Ruby Object Model

敏捷中国大会

Dave Thomas
The Pragmatic Programmers
http://pragprog.com

Simula

Object

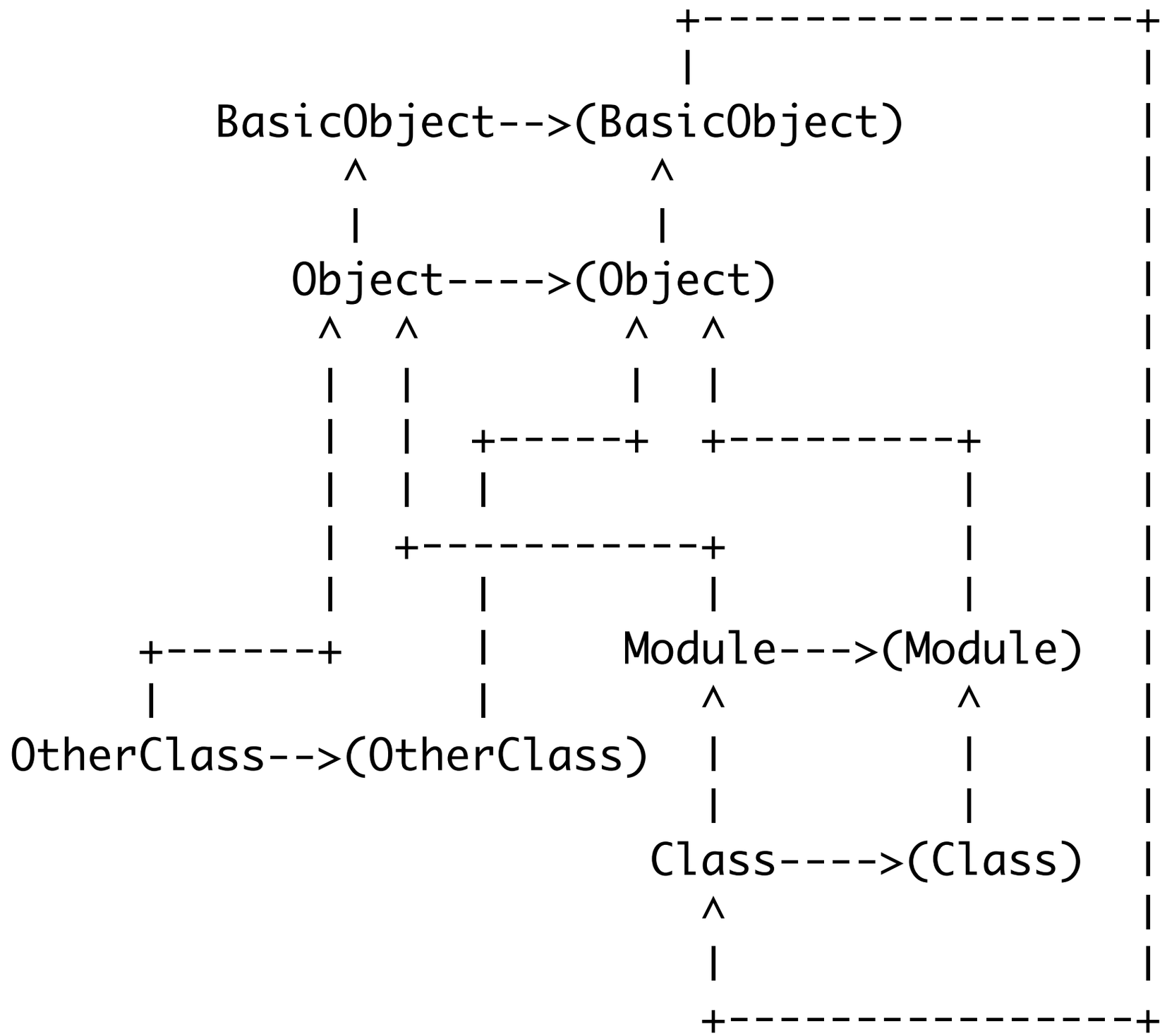
state + behaviour

Object

state + behaviour

Ruby

instance variables + methods


BasicObject
(BasicObject)
Object
(Object)
Module
(Module)
OtherClass
(OtherClass)
Class
(Class)

self

- Current object

Translation can go here...

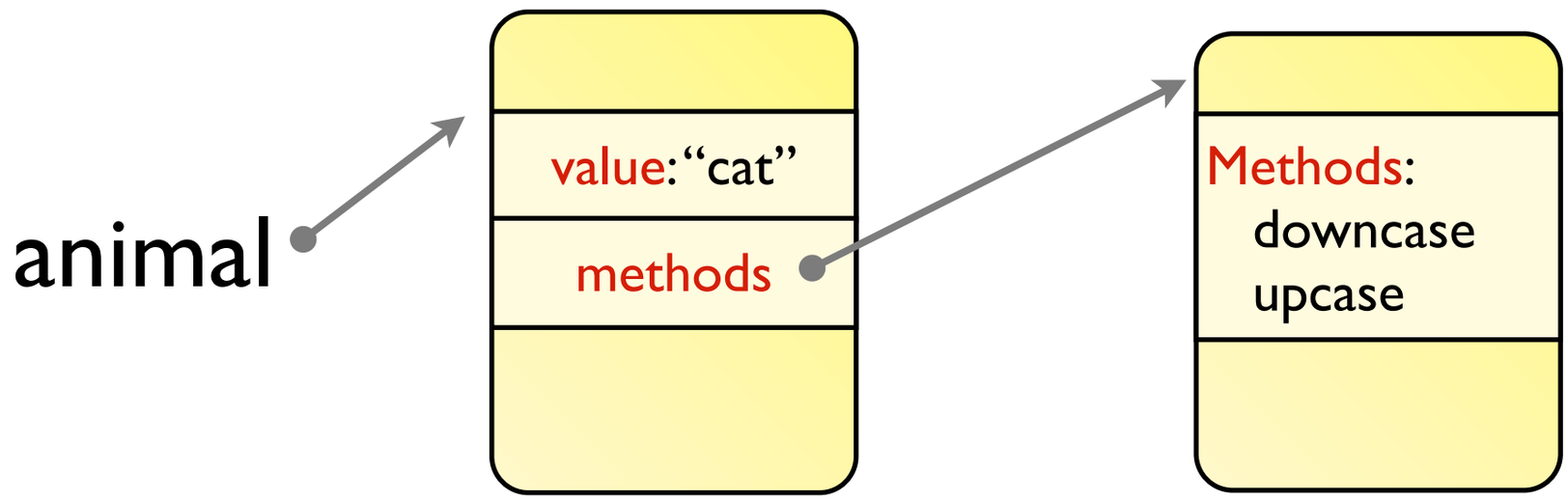
- where instance variables are found
- default receiver for method calls

