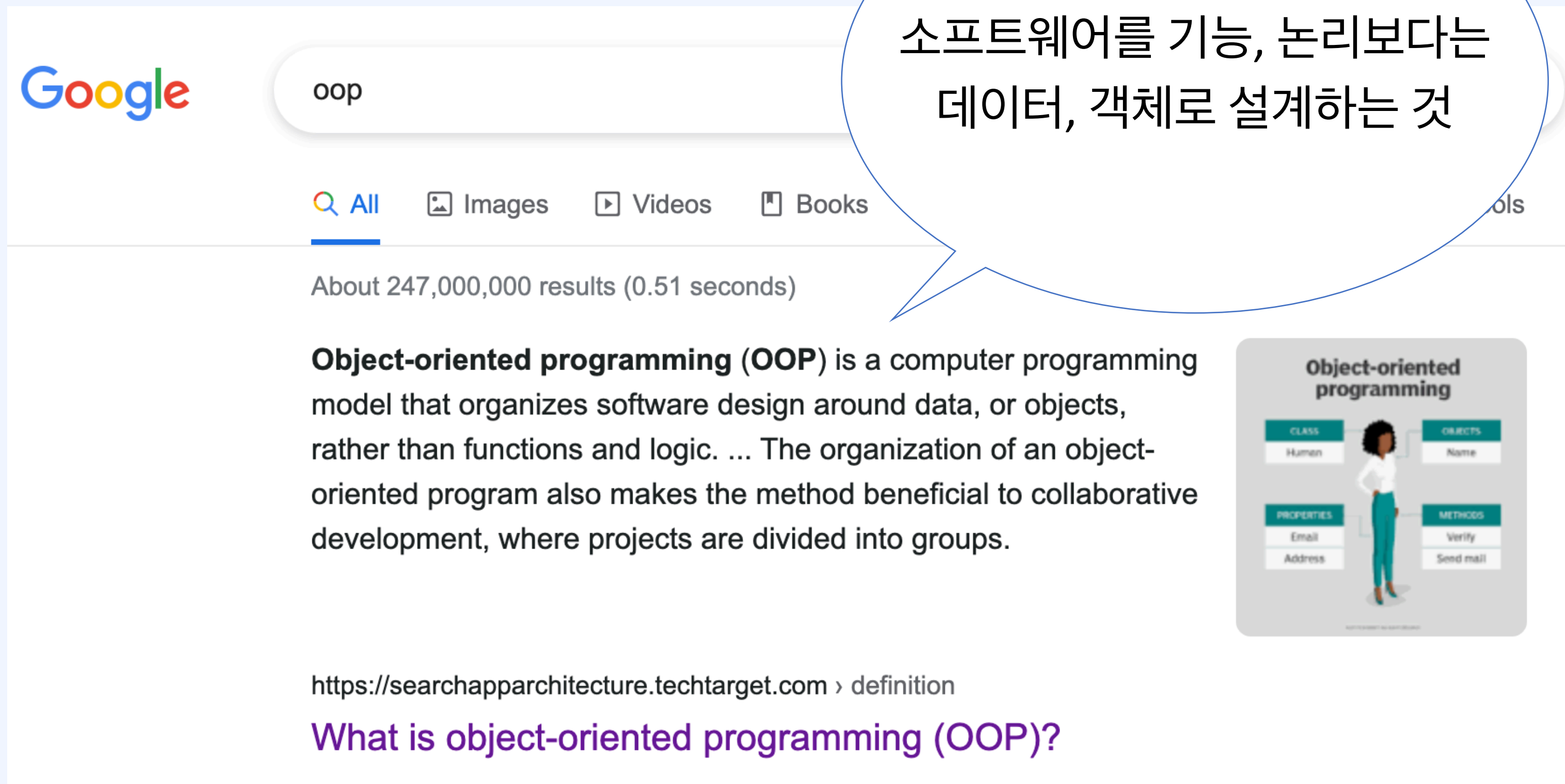
## 갑자기 왜 OOP? 정렬?

