

객체 지향 프로그래밍

OOP (Object-Oriented Programming)

Giftbot

Programming Paradigm

패러다임 : 어떤 시대·분야에서의 특징적인 사고 방식·인식의 체계, 틀

프로그램을 설계하는 방식에 대한 개념 / 방법론

프로그래머에게 프로그래밍의 관점을 갖게 해주고 결정하는 역할

- 비구조적 (Non-Structured) / 구조적 (Structured)
- 명령형 (Imperative) / 선언형 (Declarative)
- 절차적 (Procedural) / 객체지향 (Object-Oriented) / 함수형 (Functional)
- 배치 (Batch) / 이벤트 기반 (Event-Based)
- etc...

Structured / Object-Oriented

구조적 (Structured)

에초 허브 데이터스트럭처

Sequence, Alternative, Iteration

Divide and conquer

Top-down design

Pascal, C

객체지향 (Object-Oriented)

불리고 케이스

Class, object

Encapsulation, Inheritance, Polymorphism

Simula 67, Smalltalk, C++, Java

Edsger Wybe Dijkstra



에츠허르 데이크스트라 (1930~ 2002)

Edsger Wybe Dijkstra, 네덜란드의 컴퓨터 과학자, 1972년 휴링상 수상

- (모든 고수준 언어의 조상)인 ALGOL 개발 ('50년대 말)
- (1968) [Goto Statement Considered Harmful]
[소프트웨어 공학]과 Structured Programming 시대를 여심
- [데이크스트라 알고리즘], 처음으로 최단거리 알고리즘을 학문적으로 연구
- [세마포어]의 개념 사용, 운영체제의 영역까지 지대한 영향
- 1972 [Turing Award] 수상