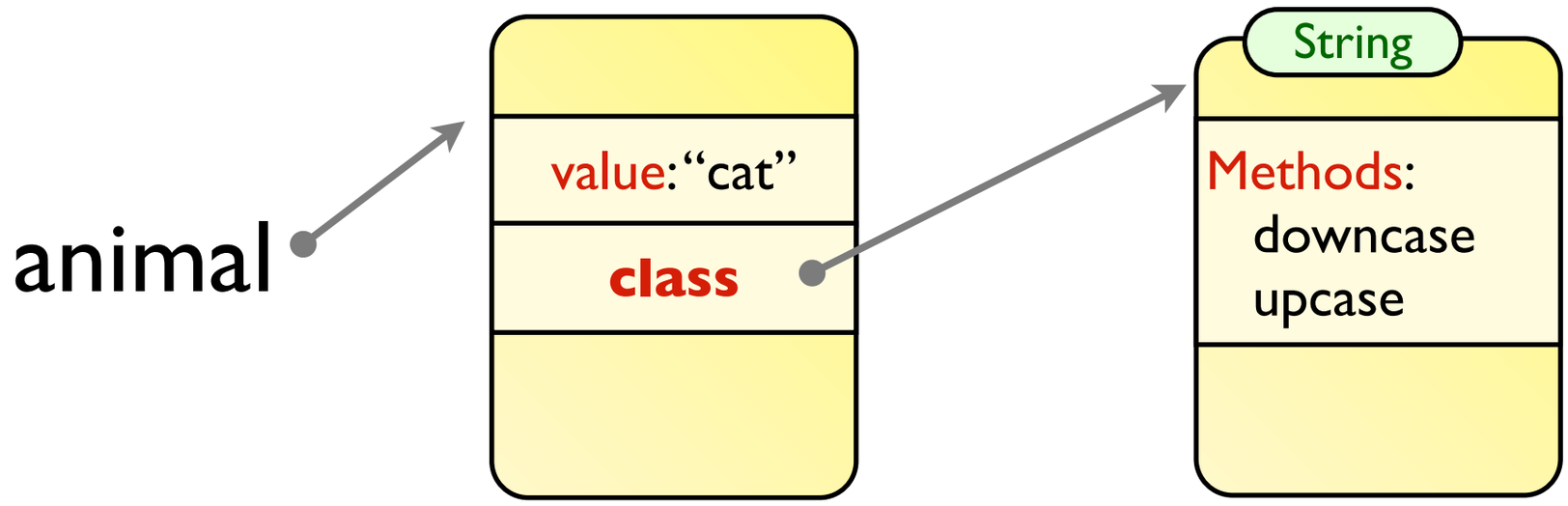
self

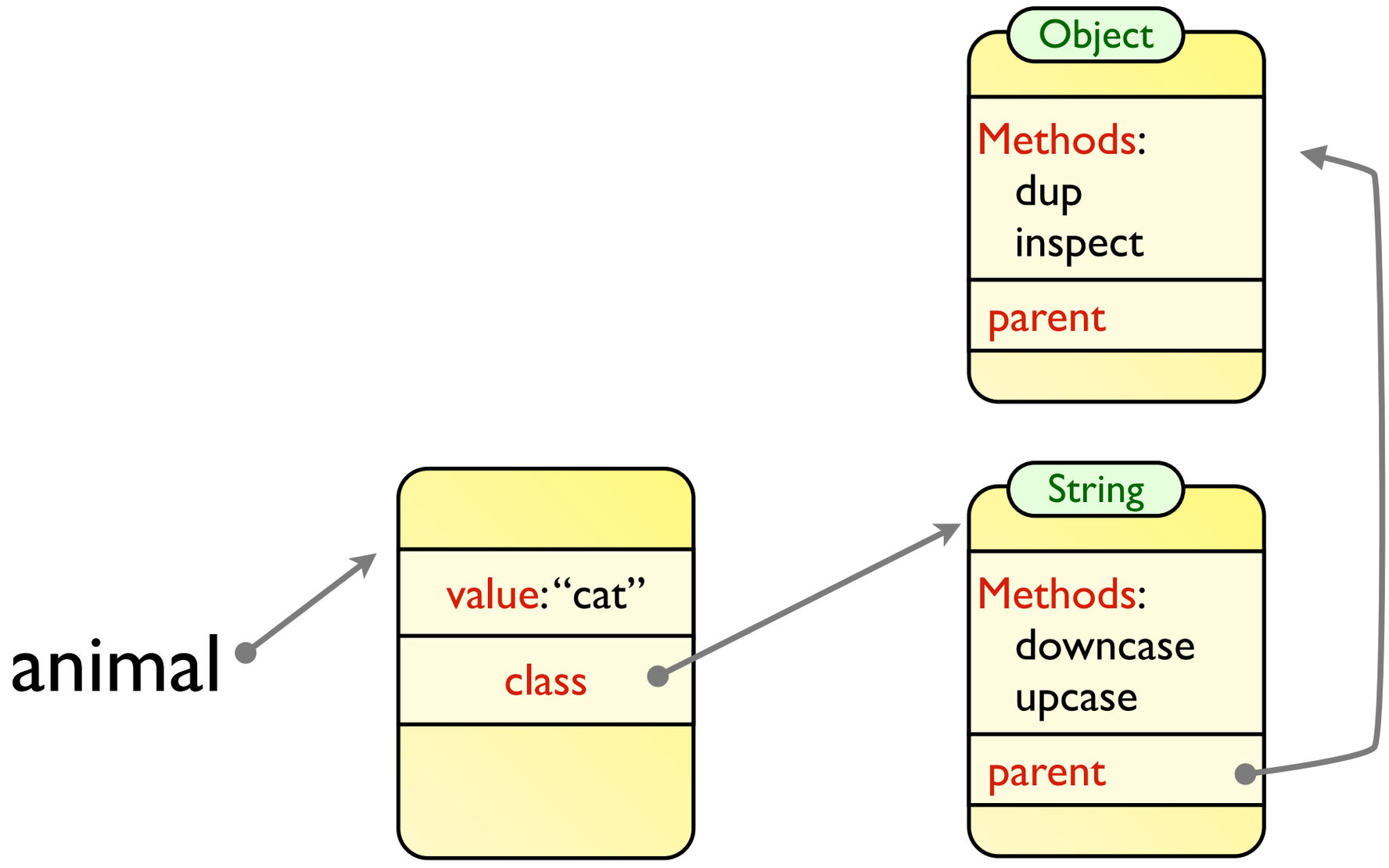
- Only two things can change self

- method call

- class/module definition


animal
value: “cat”
methods
Methods:
downcase
