클래스와 객체지향

코틀린 기초강좌

강사: 배정만

객체 지향 프로그래밍

Object-Oriented Programming(OOP)
프로그램을 수많은 '객체'라는 기본 단위로 나누고
객체의 상호작용으로 서술하는 방식

하나의 '역할'을 수행하는 함수(메소드)와 데이터의 묶음

유래

- 절차적 프로그래밍의 문제점
  - 복잡도의 증가
- '구조적 프로그래밍'을 제안
- 불완전한 구조화의 문제점
  - 전역 네임스페이스 포화
  - 이름 짓기 문제

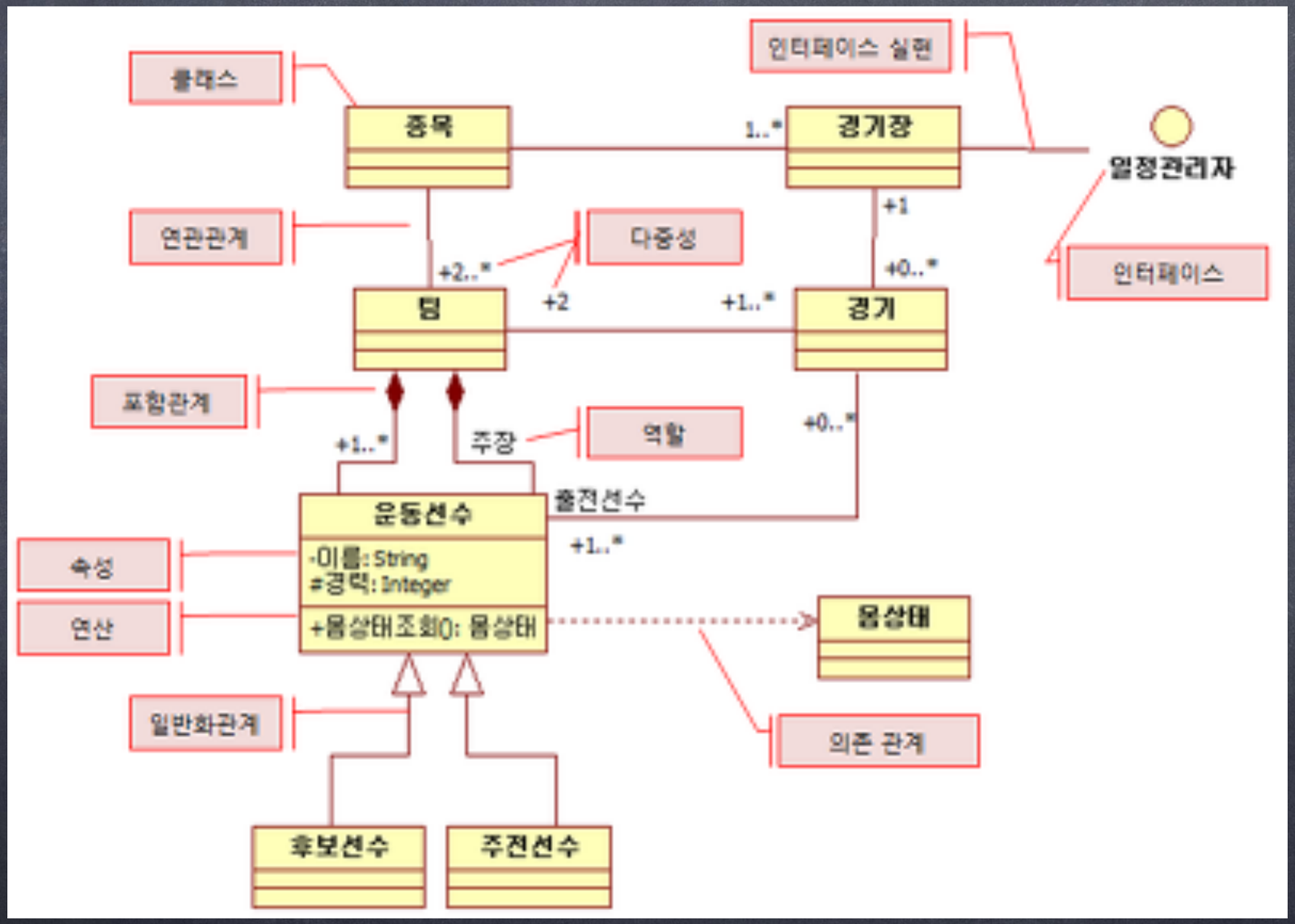
요소

- 캡슐화(Encapsulation)
  - 전역 네임스페이스 문제를 해결하고, 문제를 단순화 함
- 상속(Inheritance)
  - 캡슐화를 유지하면서도 클래스의 재사용 가능
- 다형성(Polymorphism)
  - 하나의 변수명, 함수명 등이 상황에 따라 다른 의미로 해석 될 수 있는 것
  - 오버로딩(Overloading: 중복 정의/다중 정의)