Alan Curtis Kay

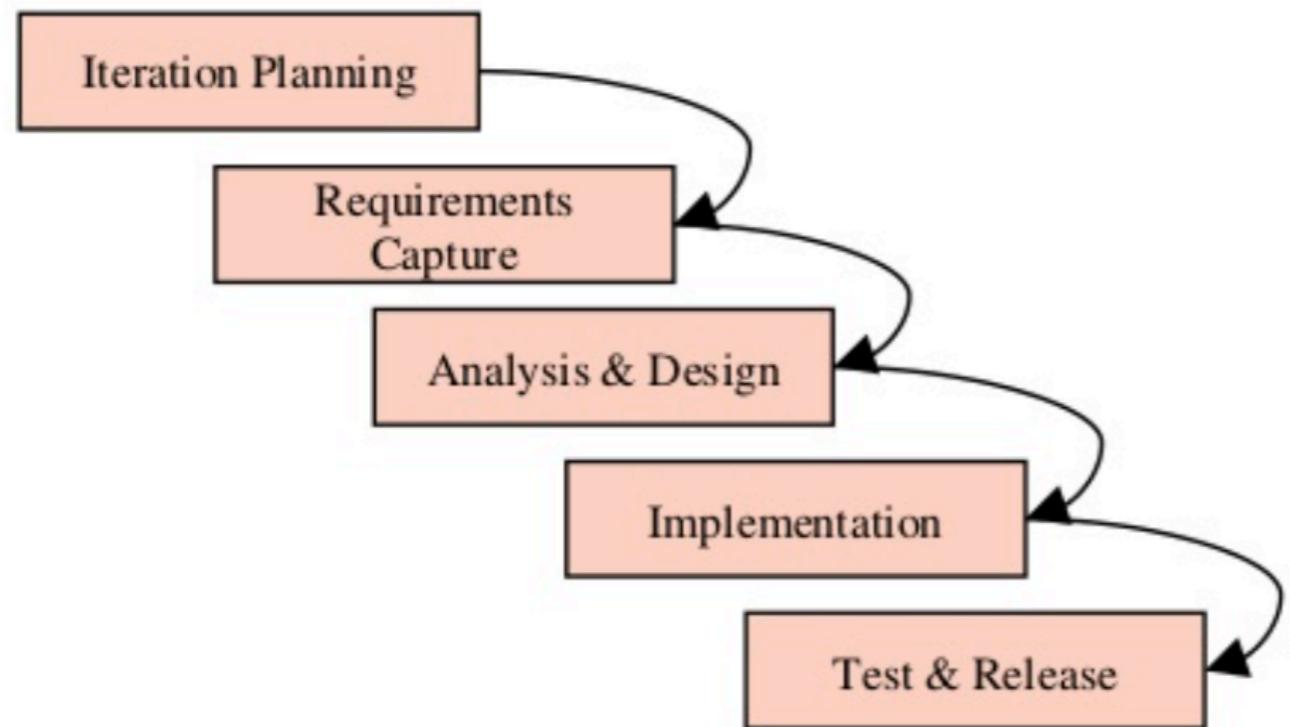


알런 케이
(1940~)