- 스프링 부트 강의의 시작점: "왜 써야 하는가?"
  - 스프링 부트를 왜 쓰는지 알고 싶다: 스프링만 써보자
  - 스프링을 왜 쓰는지 알고 싶다: 스프링 없이 개발해보자
- 우리가 하려는 것
  - 그냥 자바 애플리케이션을 만든다
  - 스프링을 적용해본다: 무엇이 편리해졌는지 느껴본다
  - 부트를 적용해본다: 무엇이 더욱 편리해졌는지 느껴본다

## OOP 프로그래밍

3.  
OOP로 만든  
정렬 구현체 3종

### 소프트웨어를 OOP 로 설계하는 것



Google

oop

All Images Videos Books


About 247,000,000 results (0.51 seconds)

**Object-oriented programming (OOP)** is a computer programming model that organizes software design around data, or objects, rather than functions and logic. ... The organization of an object-oriented program also makes the method beneficial to collaborative development, where projects are divided into groups.

<https://searcharchitecture.techtarget.com › definition>

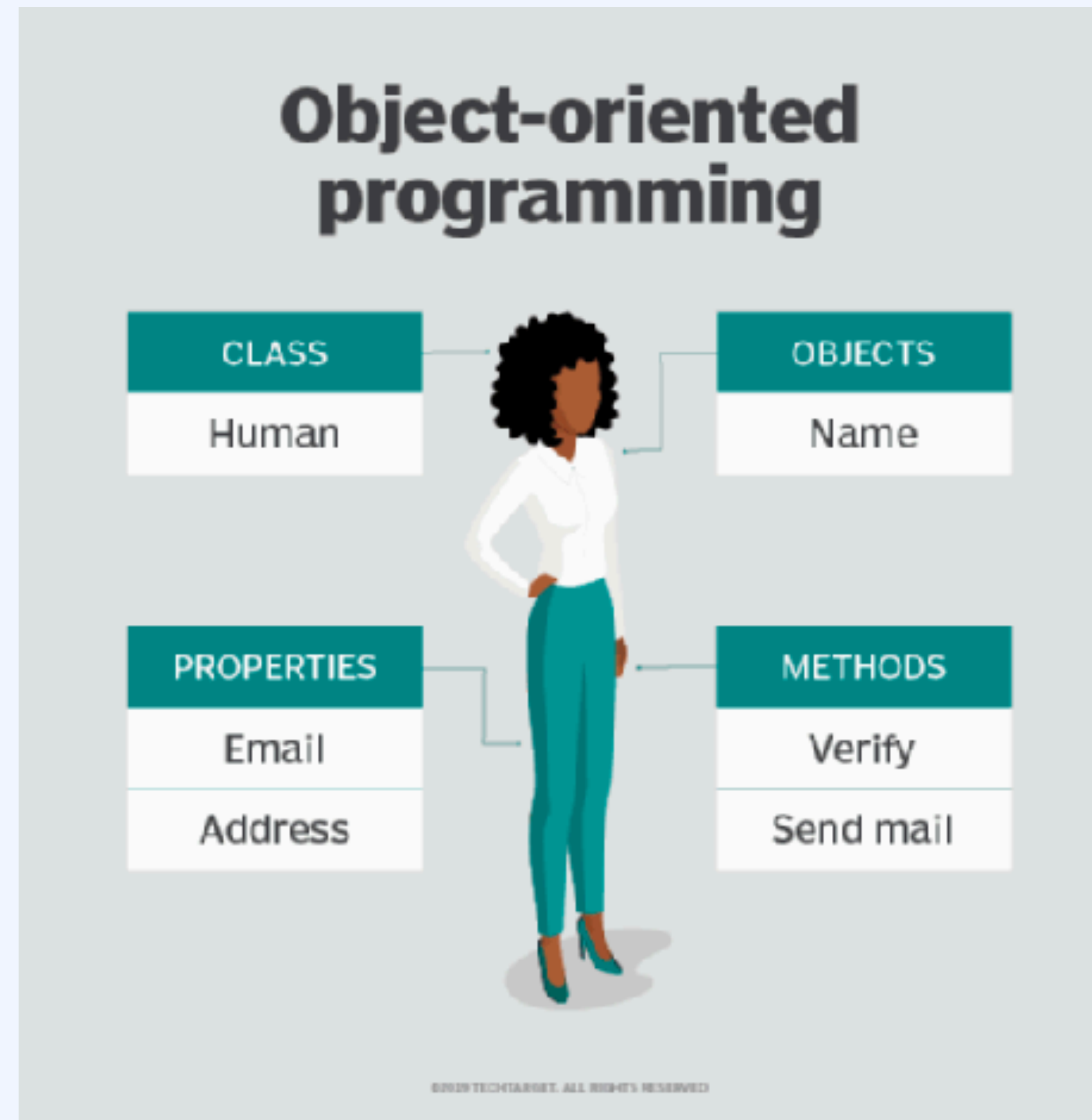
What is object-oriented programming (OOP)?

