Objects Inheritance

Why?

Why?

- code re-use

Inheritance

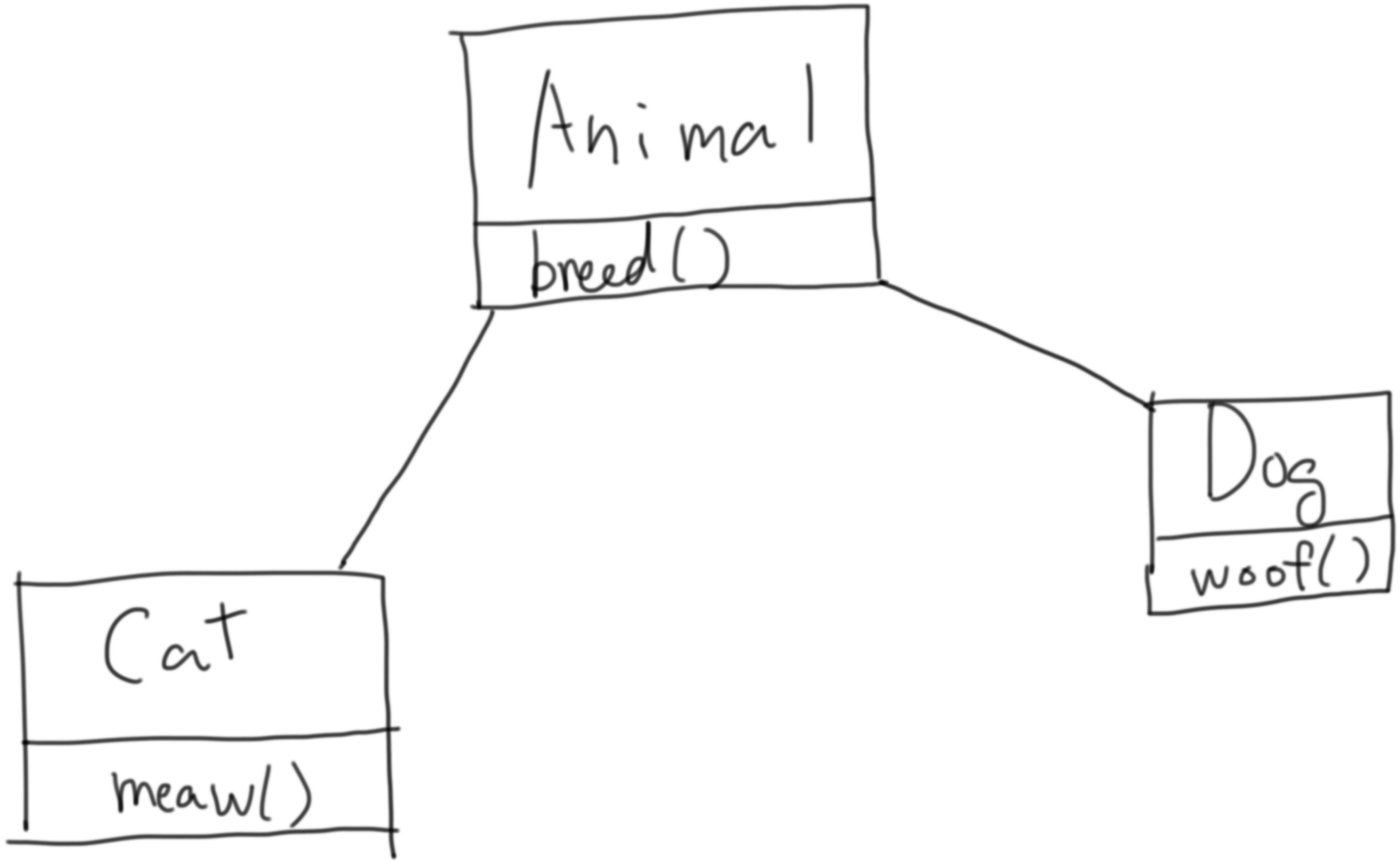
- A class can inherit another class
- The inheriting class is called the subclass
- The inherited class is called the superclass
- The subclass inherits all methods of the superclass

Example

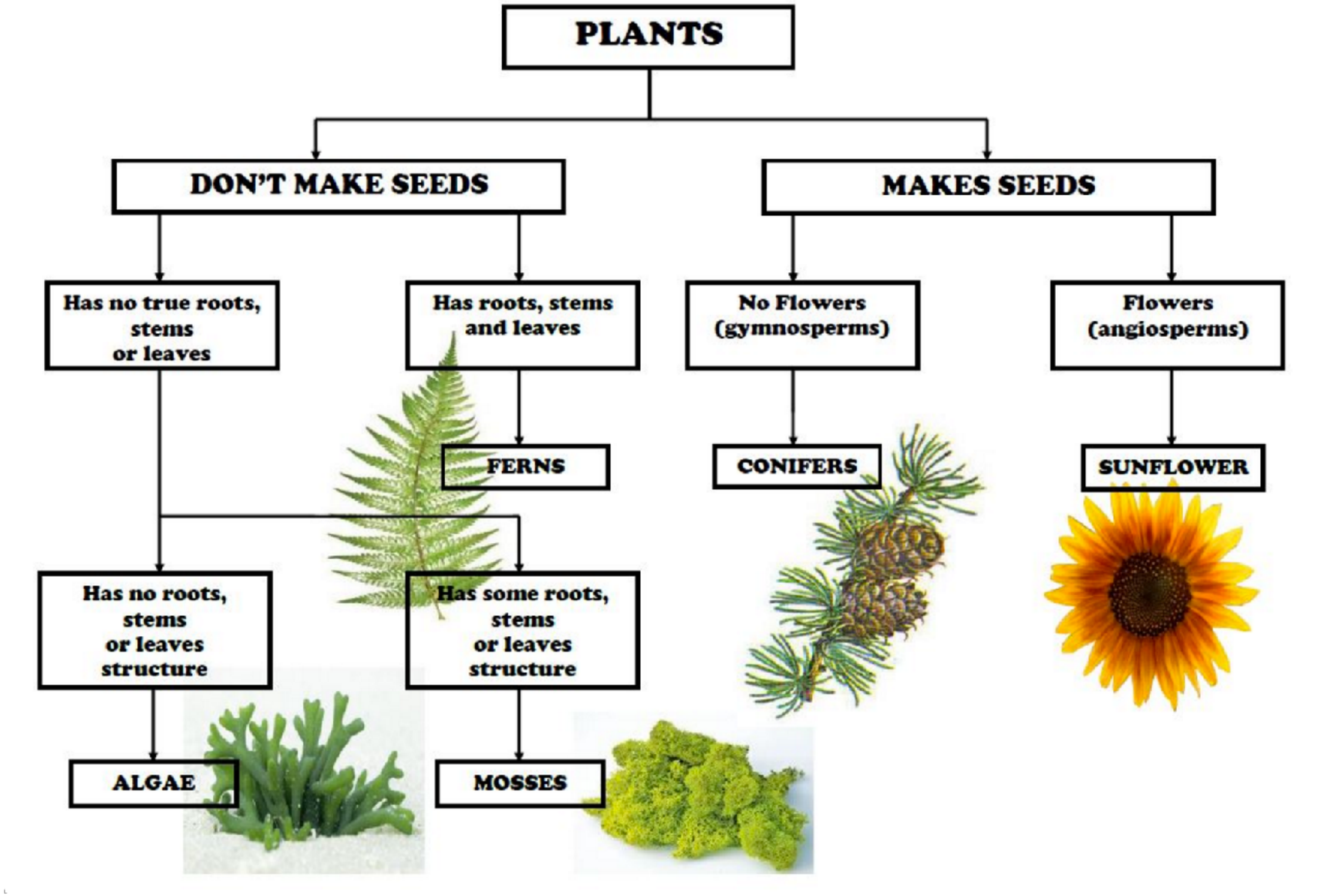
```
class Animal(object):
    def breed(self):
        return Animal()
```

```
class Dog(Animal):
    def woof(self):
        print 'Woof!'
```

```
class Cat(Animal):
    def meow(self):
        print 'Meow!'
```