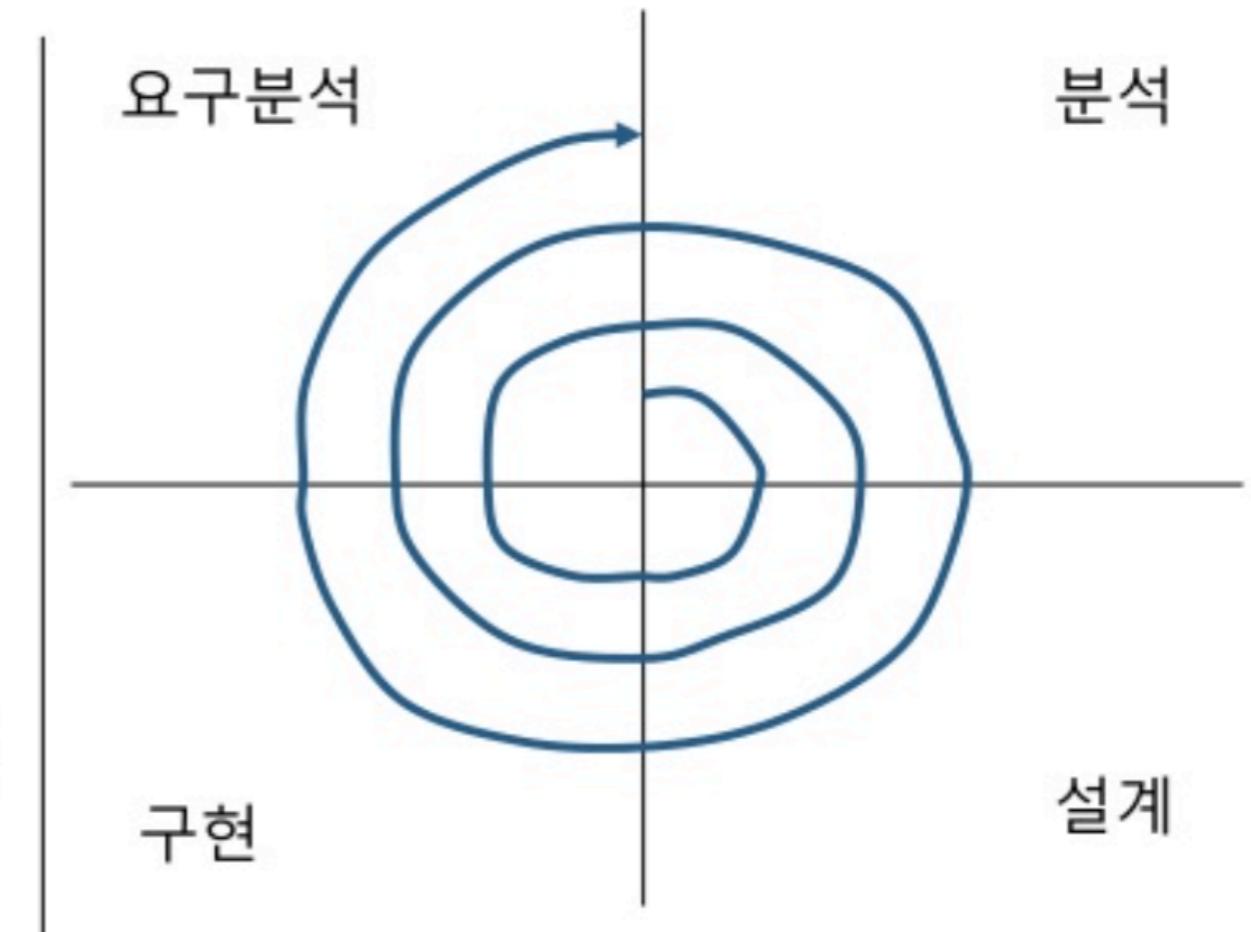
Alan curtis Kay 미국의 컴퓨터 과학자

- 이반 세덜랜드와 함께 스카치패드 개발(1960년대 유타대학교)
- 제록스 파크(PARC)에서 Smalltalk 개발(70년대)
- 객체지향 프로그래밍을 개척한 공로로 휴링상(2003)

Software development process



폭포수로델(Waterfall model)



나선 모델(Spiral model)

Multi Paradigm Language

Objective-C : OOP 언어

Swift : POP 를 지향하는 멀티 패러다임 언어

[주요 패러다임]

- POP : Protocol-Oriented Programming
- OOP : Object-Oriented Programming
- FP : Functional Programming

Object-Oriented Programming

객체지향 프로그래밍 이란 캡슐화, 다형성, 상속 을 이용하여 코드 재사용을 증가시키고, 유지보수를 감소시키는 장점을 얻기 위해서 객체들을 연결시켜 프로그래밍 하는 것

언어 또는 기술이 다음 사항들을 직접 지원한다면 객체 지향

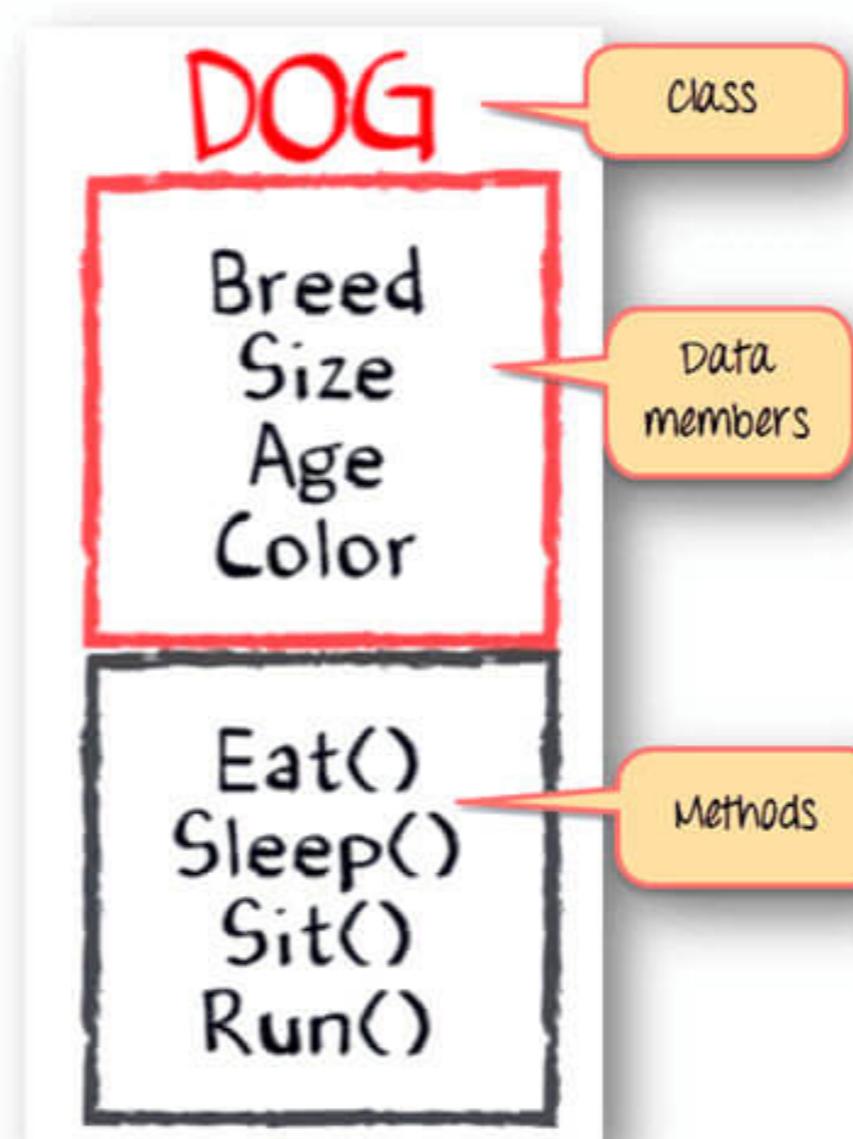
- 추상화 : 클래스나 객체를 제공
- 상속 : 이미 존재하는 것으로부터 새로운 추상화를 만들어 낼 능력을 제공
- 런타임 다형성 : 수행 시간에 바인딩 할 수 있는 어떠한 폼을 제공