클래스 다이어그램

클래스 다이어그램은 시스템을 구성하는 클래스의 내부 구조
클래스의 속성과 연산 클래스간의 관계를 표현
-> 문제를 해결하기 위한 설계도

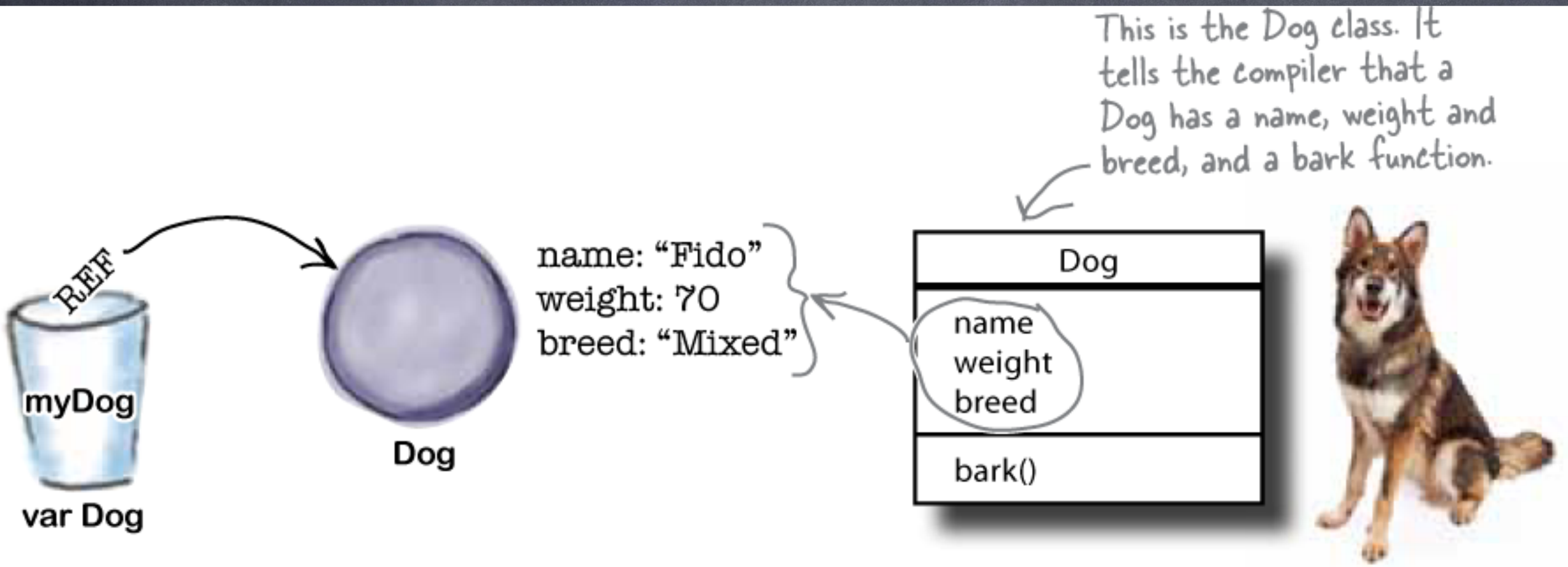
클래스 다이어그램


클래스
종목
1..*
경기장
인터페이스 실현
일정관리자
연관관계
다중성
+2..*
+2
+1
+0..*
+1..*
팀
경기
인터페이스
포함관계
+1..*
주장
역할
+0..*
출전선수
운동선수
+1..*
속성
-이름: String
#경력: Integer
연산
+몸상태조회(): 몸상태
몸상태
일반화관계
의존 관계
후보선수
주전선수