Animal
breed()
Cat
meaw()
Dog
woof()

Plant Taxonomy



Inheritance Syntax

Cat inherits Animal
Cat is a type of Animal
Cat is-a Animal

Inheriting methods

```
my_cat = Cat()
my_cat.meow()
kitten = my_cat.breed()
```

An instance of `Cat` inherits the `breed` method of `Animal`

Inheriting methods

```
my_dog = Dog()
my_dog.woof()
doggie = my_dog.breed()
```

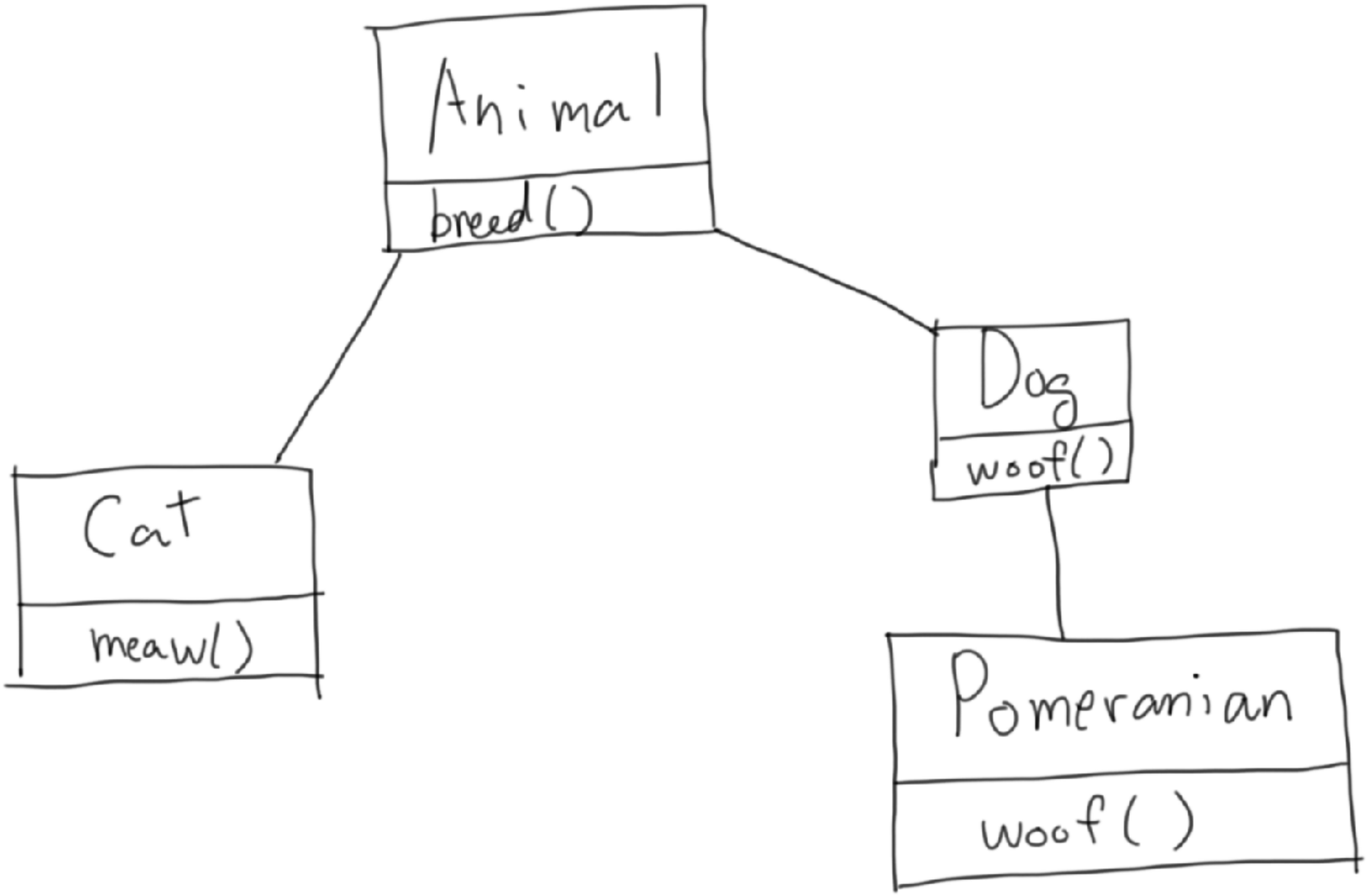
So does **Dog**

Multiple-Level Inheritance

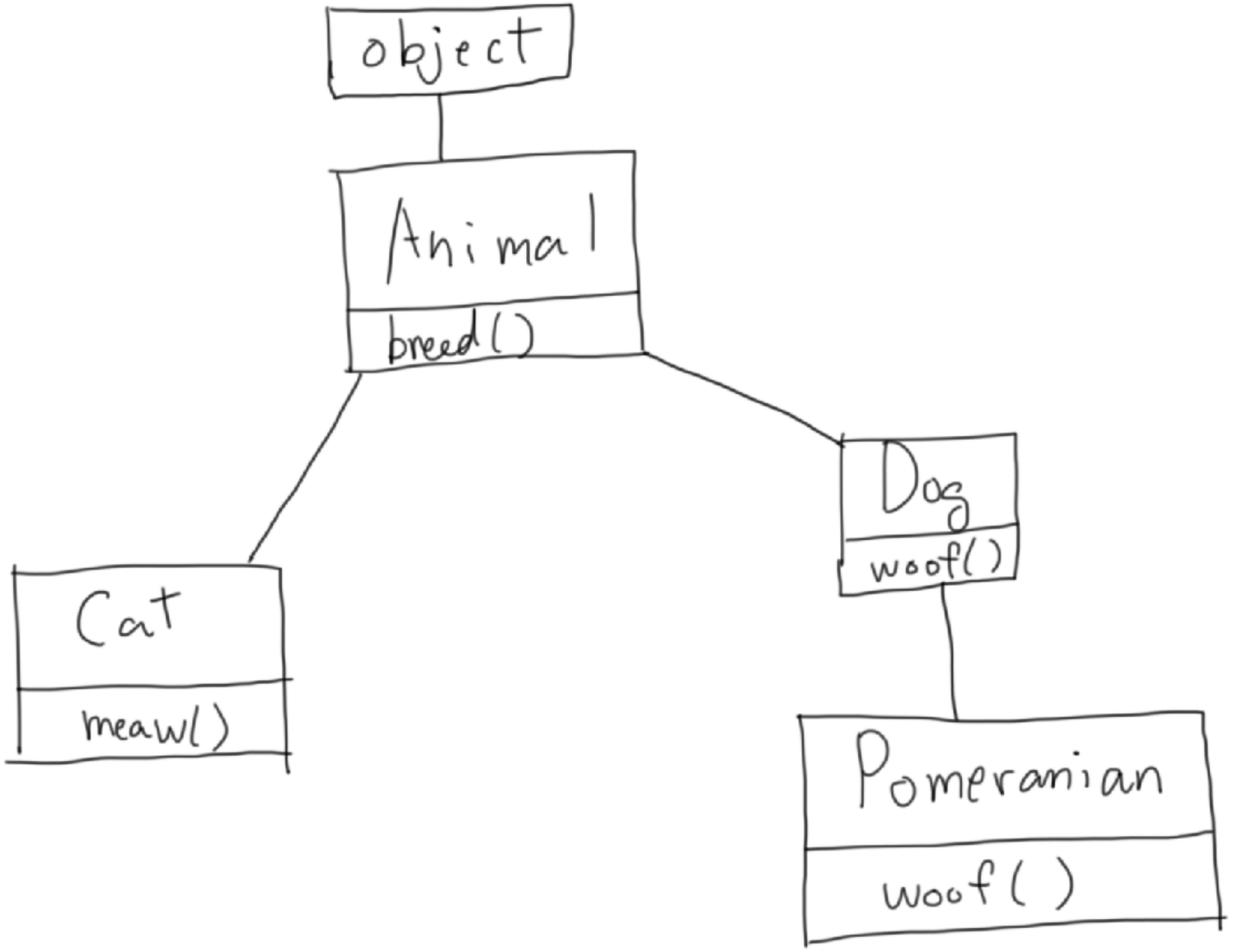
```
class Pomeranian(Dog):
    def woof(self):
        print 'WOOOOOOOOF!'
```