Object-Oriented Programming

단순한 데이터 처리 흐름에서 벗어나 각 역할을 지닌 객체들의 상호작용으로 동작

객체 : 데이터 (상태) + 메서드 (행위)

최초의 OOP 언어 : Smalltalk / Smalltalk + C → Objective-C



Swift Class

```
class ClassName {  
  
    var variable1 = 1  
    var variable2 = "2"  
  
    func functionName1(param: Int) {  
        // code  
    }  
  
    func functionName2(param: String) {  
        // code  
    }  
}
```

Swift Class

```
class ClassName {
```

```
    var variable1 = 1  
    var variable2 = "2"
```

속성, 데이터, 상태

```
        func functionName1(param: Int) {  
            // code  
        }
```

```
        func functionName2(param: String) {  
            // code  
        }
```

클래스 (class)

행위, 메서드, 동작

```
}
```

Free Function vs Method

Q. 다음 용어의 차이점은?

- 함수 (Function) ?
- 메서드 (Method) ?

Object

쉽게 배우는 소프트웨어 공학

객체

실세계에 존재하거나 생각할 수 있는 것을 객체(object)라고 한다. 흔히 볼 수 있는 책상, 의자, 전화기 같은 사물은 물론이고 강의, 수강 신청 같은 개념으로 존재하는 것도 모두 객체이다. 다시 말해 사전에 나와 있는 명사뿐 아니라 동사의 명사형까지도 모두 객체인 것이다. 그리고 더 넓게 보면 인간이 생각하고 표현할 수 있는 모든 것이 객체이다.

이런 객체는 관점에 따라 다음과 같이 여러 개념으로 이해된다.

- **모델링 관점** : 객체는 명확한 의미를 담고 있는 대상 또는 개념이다.
- **프로그래머 관점** : 객체는 클래스에서 생성된 변수이다.
- **소프트웨어 개발 관점** : 객체는 소프트웨어 개발 대상으로, 어떤 한 시점에 객체 상태를 나타내는 데이터와 해당 데이터를 처리하고 참조하는 동작을 의미하는 메서드(함수)를 모아놓은 '데이터+메서드' 형태의 소프트웨어 모듈이다.
- **객체지향 프로그래밍 관점** : 객체는 데이터와 함수를 속성(attribute)과 메서드(method) 용어로 구현한다.