upcase


animal
value: “cat”
class
String
Methods:
downcase
upcase

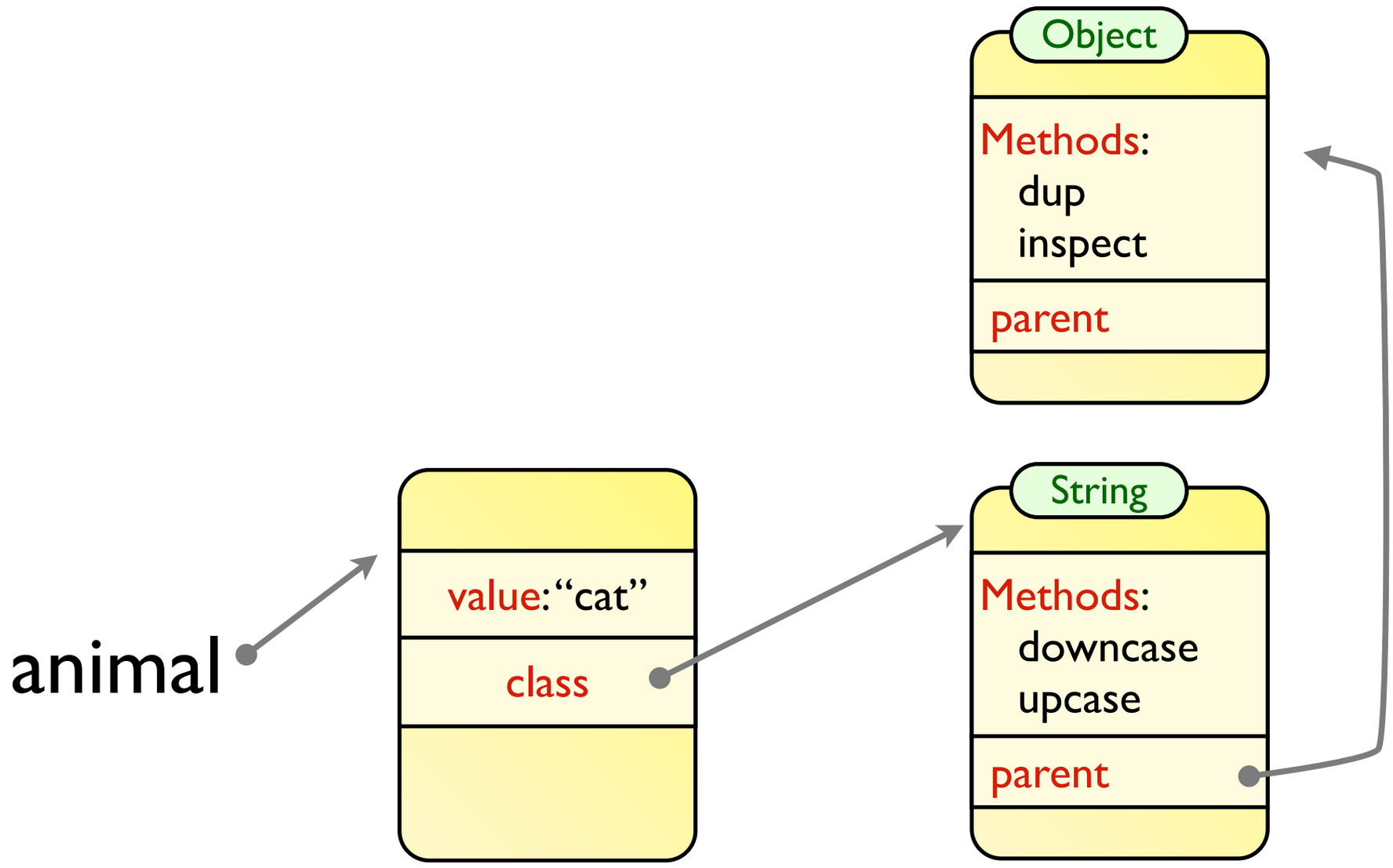

Object
Methods:
dup
inspect
parent
String
Methods:
downcase
upcase
parent
value:“cat”
class
animal