`Pomeranian` inherits `Dog`

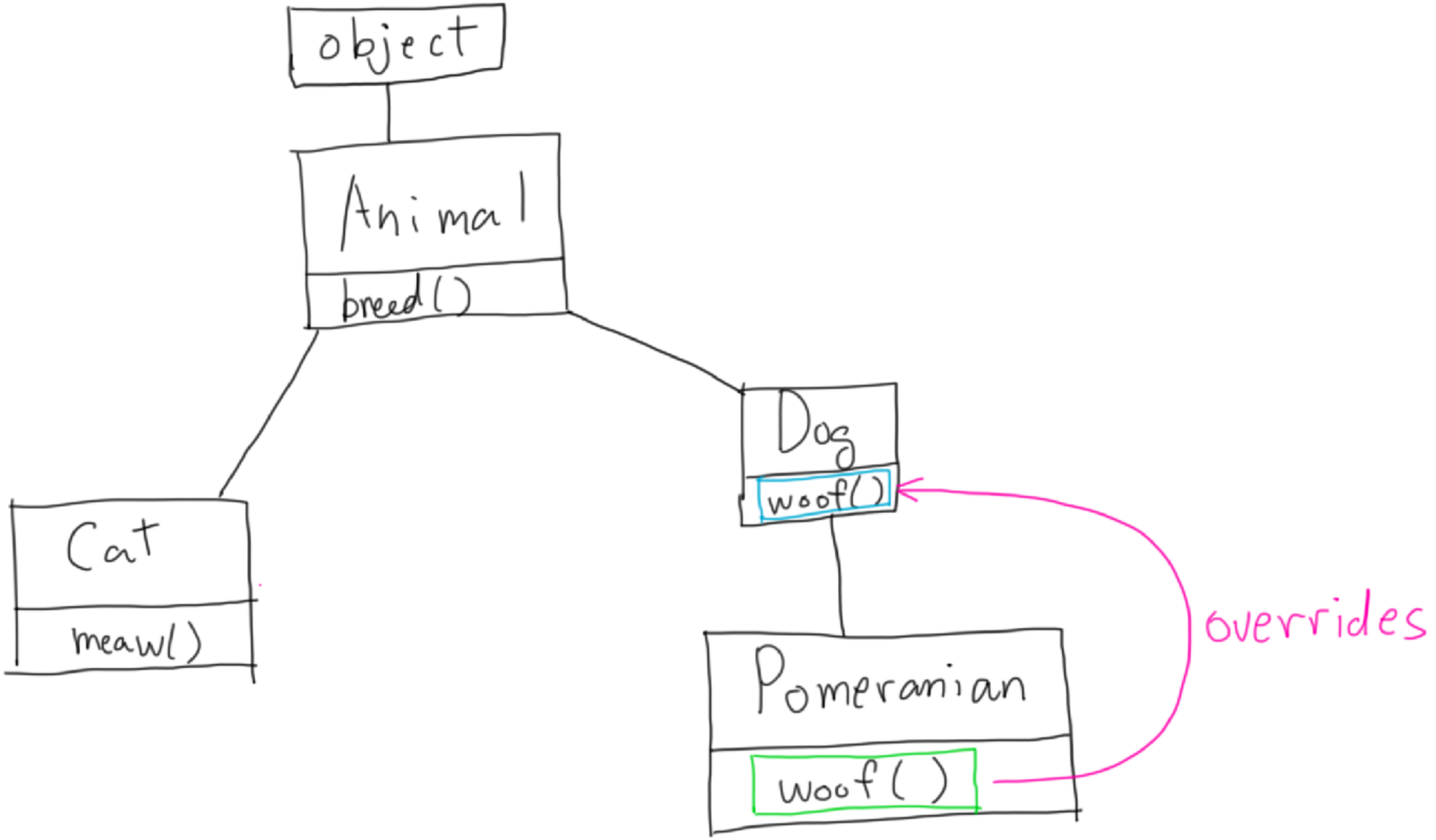
Multiple-Level Inheritance



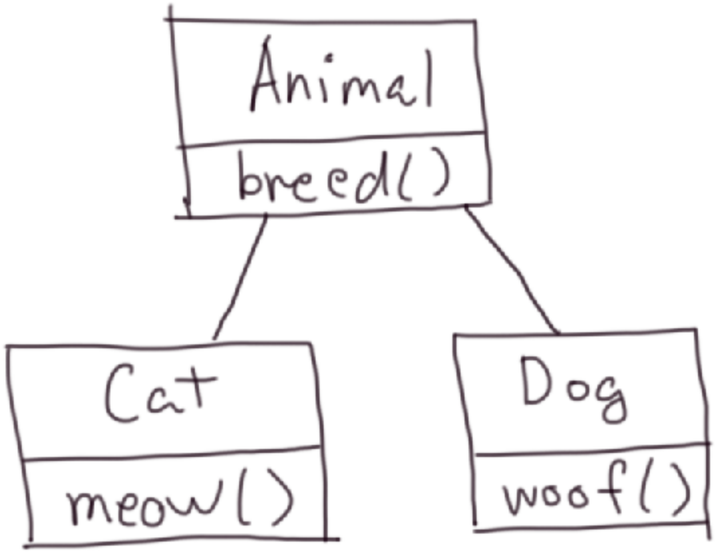
Multiple-Level Inheritance

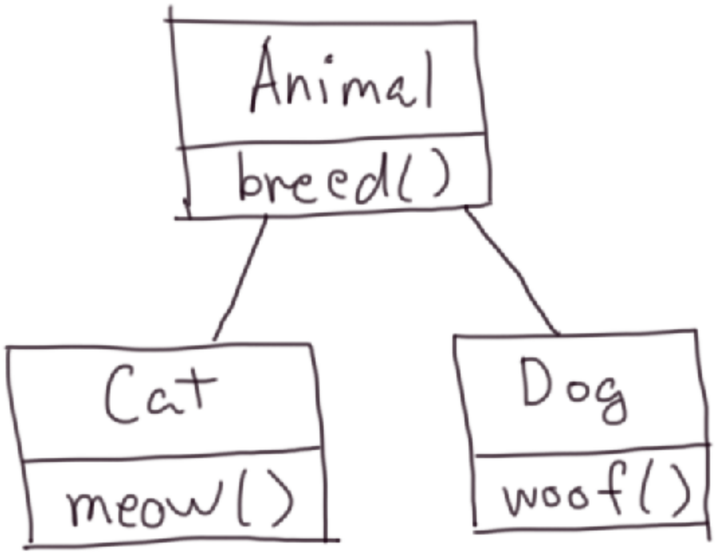