Objects

손님 객체



손님은 메뉴판에서 커피를 선택할 수 있다

손님은 바리스타에게 커피를 주문한다

메뉴판 객체

Menu	
아메리카노	1,500원
카푸치노	2,000원
캬라멜 마키아또	2,500원
에스프레소	2,500원

메뉴 항목
객체들

캬라멜 마키아또 객체



커피를 제조한다

아메리카노 객체

에스프레소 객체

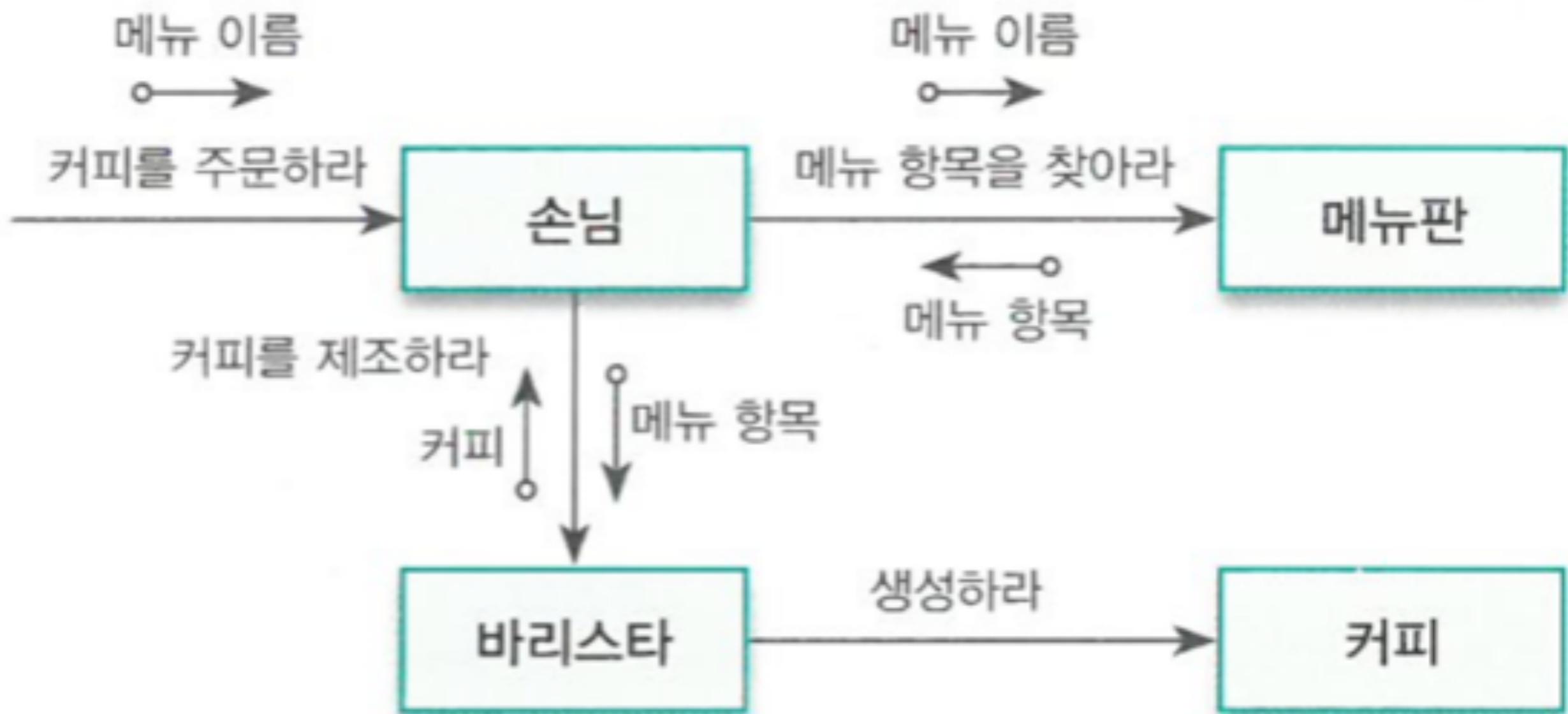
카푸치노 객체



바리스타 객체

Message

객체 지향 프로그래밍 - 각 객체와 그 객체들간의 관계를 설계하는 것



Class, Object

"많은 개체들이 공통된 명칭을 가질 때 그것들은 언제나 또 하나의 이데아, 즉 형상을 갖는다. 가령 침대는 무수히 많지만 침대의 이데아, 즉 그 형상은 오직 하나이다. 여러가지 개개의 침대는 실재가 아니며 오직 그 이데아의 모사(模寫)에 의해 만들어졌을 뿐이다." - 플라톤

[Class]