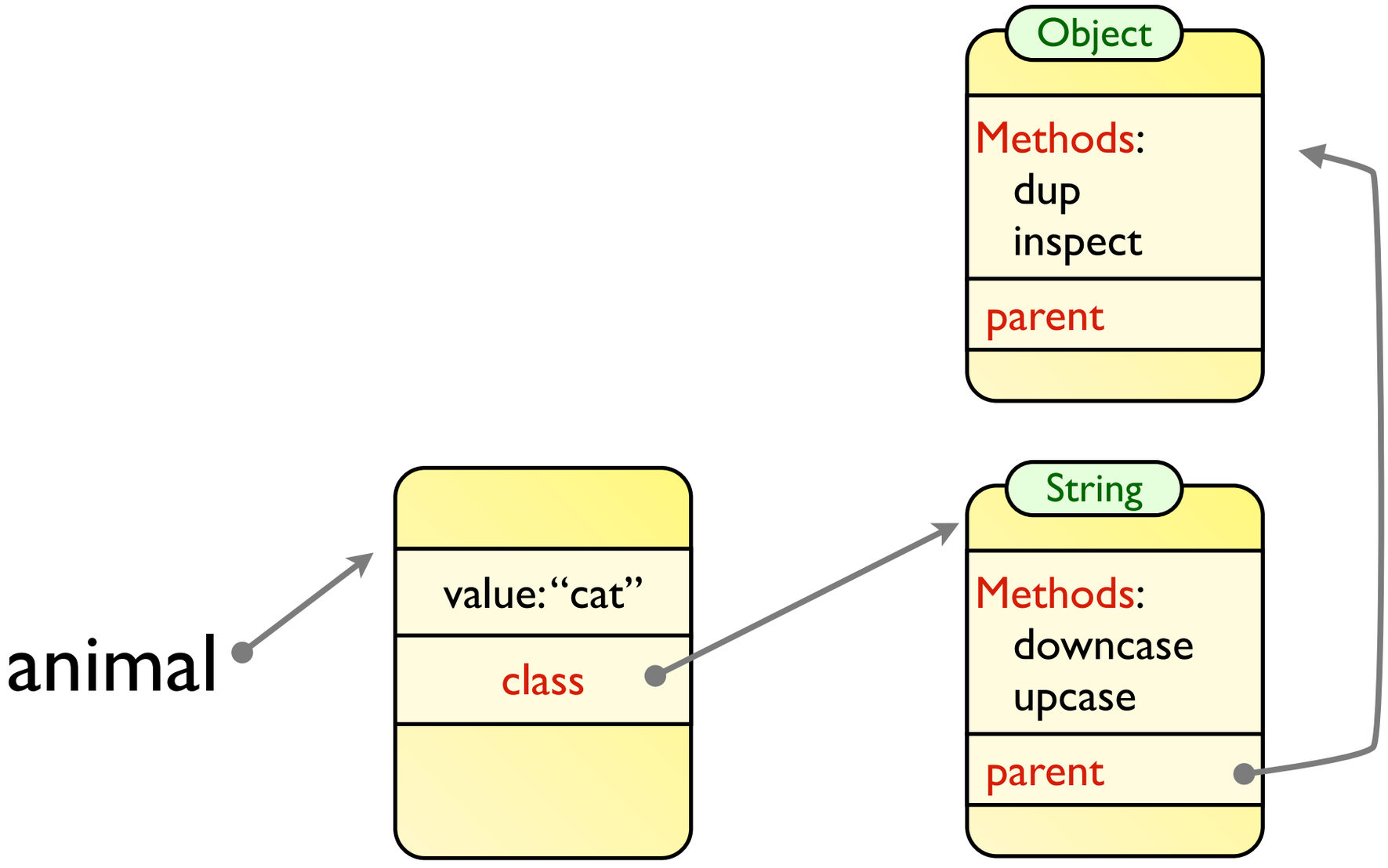
method call

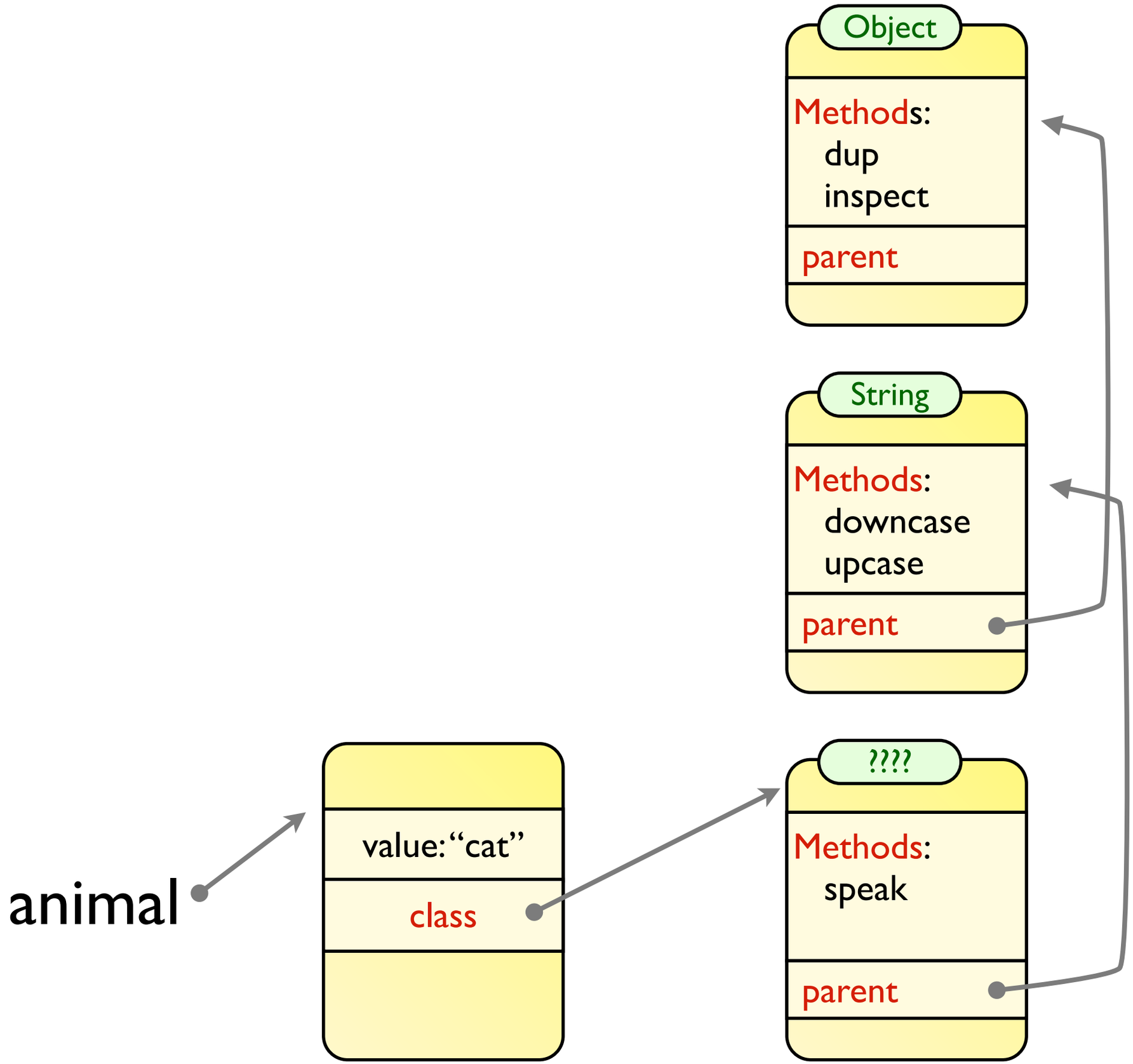
- Switch self to receiver
- Look up method name in self's method table
- Invoke method
- Restore original value of self

Method lookup:

- move one to the right
- then up




Object
Methods:
dup
inspect
parent
String
Methods:
downcase
upcase
parent
value: “cat”
class
animal


Object
Methods:
dup
inspect
parent
String
Methods:
downcase
upcase
parent
????
Methods:
speak
parent
animal
value: “cat”
class

Singleton Class

- sometimes called: Eigenclass, Metaclass, Ghost Class, Virtual Class, ...

