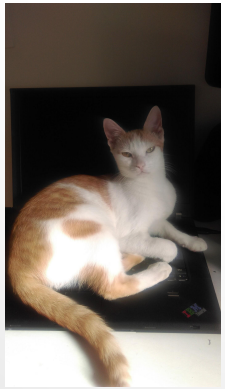
StatPro REVOLUTION®

Power to Analytics!

… is hiring



@code_kitten

Eigenclasses

… and Other Meta-Programming Concepts

Eigenclasses

… Why do we care?

Include some methods in an obj? # 1

```ruby
# attempt 1
module MyModule
  def my_method
    puts :in_my_method
  end
end

class << self
  ...

  self.class_eval do
    include MyModule
  end
end
```

Include some methods in an obj? # 2

```ruby
# attempt 2
module MyModule
  def my_method
    puts :in_my_method
  end
end

class << self
  ...

  self.instance_eval do
    include MyModule
  end
end
```

Include some methods in an obj? # 3

```ruby
# attempt 3
module MyModule
  def my_method
    puts :in_my_method
  end
end

class << self
  ...

  self.instance_eval do
    extend MyModule
  end
end
```

Include some methods in an obj? # 4

```ruby
# attempt 4
module MyModule
  def self.my_method
    puts :in_my_method
  end
end

class << self
  ...

  self.class_eval do
    extend MyModule
  end
end
```


SO CONFUSE
SUCH WIERD
VERY HUH
imgflip.com

What was I not getting?

`class_eval` and `instance_eval`:

- Eigenclasses
- Class Scope
- What is `self`

`class << self`:

- Eigenclasses

`include` and `extend`:

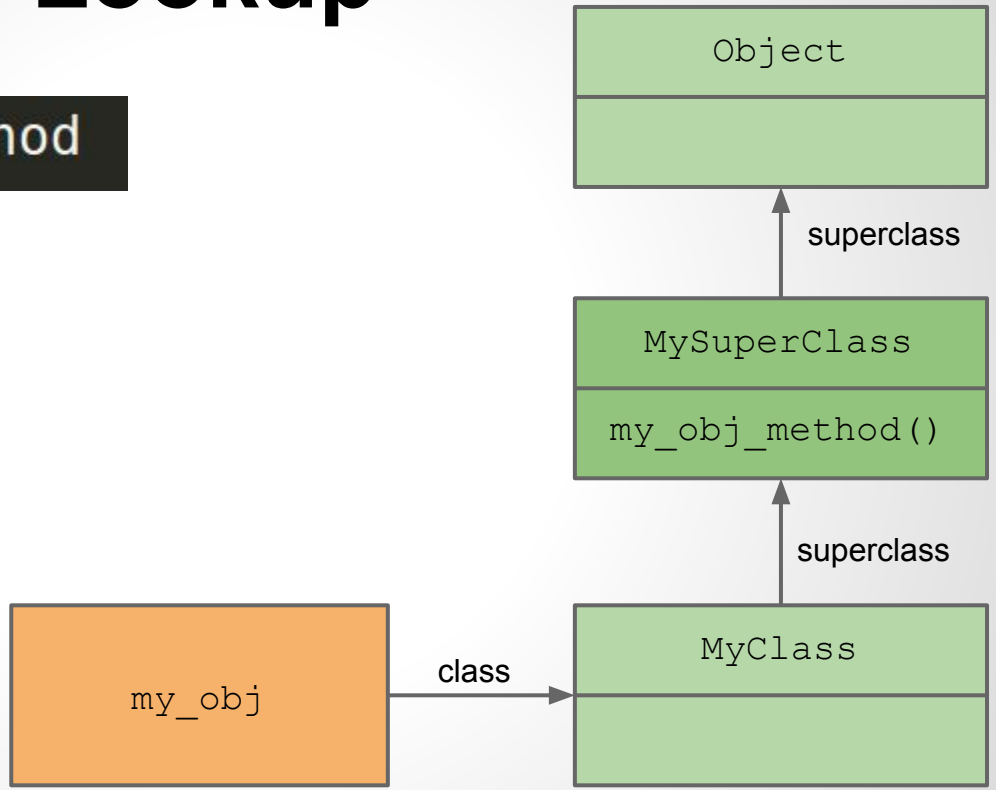
- Eigenclasses

Eigenclasses

```ruby
class MyClass
  class << self
    def my_method
      puts :my_method
    end
  end
end
```