Method Overriding



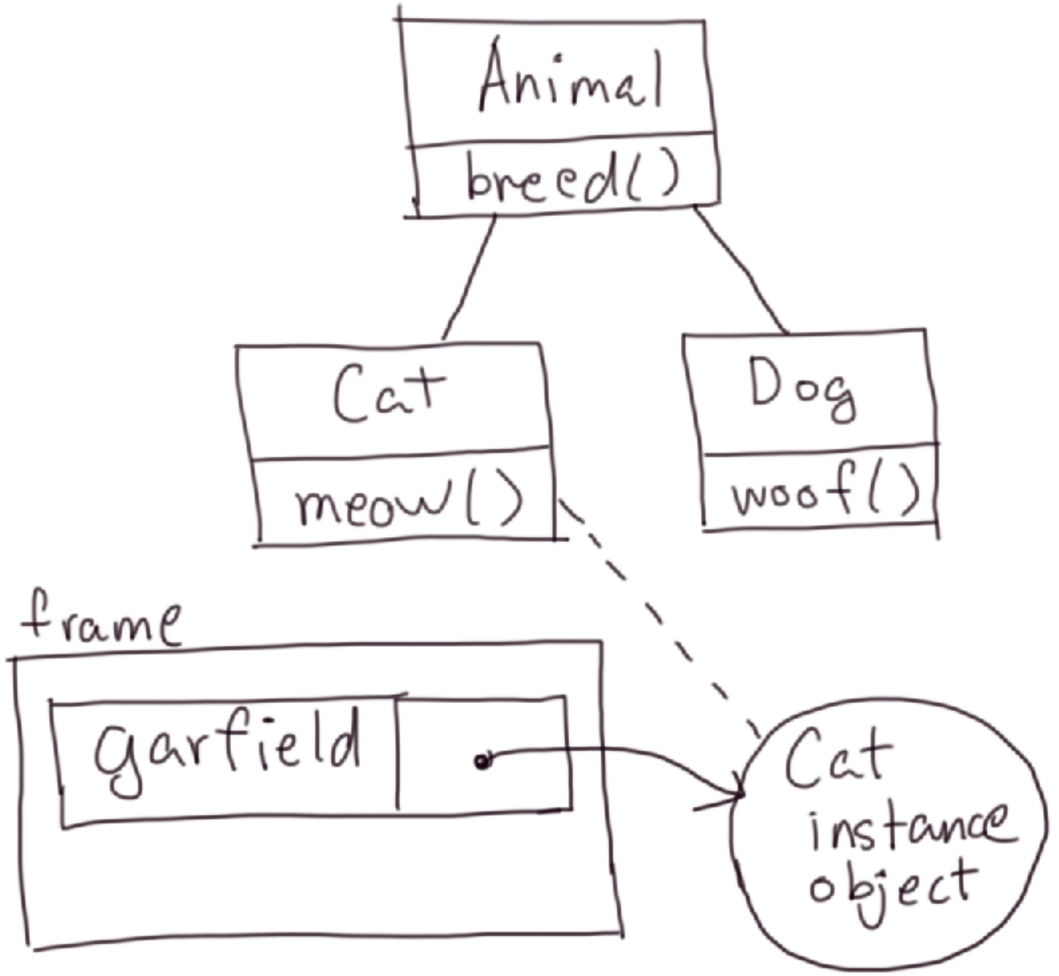
Method Lookup


Animal
breed()
Cat
meow()
Dog
woof()

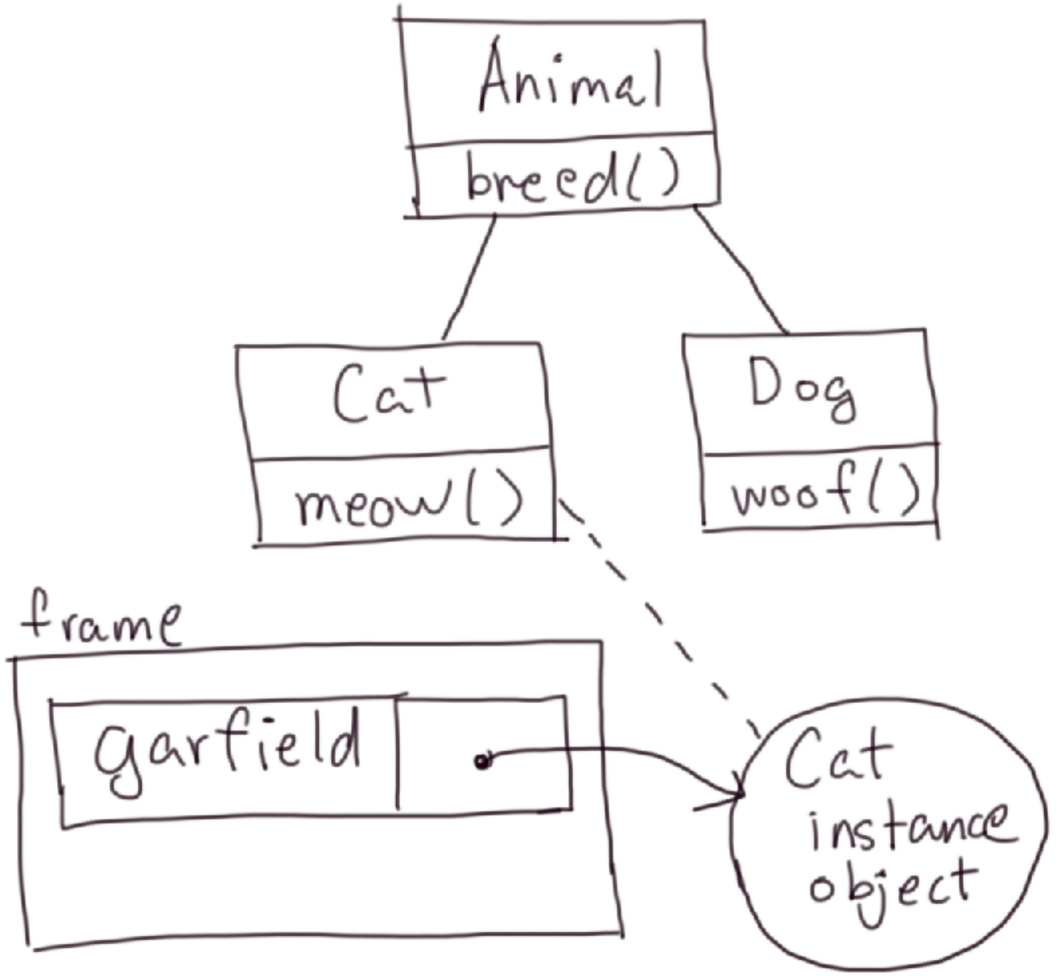


garfield = Cat()