- 추상 (abstract) , 표현 대상에 대한 이데아(형상)
- 이상적인 존재 (이미지, 설계도, 틀, 설명서)
- 공통의 특징

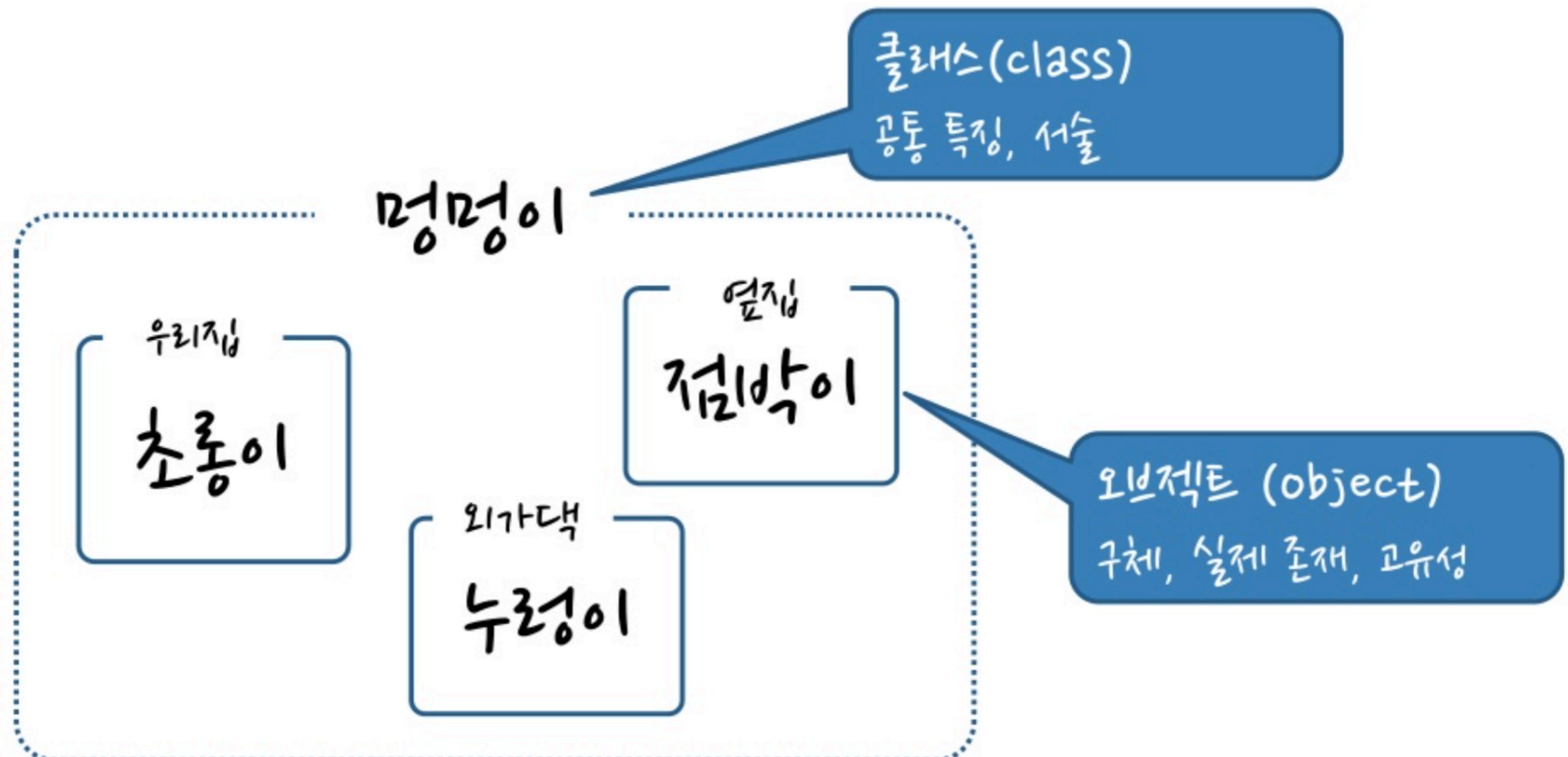
[Object]