Ruby Method Lookup

```
my_obj.my_obj_method
```

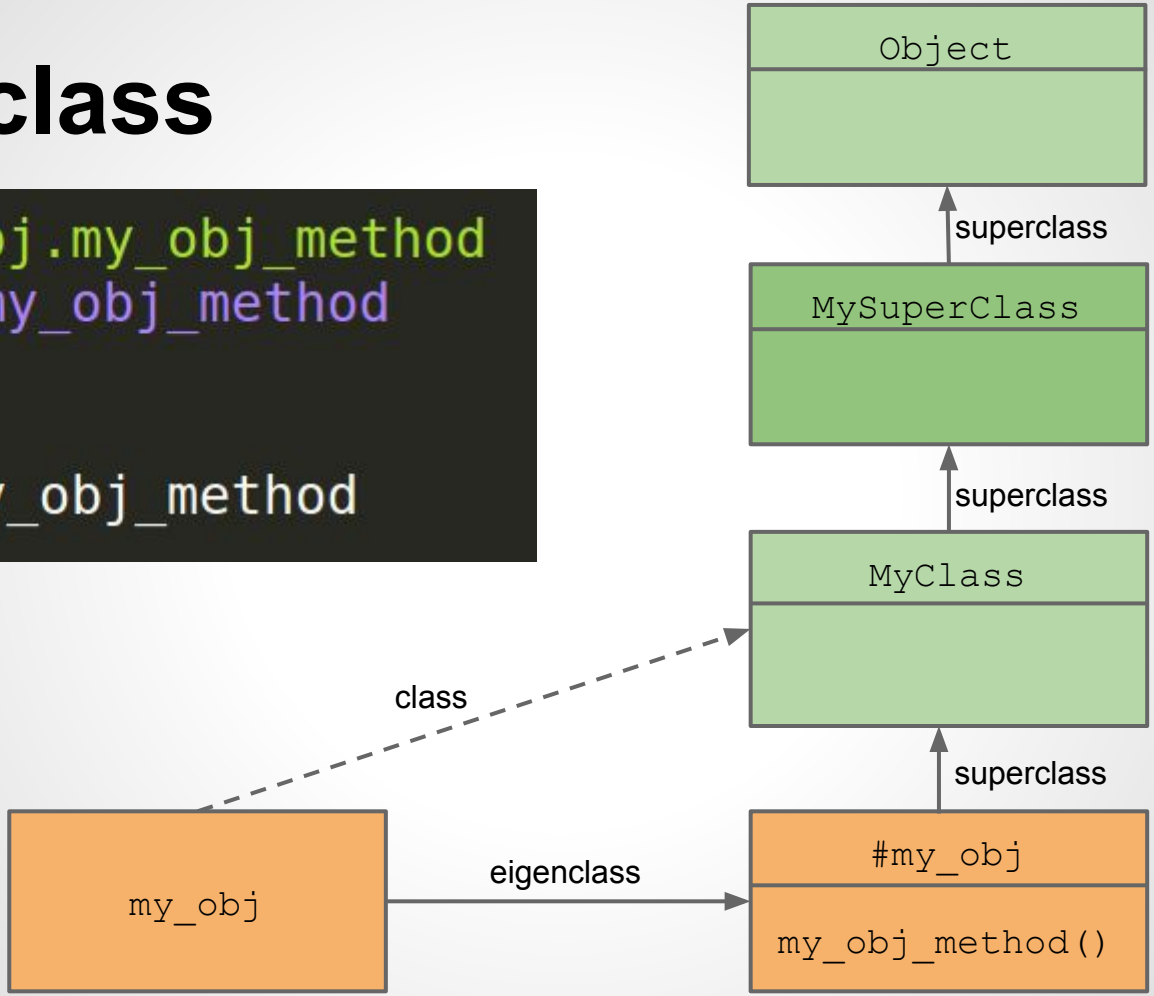


Singleton Methods

```ruby
def my_obj.my_obj_method
  puts :my_obj_method
end

my_obj.my_obj_method
```