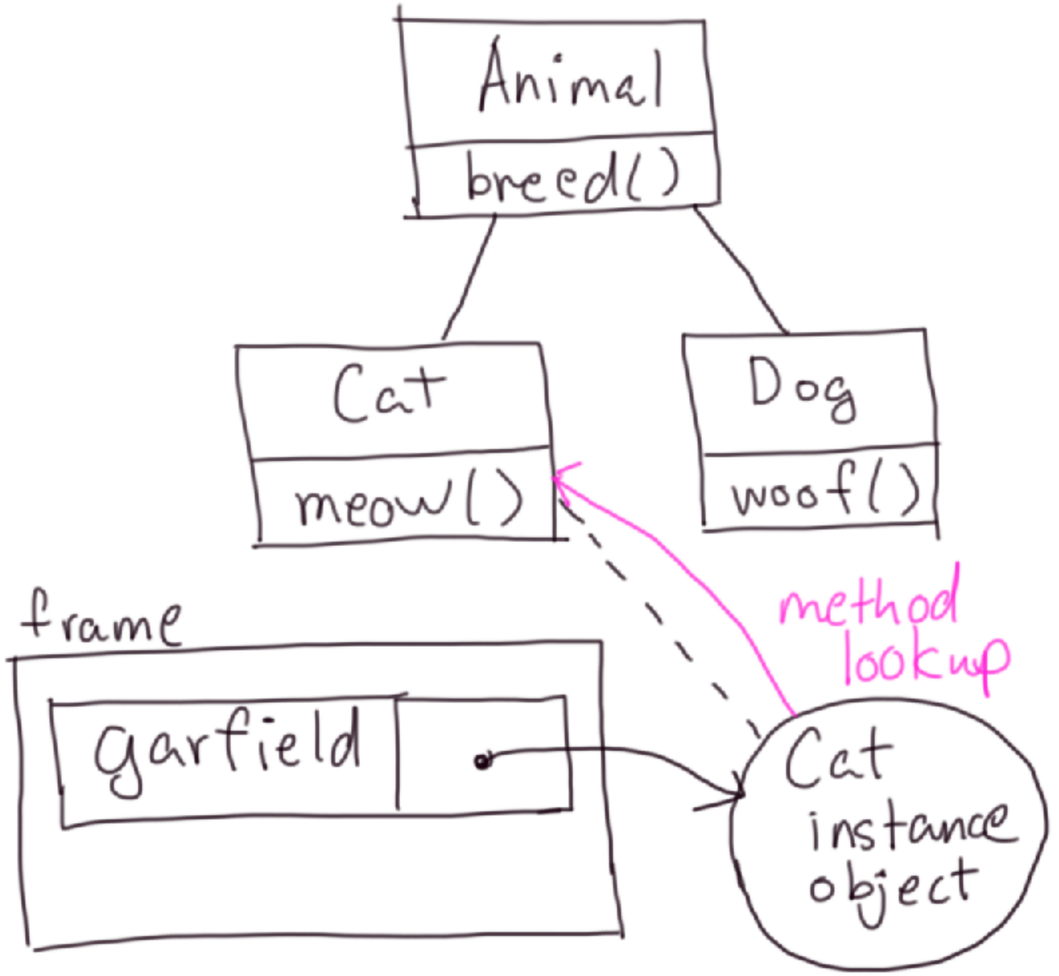
garfield = Cat()



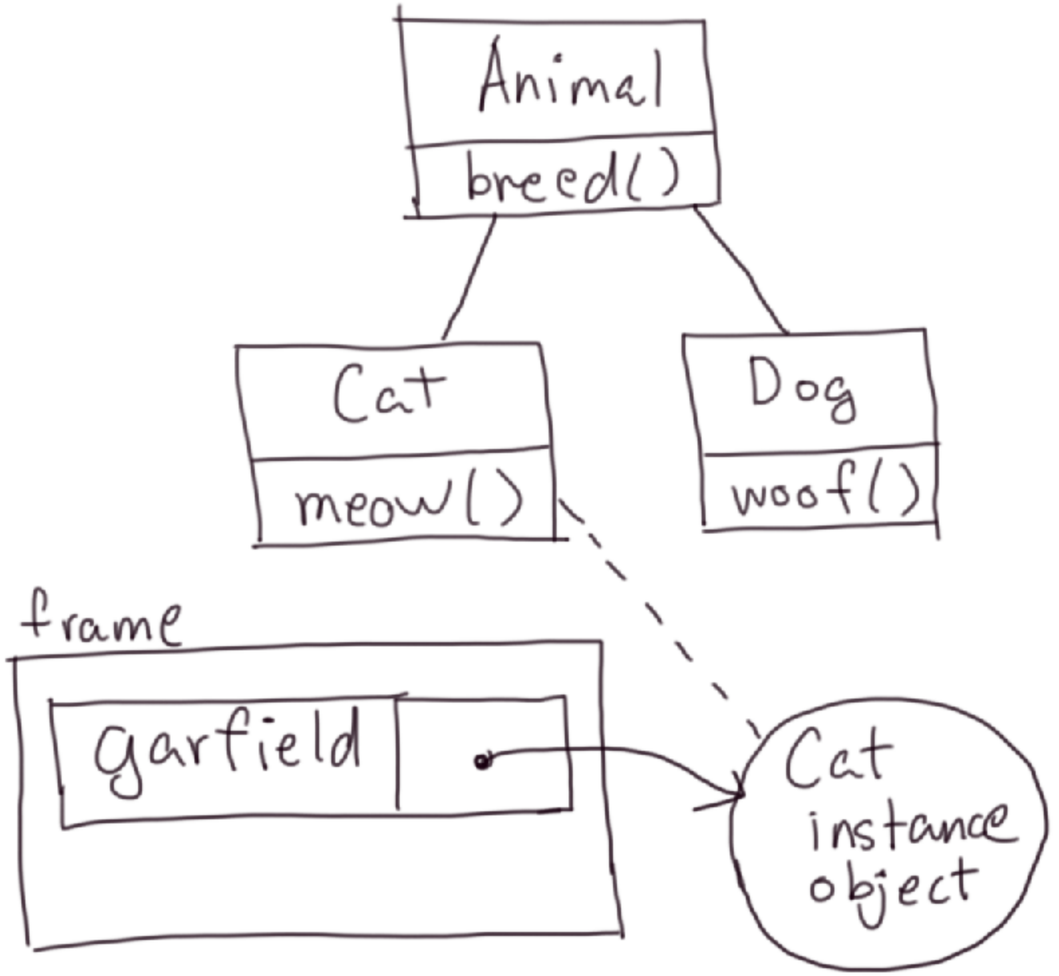
```
garfield = Cat()
garfield.meow()
```