- 실제 (instance) , 추상을 실체화한 대상
- 이데아의 모사
- 개별 속성

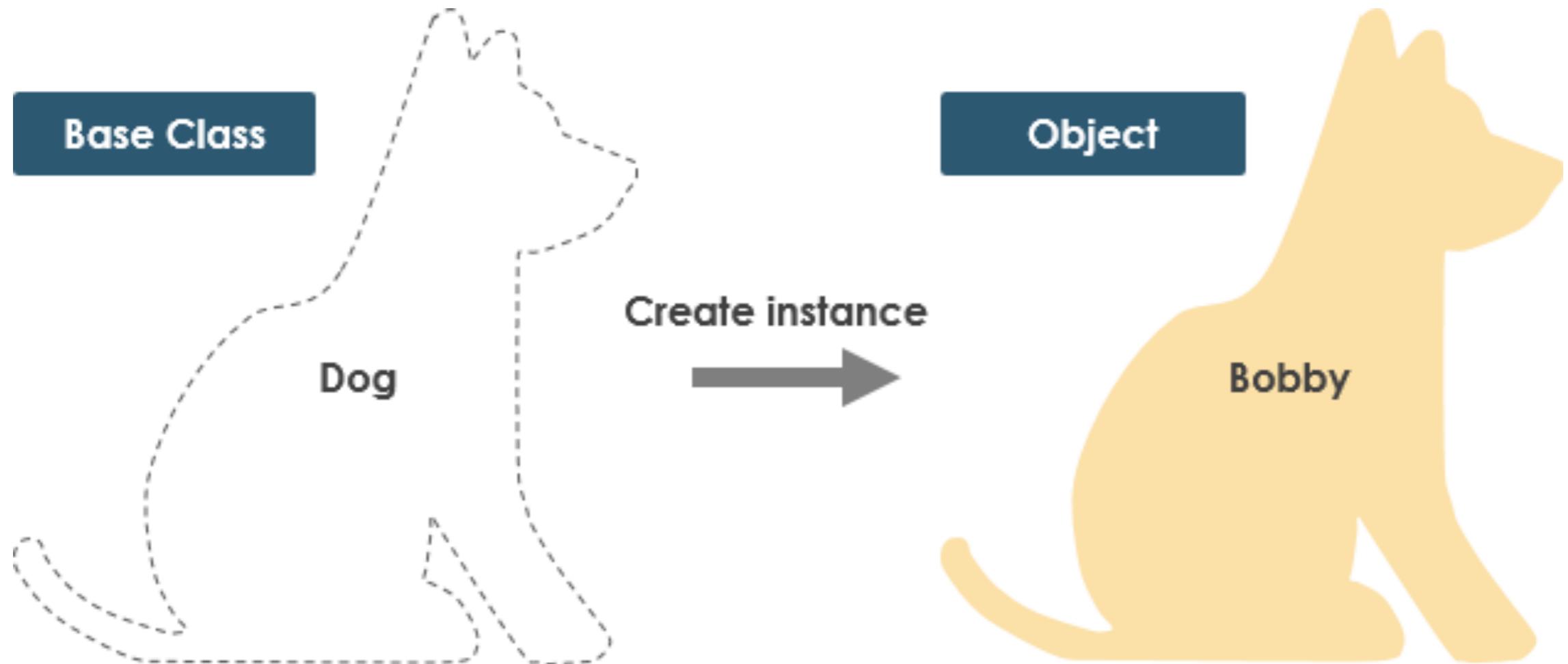
Class, Object



Class, Object



Class, Object


Properties

Color

Eye Color

Height

Length

Weight

Methods

Sit

Lay Down

Shake

Come

Property Values

Color: Yellow

Eye Color: Brown

Height: 17 in

Length: 35 in

Weight: 24 pounds

Methods

Sit

Lay Down

Shake

Come

Create Instance

Objective-C 인스턴스 생성

- `[[ClassName alloc] init];`

Swift 인스턴스 생성

- `ClassName()`

Class, Object

```
class Dog {  
    var color: String = "black"  
    var eyeColor: String = "black"  
    var height: Double = 40.0  
    var weight: Double = 6.0  
  
    func sit() {}  
    func layDown() {}  
    func shake() {}  
}
```

```
let bobby: Dog = Dog()  
bobby.color = "white"  
bobby.sit()  
  
let tory = Dog()  
tory.color = "brown"  
tory.layDown()
```