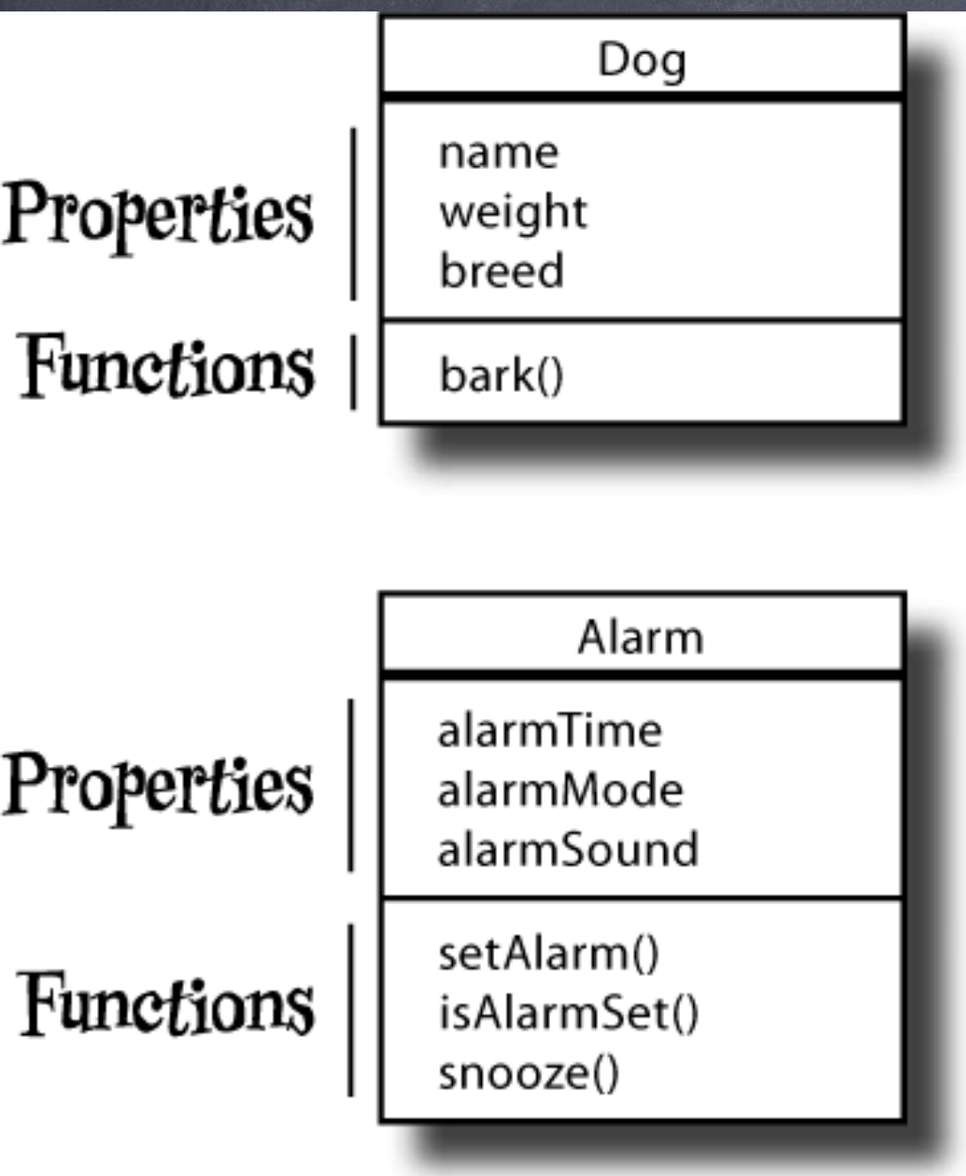
Kotlin 클래스

코틀린에서 클래스 선언과 메모리 할당 과정을 설명

클래스 정의하기


This is the Dog class. It tells the compiler that a Dog has a name, weight and breed, and a bark function.
REF
myDog
var Dog
Dog
name: "Fido"
weight: 70
breed: "Mixed"
Dog
name
weight
breed
bark()

클래스 디자인



- 객체가 자신에 대해 알고있는 것들은 프로퍼티