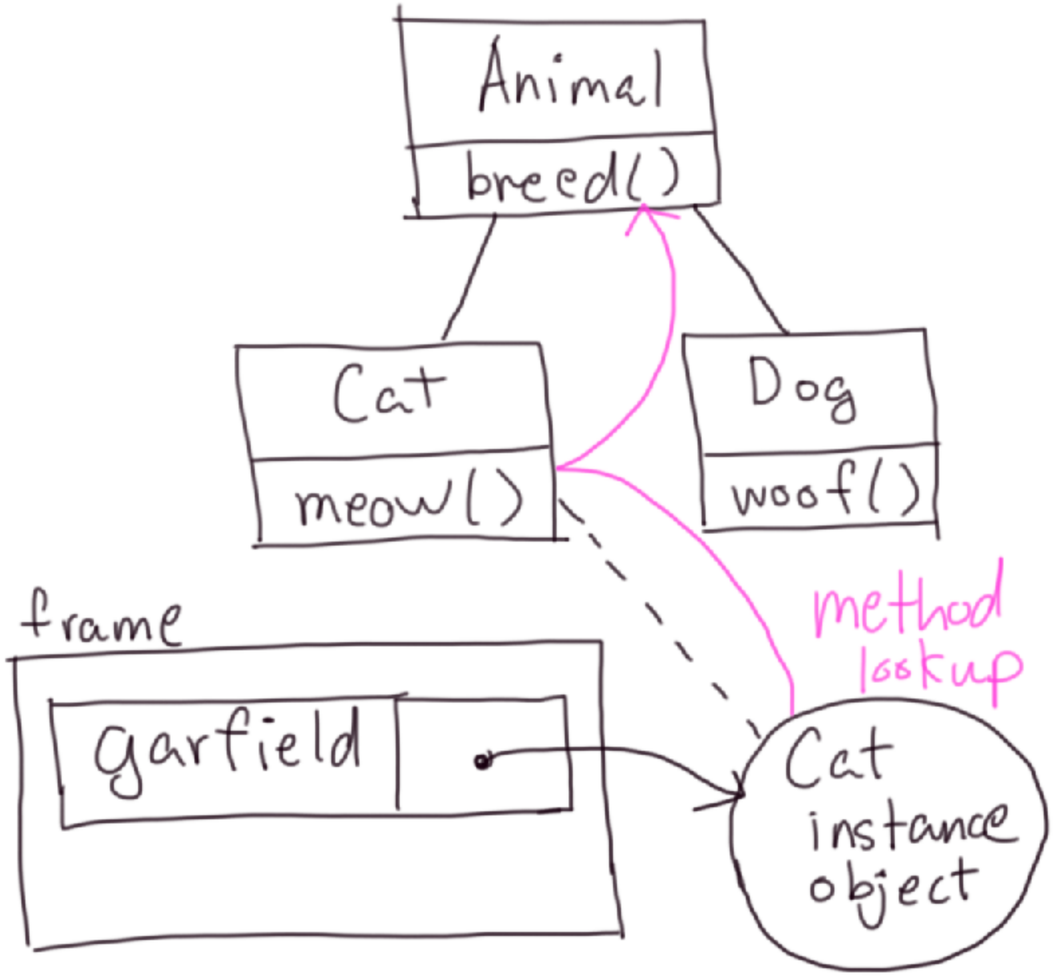
```
garfield = Cat()
garfield.meow()
```

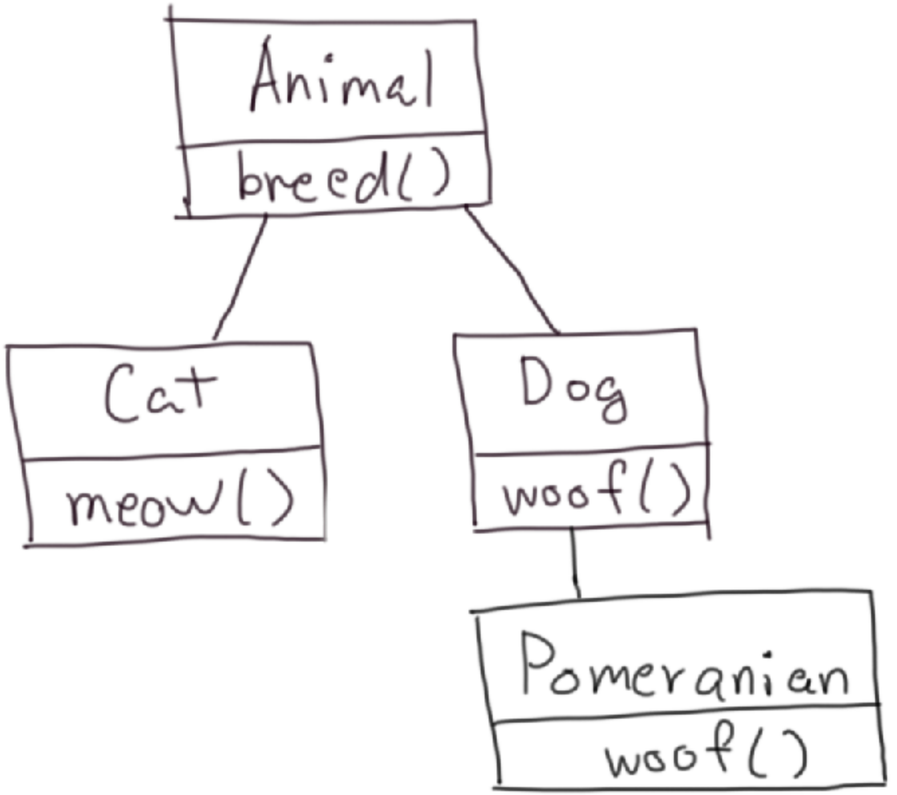


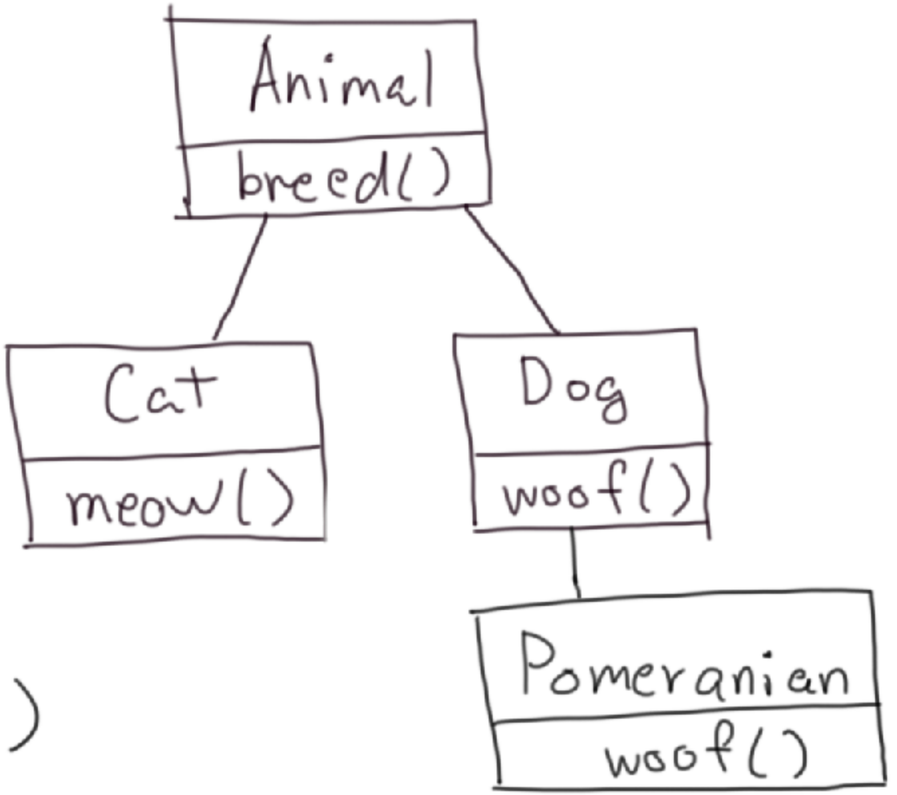
```
garfield = Cat()
garfield.meow()
garfield.breed()
```