- Normal class, but hidden

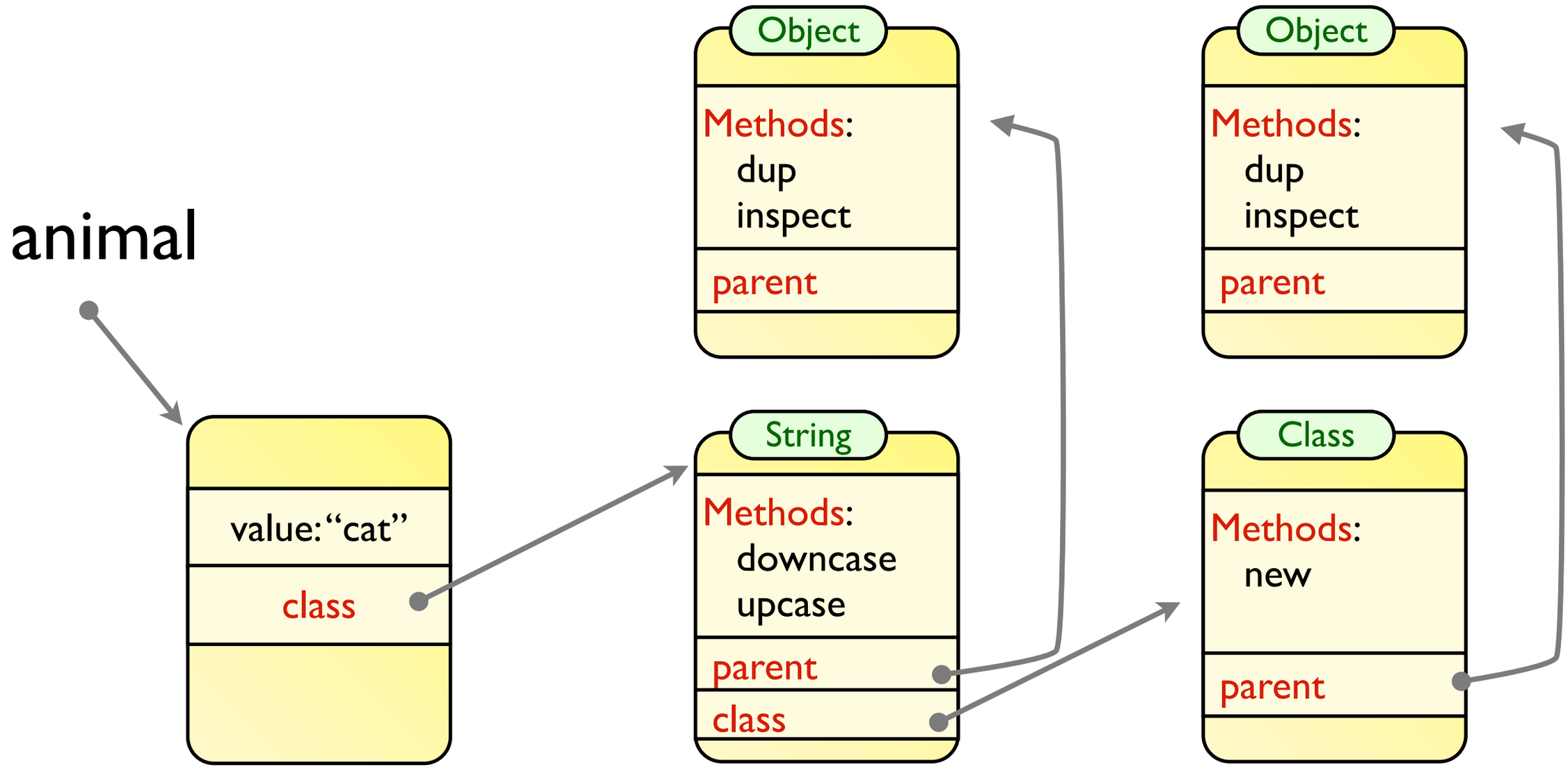
- Only one per object

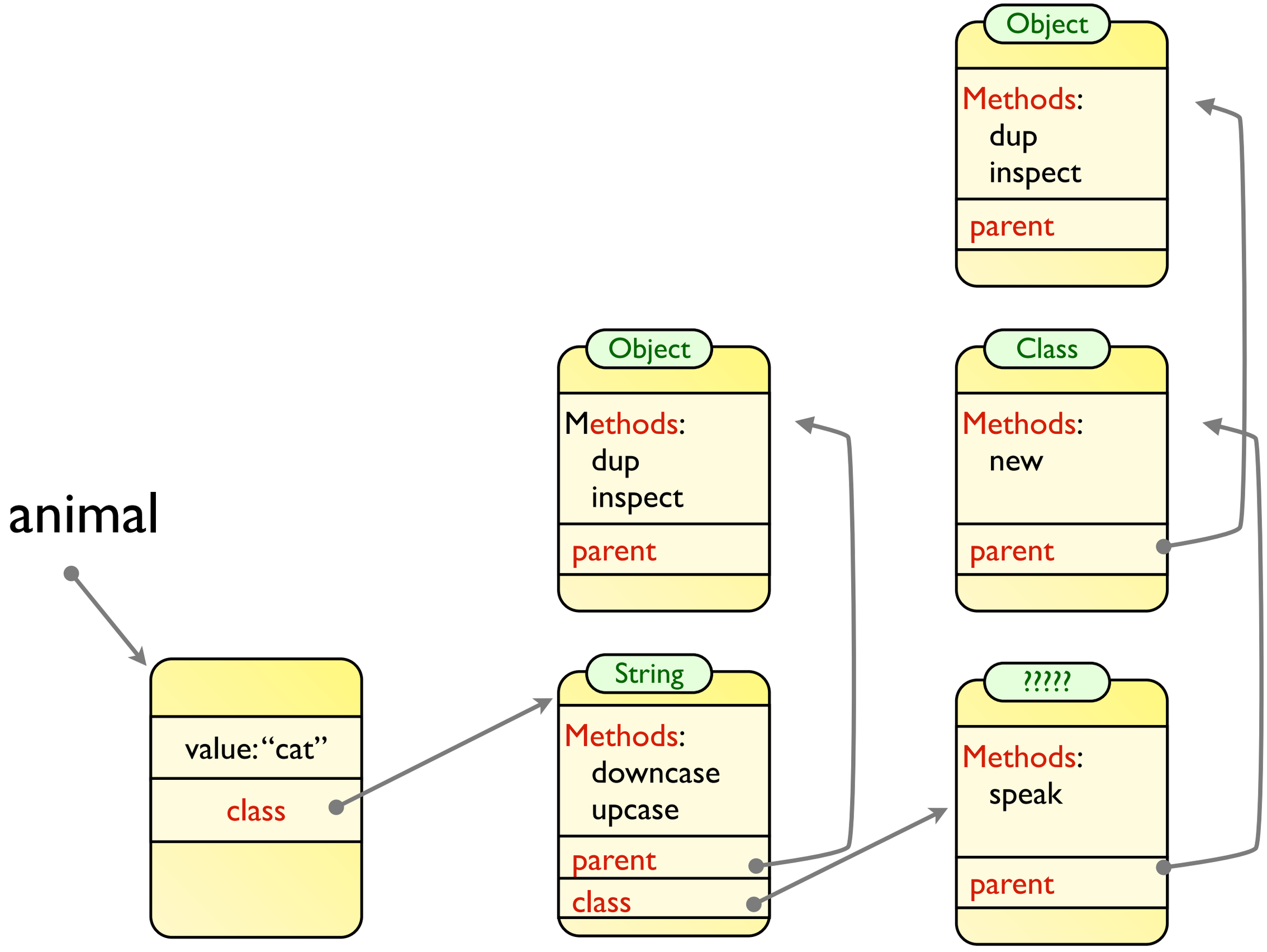
self

- Only two things can change

✓ method call

➡ class/module definition


Object
Methods:
dup
inspect
parent
Object
Methods:
dup
inspect
parent
animal
value: “cat”
class
String
Methods:
downcase
upcase
parent
class
Class
Methods:
new
parent


Object
Methods:
dup
inspect
parent
Class
Methods:
new
parent
Object
Methods:
dup
inspect
parent
animal
value: “cat”
class
String
Methods:
downcase
upcase
parent
class
?????
Methods:
speak
parent