- 객체가 할 수있는 일은 함수

클래스 객체 생성

```
var myDog = Dog("Fido", 70, "Mixed")
```

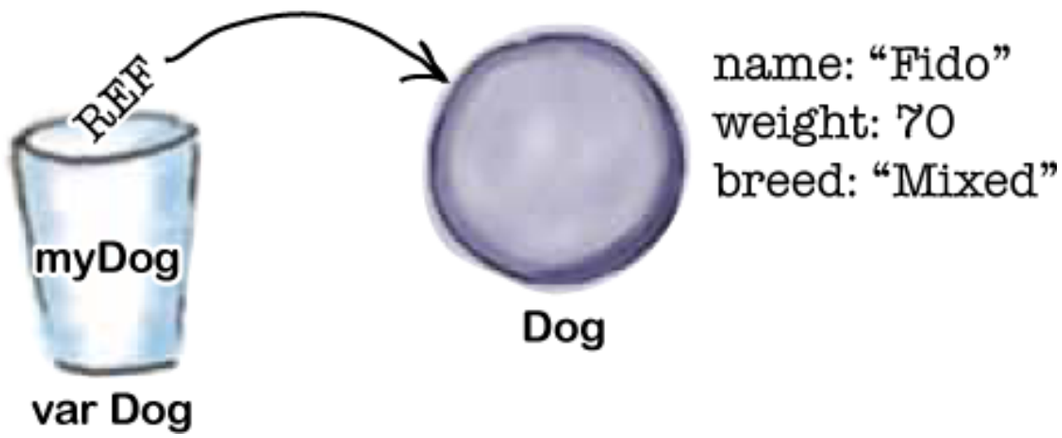
변수 선언

객체 생성

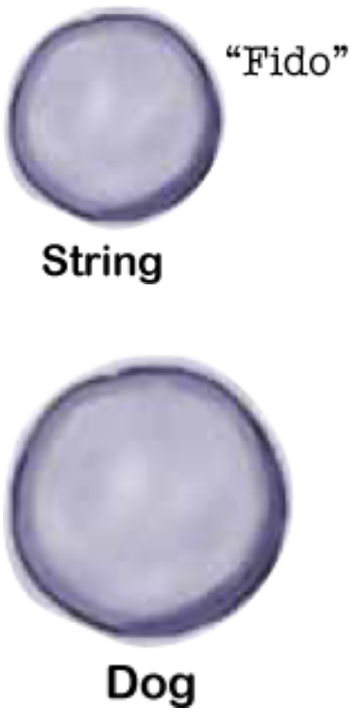
```
var myDog = Dog("Fido", 70, "Mixed")
```

객체 참조 연결

```
var myDog = Dog("Fido", 70, "Mixed")
```



var myDog = Dog ("Fido", 70, "Mixed")