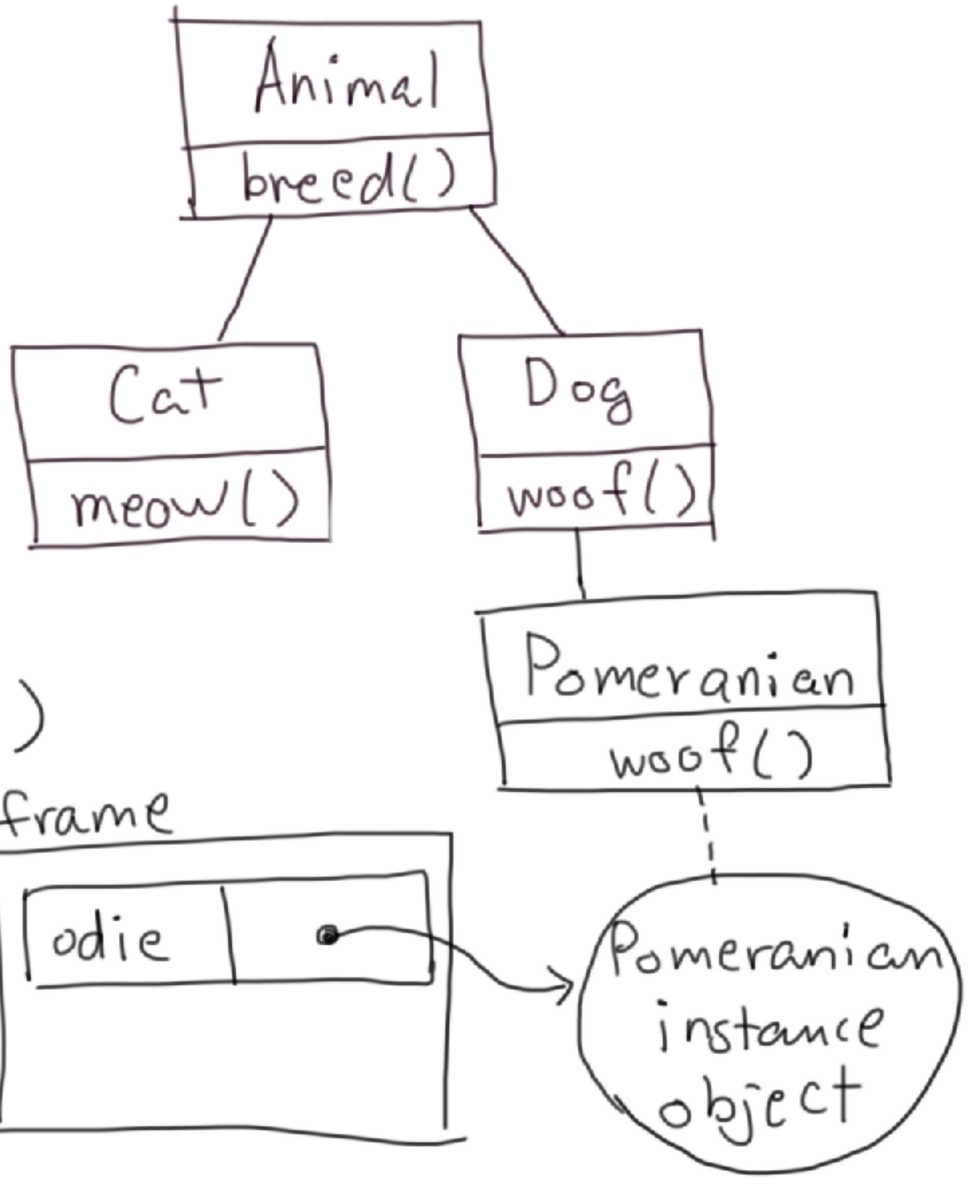
```
garfield = Cat()
garfield.meow()
garfield.breed()
```