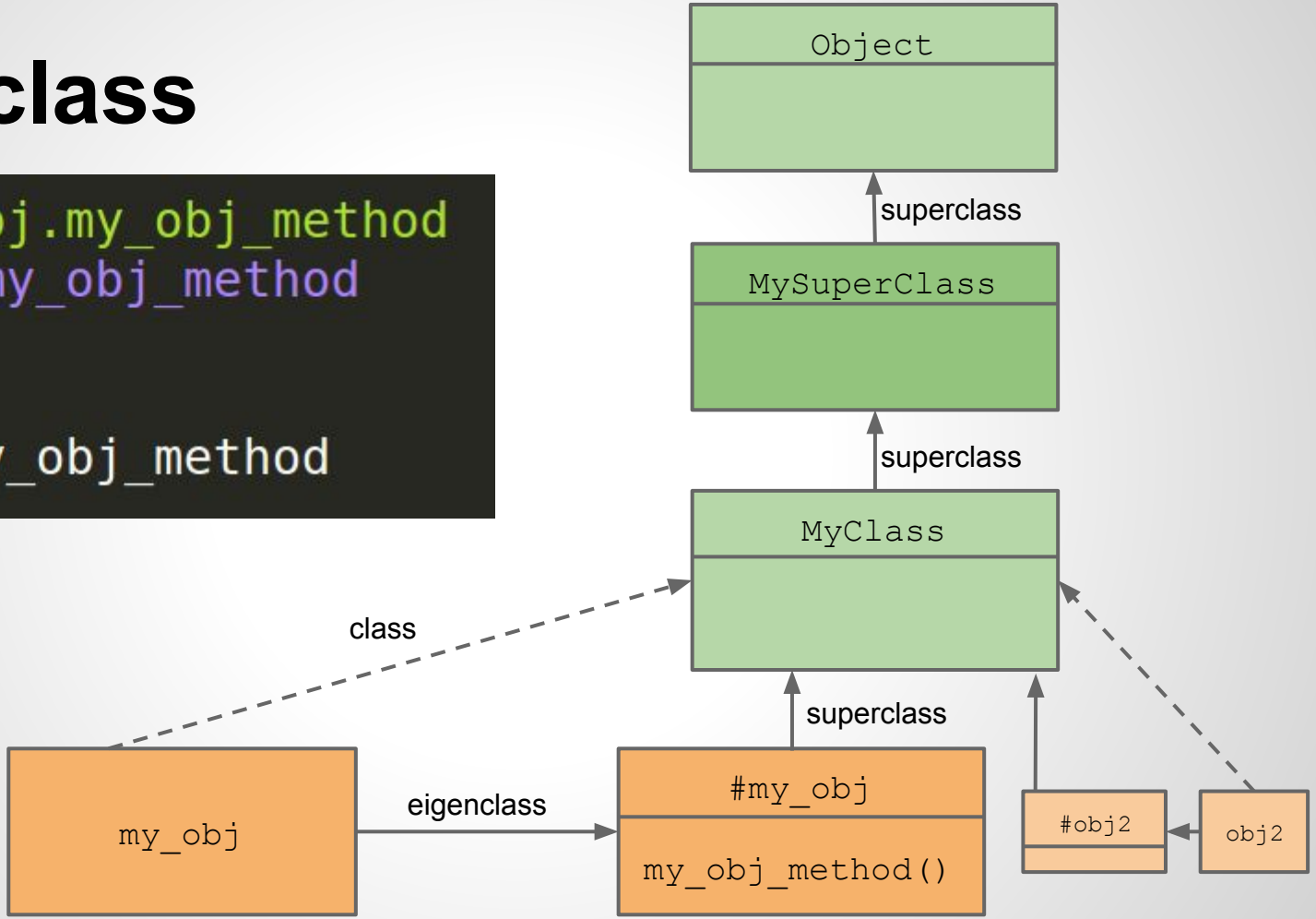
Eigenclass

```ruby
def my_obj.my_obj_method
  puts :my_obj_method
end

my_obj.my_obj_method
```



Eigenclass

```
def my_obj.my_obj_method
  puts :my_obj_method
end

my_obj.my_obj_method
```