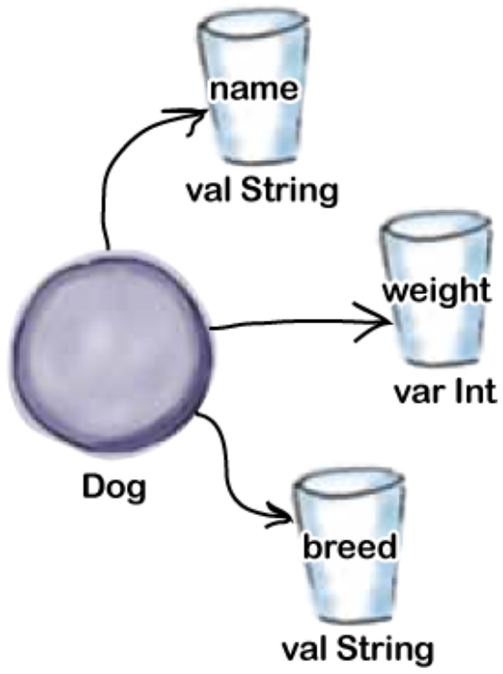


인수들이 먼저 생성됨

객체 공간을 할당하고 생성자 호출

```
class Dog(val name: String,
          var weight: Int,
          val breed: String) {

}
```



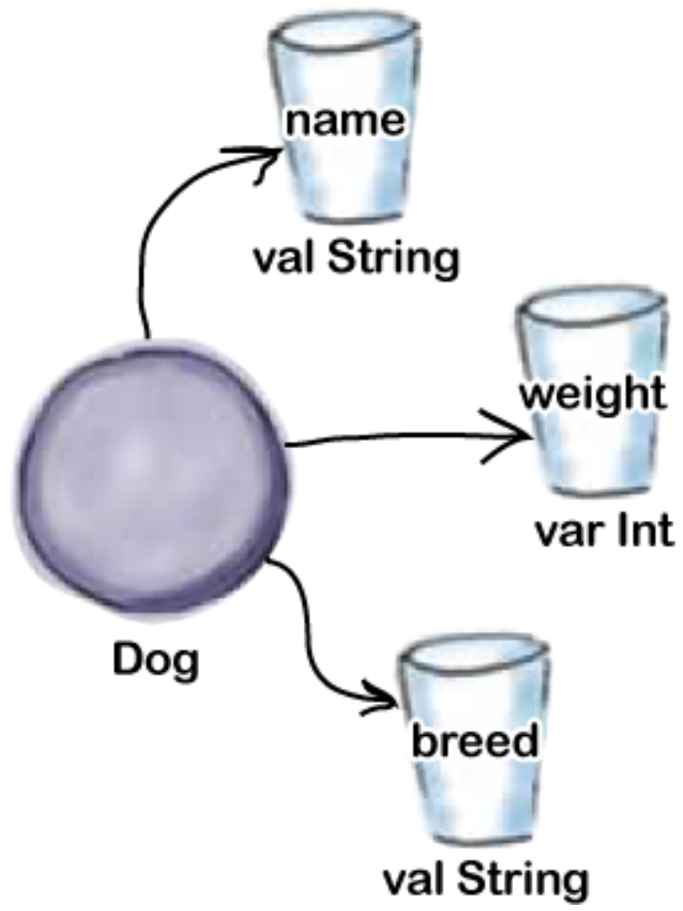
Dog 생성자는 name='fido',
weight = 70, breed = 'Mixed'
라는 세 가지 속성을 정의합니다.

var myDog = Dog ("Fido", 70, "Mixed")

Dog 생성자는 name='Fido', weight = 70, breed = 'Mixed'
라는 세 가지 속성을 정의합니다.