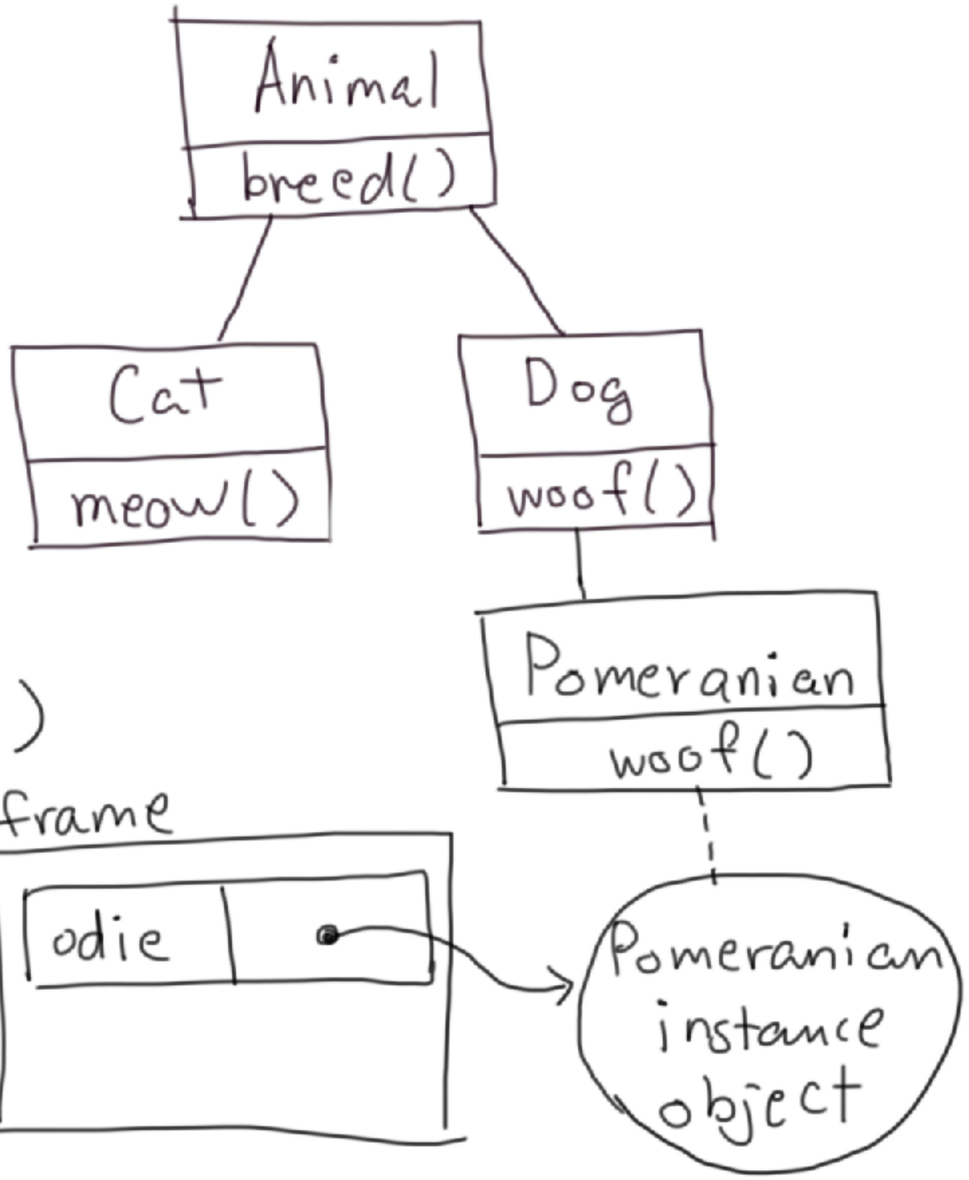


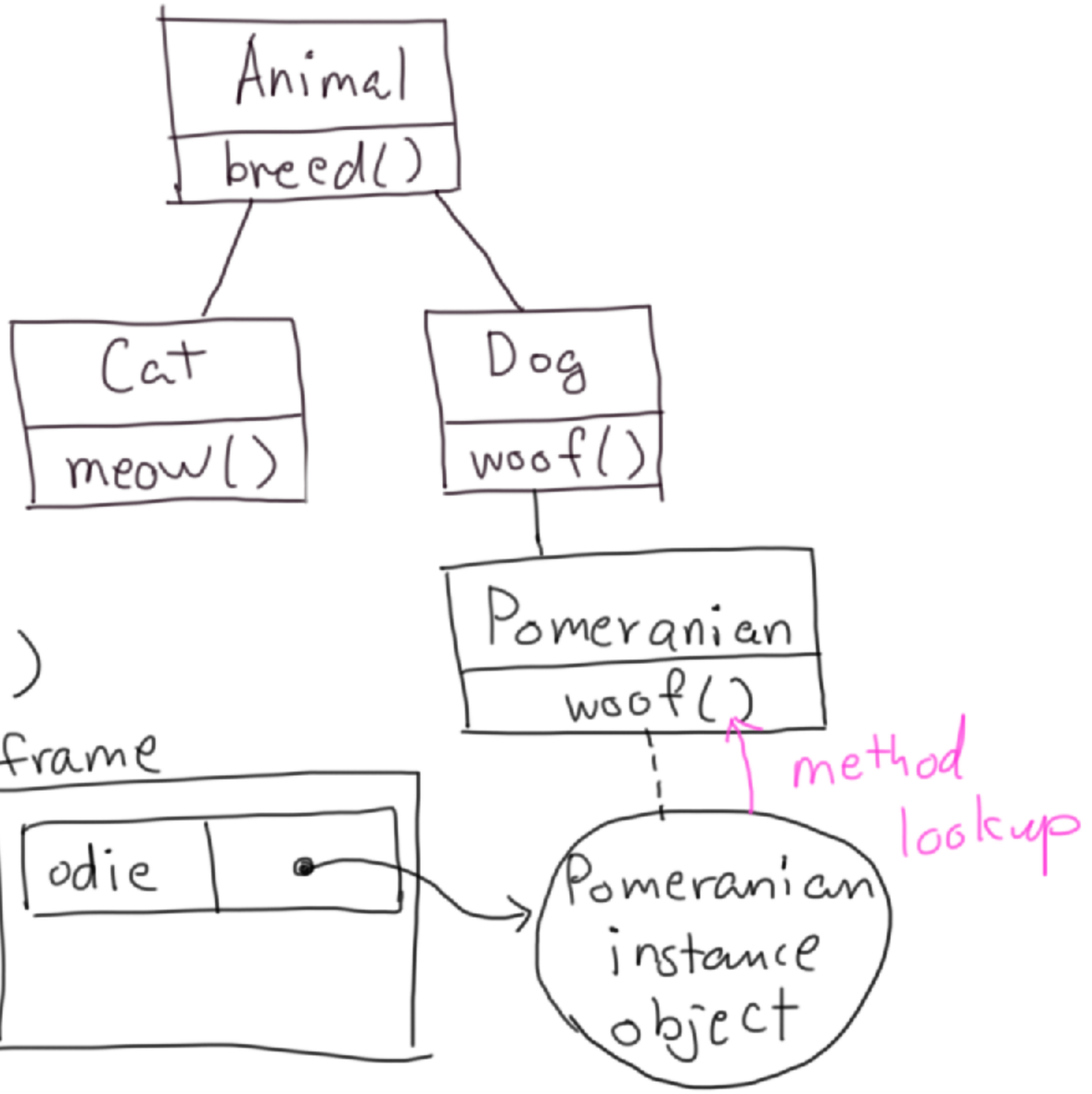

Animal
breed()
Cat
meow()
Dog
woof()
Pomeranian
woof()



odie = Pomeranian()


Animal
breed()
Cat
meow()
Dog
woof()
Pomeranian
woof()
odie = Pomeranian()
frame
odie
Pomeranian instance object


Animal
breed()
Cat
meow()
Dog
woof()
Pomeranian
woof()
odie = Pomeranian()
odie.woof()
frame
odie
Pomeranian instance object


Animal
breed()
Cat
meow()
Dog
woof()
Pomeranian
woof()
method lookup
odie = Pomeranian()
odie.woof()
frame
odie
Pomeranian instance object

More Method Overriding

Calling a super class's method of the same name

```
class Dog(Animal):
    def woof(self):
        print 'Woof!'


class EagerDog(Dog):
    def woof(self):
        Dog.woof(self)
        Dog.woof(self)
```

`EagerDog`'s `woof` overrides `Dog`'s but it also calls `Dog`'s `woof` internally

Calling a super class's method of the same name

```
class ExplicitDog(Dog):
    def woof(self):
        print 'Preparing to bark...'
        super(ExplicitDog, self).woof()
        print 'Barking complete.'
```

super is used to called a method
of the same name on the super class

Using the `super` function

```
class EagerDog(Dog):
    def woof(self):
        super(EagerDog, self).woof()
        super(EagerDog, self).woof()


class ExplicitDog(Dog):
    def woof(self):
        print 'Preparing to bark...'
        super(ExplicitDog, self).woof()
        print 'Barking complete.'
```

super: WTF?

```
class ExplicitDog(Dog):
    def woof(self):
        print 'Preparing to bark...'
        super(ExplicitDog, self).woof()
        print 'Barking complete.'
```

`super` is a function that takes 2 arguments:

1. class of the object in question
2. the object in question itself

and it returns a “super” object which represents a version of the object as its super type, in this case a Dog