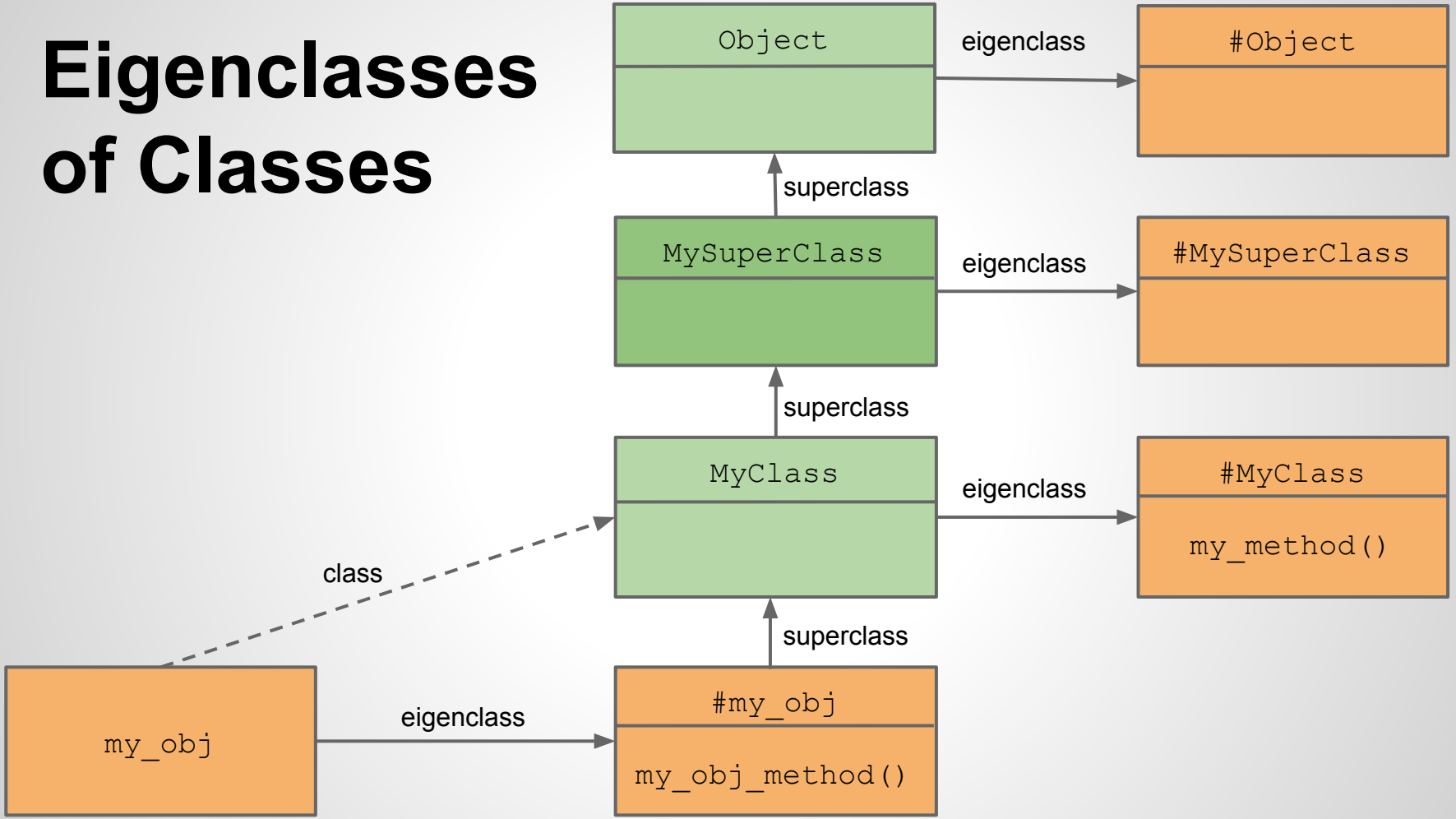
Inside the Eigenclass

```ruby
class << my_obj
  def my_obj_method
    puts :my_obj_method
  end
end

eigenclass = class << my_obj
  self
end

eigenclass # #<Class:#<MyClass:0x0000000bfdcf40>>
eigenclass.instance_methods # includes :my_obj_method
eigenclass.superclass # MyClass
```