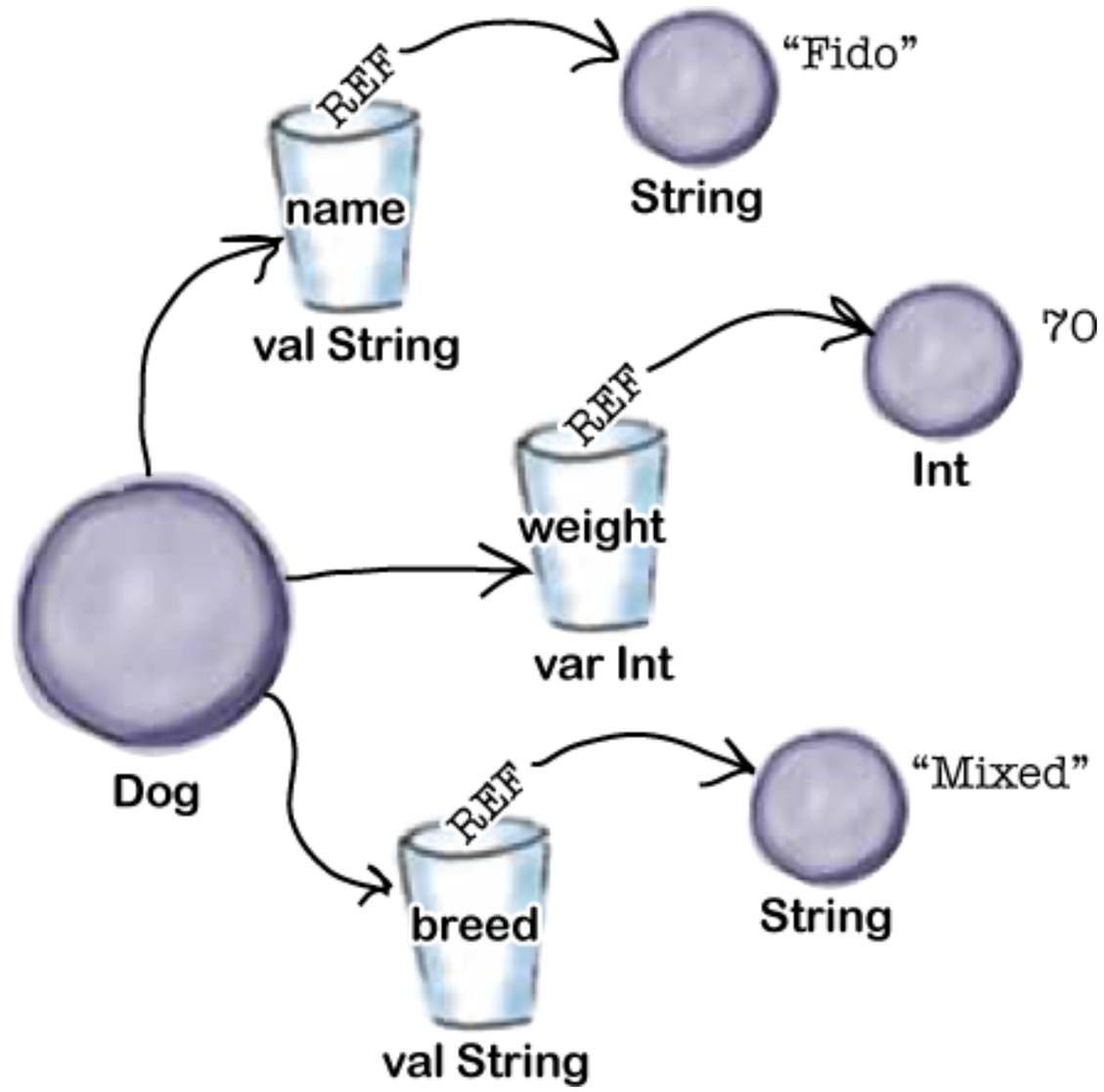
```
class Dog(val name: String,
          var weight: Int,
          val breed: String) {

}
```



var myDog = Dog ("Fido", 70, "Mixed")

각 프로퍼티에 값 객체 참조가 지정됨



var myDog = Dog ("Fido", 70, "Mixed")

Dog 객체에 대한 참조가
myDog라는 새 Dog 변수에 지정됨