Golden Rule

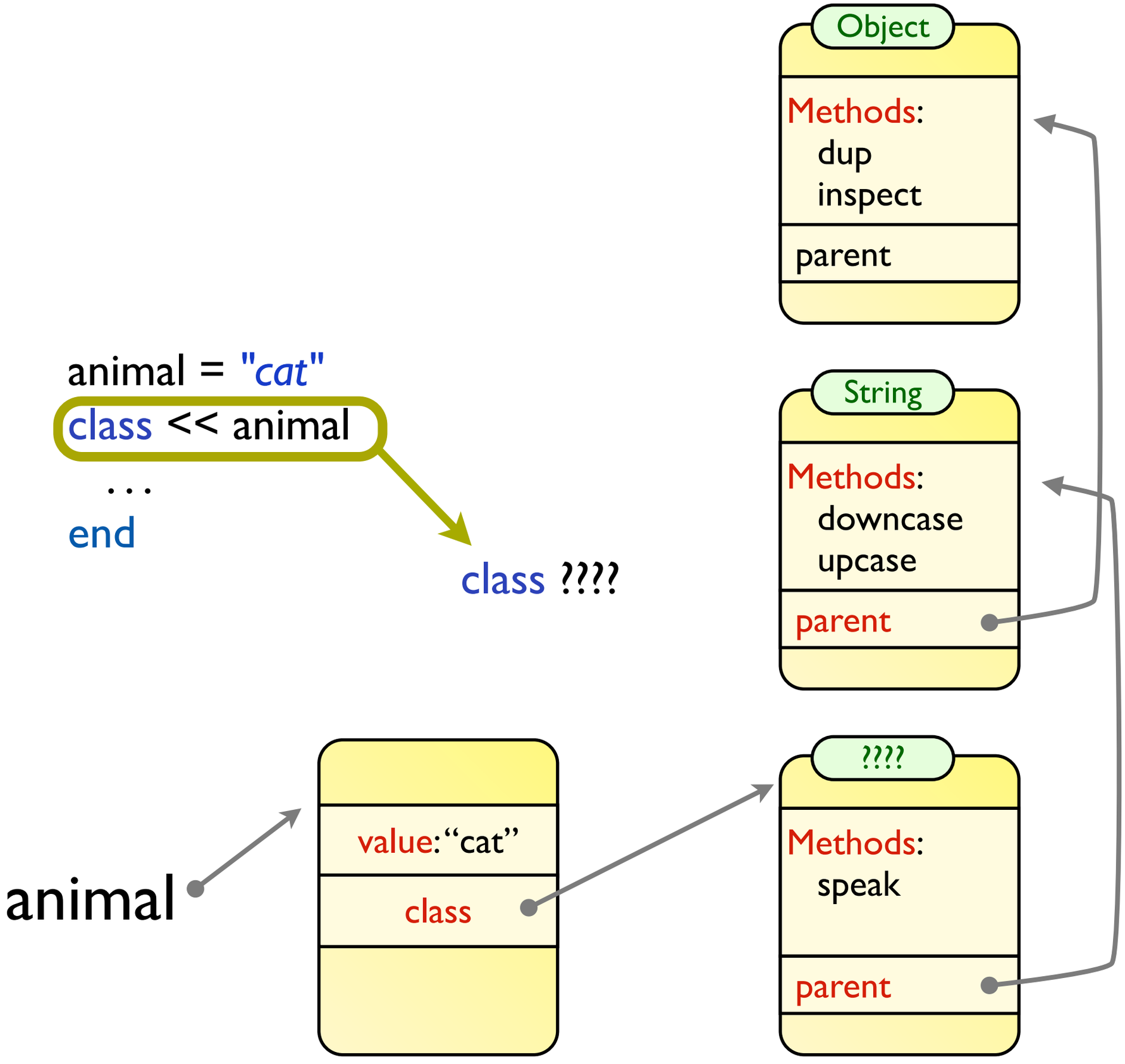
- Instance variables: look up in self
- Methods: look up in self's class

Inherit from Object

```
animal = "cat"
class << animal
  def speak
    puts "miaow"
  end
end

animal.speak # >> miaow

p animal.methods.sort # >> [ ... "sort_by", "speak", "split", ...]
```

```
animal = "cat"
class << animal
  ...
end
```

class ????



Inherit from Object

```
class MyClass
  class << self
    def class_method
      puts "Hi!"
    end
  end
end

MyClass.class_method   # >> Hi!
```

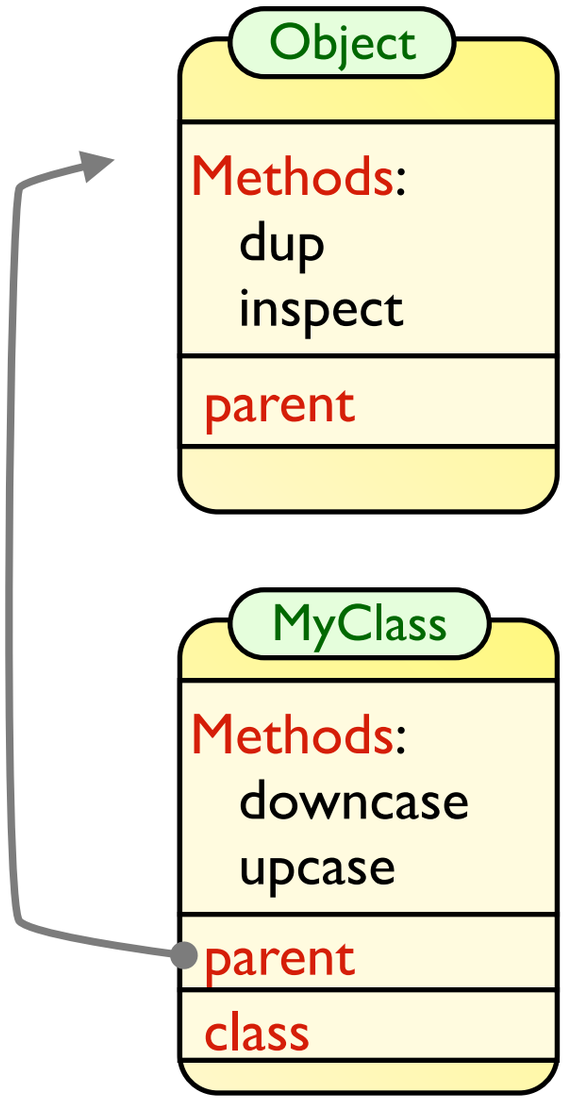
Inherit from Object

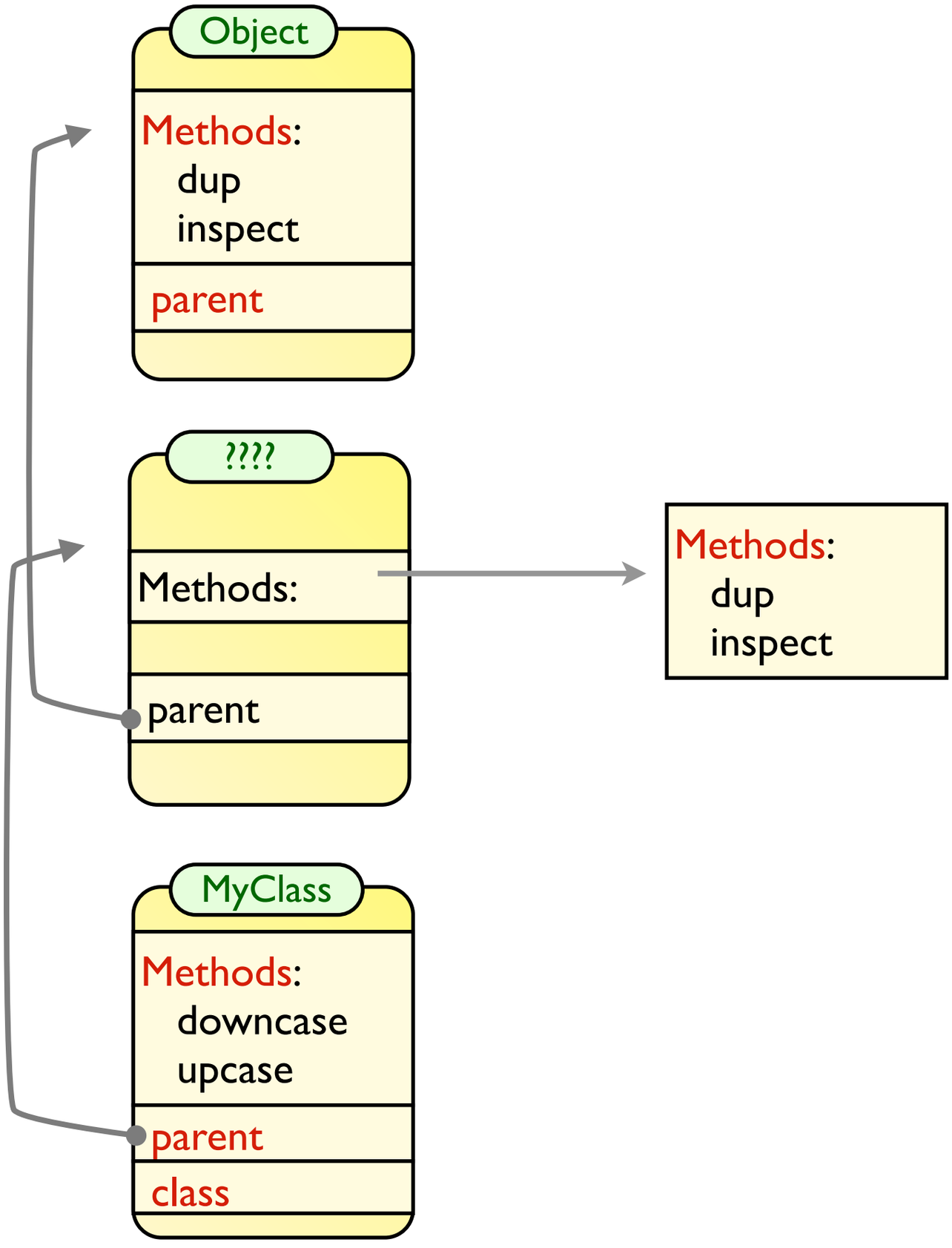
```
class Example
  @iv = 123
  class << self
    attr_reader :iv
  end
end

Example.iv = 123
puts Example.iv
```