소프트웨어를 기능, 논리보다는 데이터, 객체로 설계하는 것



## OOP 프로그래밍

3.  
OOP로 만든  
정렬 구현체 3종




출처: <https://searcharchitecture.techtarget.com/definition/object-oriented-programming-OOP>

## OOP 프로그래밍

3.  
OOP로 만든  
정렬 구현체 3종

# 객체 지향 설계 (SOLID)



### 위키백과

우리 모두의 백과사전

[대문](#)  
[최근 바뀜](#)  
[요즘 화제](#)  
[임의의 문서로](#)  
[기부](#)

[사용자 모임](#)

[사랑방](#)  
[사용자 모임](#)  
[관리 요청](#)

[편집 안내](#)

[도움말](#)  
[정책과 지침](#)  
[질문방](#)

[도구](#)  
[여기를 가리키는 문서](#)  
[가리키는 글의 최근 바뀜](#)  
[파일 올리기](#)  
[특수 문서 목록](#)  
[특수 검색](#)

문서

토론

## SOLID (객체 지향 설계)

위키백과, 우리 모두의 백과사전.

컴퓨터 프로그래밍에서 **SOLID**란 **로버트 마틴**<sup>[1][2]</sup>이 2000년대 초반<sup>[3]</sup>에 명명한 **객체 지향 프로그래밍** 및 **설계**의 다섯 가지 기본 원칙을 마이클 페더스가 **두문자어 기억술**로 소개한 것이다. 프로그래머가 시간이 지나도 **유지 보수**와 확장이 쉬운 시스템을 만들고자 할 때 이 원칙들을 함께 적용할 수 있다.<sup>[3]</sup> SOLID 원칙들은 소프트웨어 작업에서 프로그래머가 **소스 코드**가 읽기 쉽고 확장하기 쉽게 될 때까지 소프트웨어 소스 코드를 **리팩터링**하여 **코드 냄새**를 제거하기 위해 적용할 수 있는 지침이다. 이 원칙들은 **애자일 소프트웨어 개발**과 **적응적 소프트웨어 개발**의 전반적 전략의 일부다.<sup>[3]</sup>

목차 [숨기기]

1 개요

2 같이 보기

2.1 기본 개념과 연관 주제

2.2 설계와 개발 원칙

3 출처

## OOP 프로그래밍

3.  
OOP로 만든  
정렬 구현체 3종

### 객체 지향 설계 (SOLID)

- SRP: 한 클래스는 하나의 책임만 가져야 한다.
- OCP: 소프트웨어 요소는 확장에는 열려 있으나 변경에는 닫혀 있어야 한다.
- LSP: “프로그램의 객체는 프로그램의 정확성을 깨뜨리지 않으면서 하위 타입의 인스턴스로 바꿀 수 있어야 한다.”
- ISP: “특정 클라이언트를 위한 인터페이스 여러 개가 범용 인터페이스 하나보다 낫다.”
- DIP: 프로그래머는 “추상화에 의존해야지, 구체화에 의존하면 안된다.”

그래서 하긴 했는데...

3.  
OOP로 만든  
정렬 구현체 3종

To Be Continued...

어떻게 해야 잘하지?

언제 다 하지?

비즈니스는 언제 처리하지?