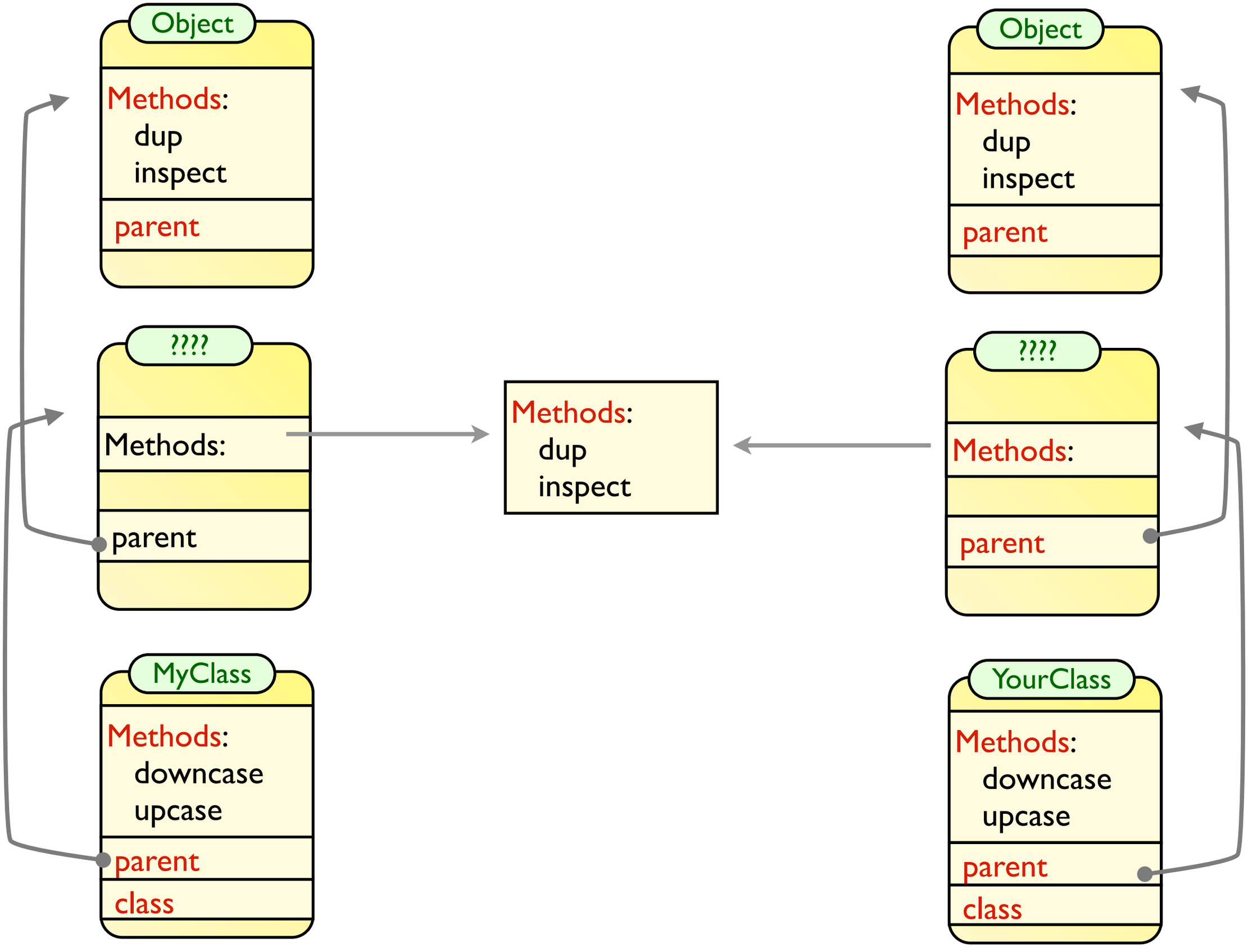
Inherit from Expression

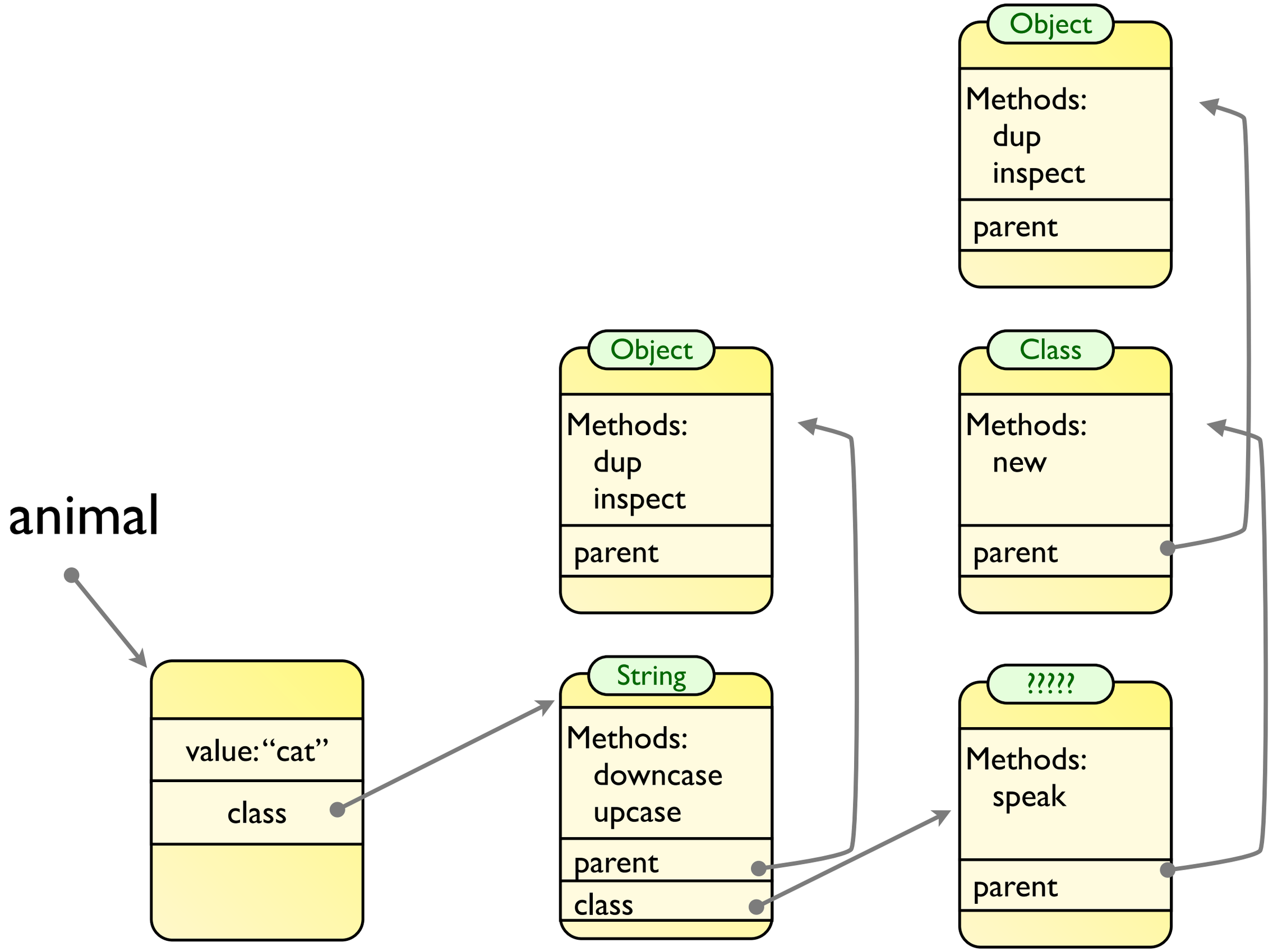
```
class Address < Struct.new(:street, :city, :province)
  def to_s
    "#{self.street}\n#{self.city}, #{self.province}"
  end
end

add = Address.new("123 Main", "Beijing", "75123")

puts add.to_s    # => 123 Main
                 # => Beijing, 75123
```

Include & Extend


Object
Methods:
dup
inspect
parent
MyClass
Methods:
downcase
upcase
parent
class

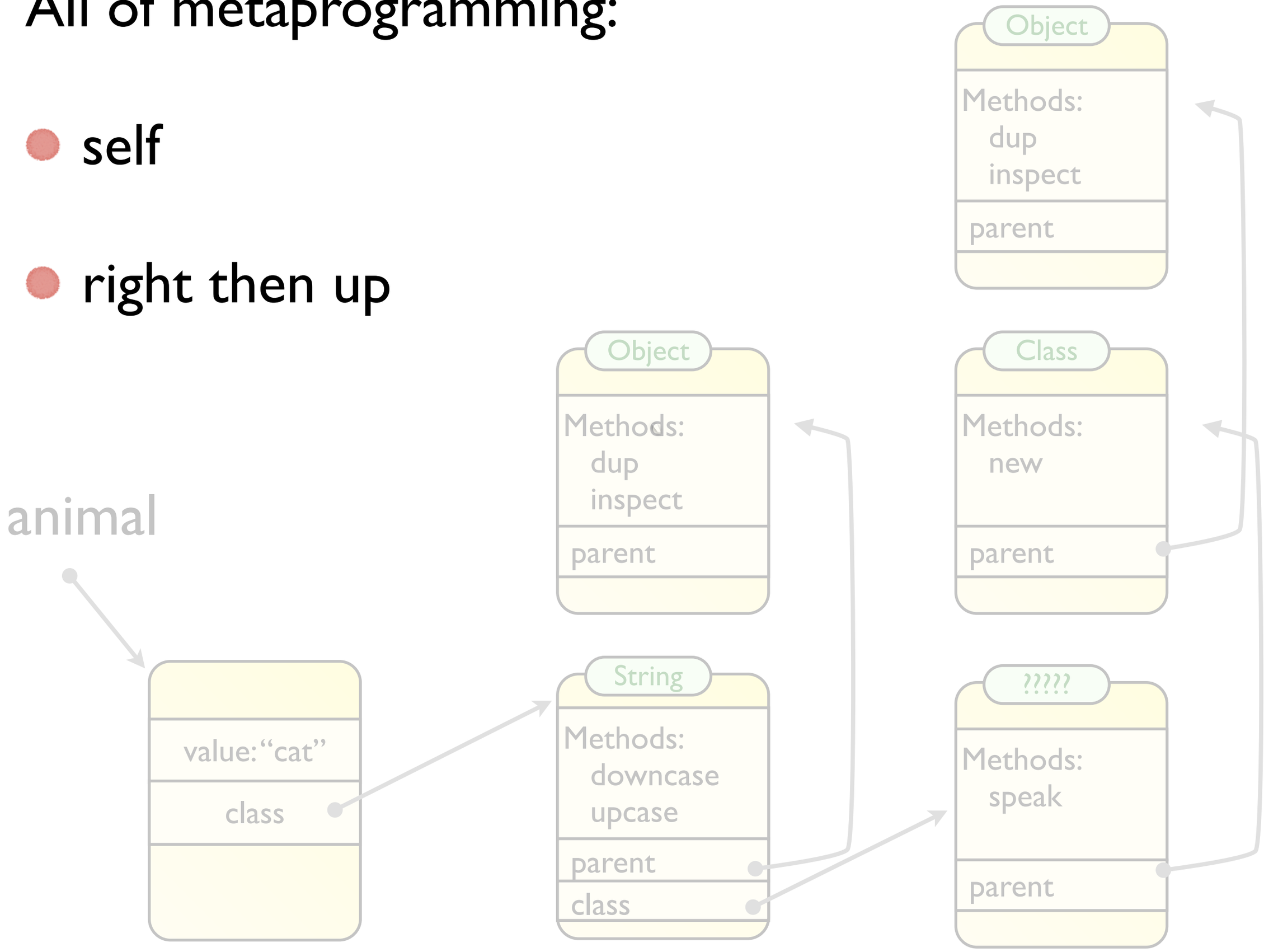

Object
Methods:
dup
inspect
parent
????
Methods:
parent
Methods:
dup
inspect
MyClass
Methods:
downcase
upcase
parent
class


Object
Methods:
dup
inspect
parent
Object
Methods:
dup
inspect
parent
????
Methods:
parent
Methods:
dup
inspect
????
Methods:
parent
MyClass
Methods:
downcase
upcase
parent
class
YourClass
Methods:
downcase
upcase
parent
class


Object
Methods:
dup
inspect
parent
Class
Methods:
new
parent
Object
Methods:
dup
inspect
parent
animal
value: “cat”
class
String
Methods:
downcase
upcase
parent
class
?????
Methods:
speak
parent

All of metaprogramming:

- self
- right then up



Ruby Object Model

Dave Thomas
The Pragmatic Programmers
http://pragprog.com