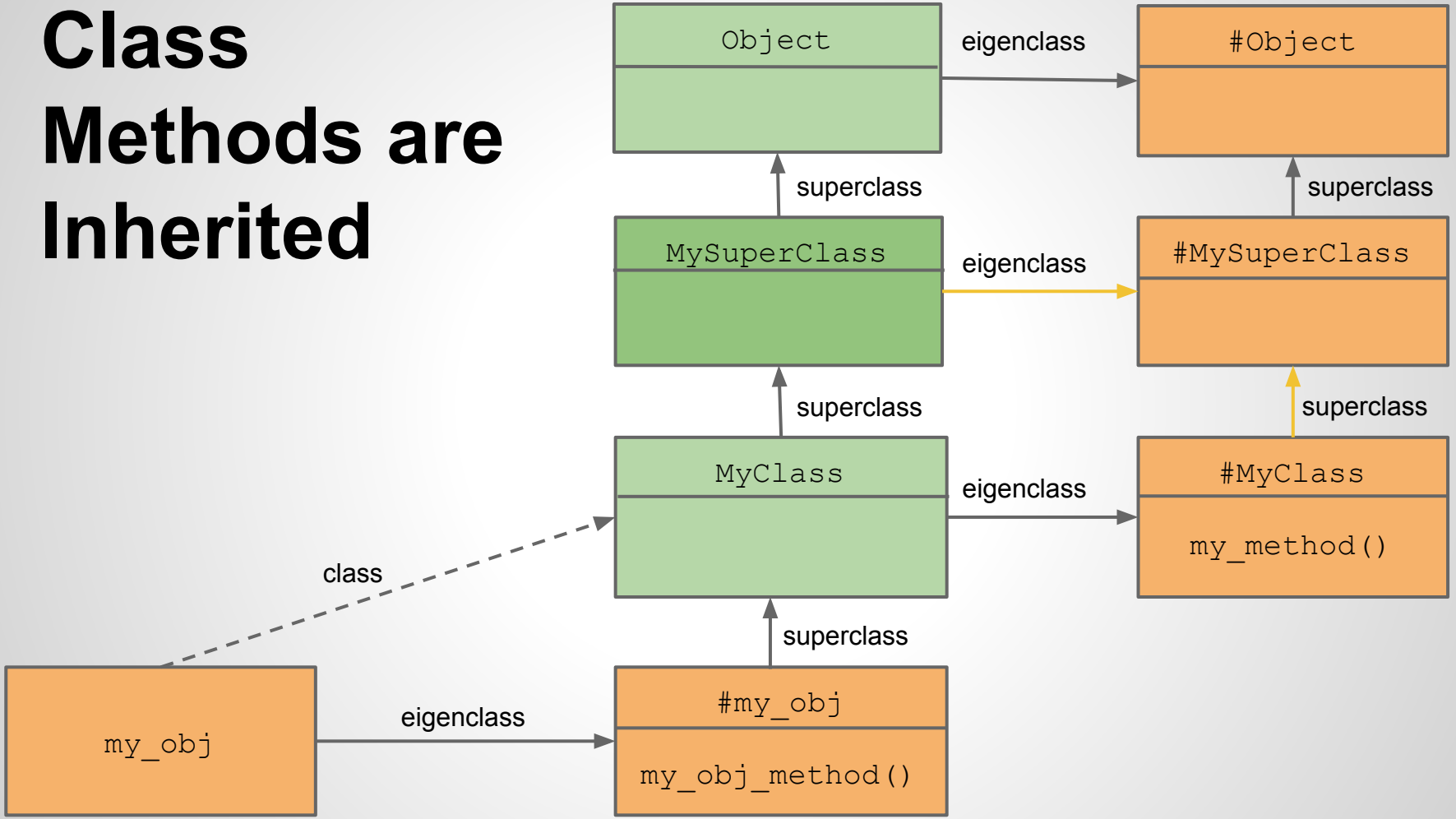
Class Methods
=
Singleton Methods

```ruby
def MyClass.my_method
  puts :my_method
end

class MyClass
  def self.my_method
    puts :my_method
  end
end

class MyClass
  class << self
    def my_method
      puts :my_method
    end
  end
end
```

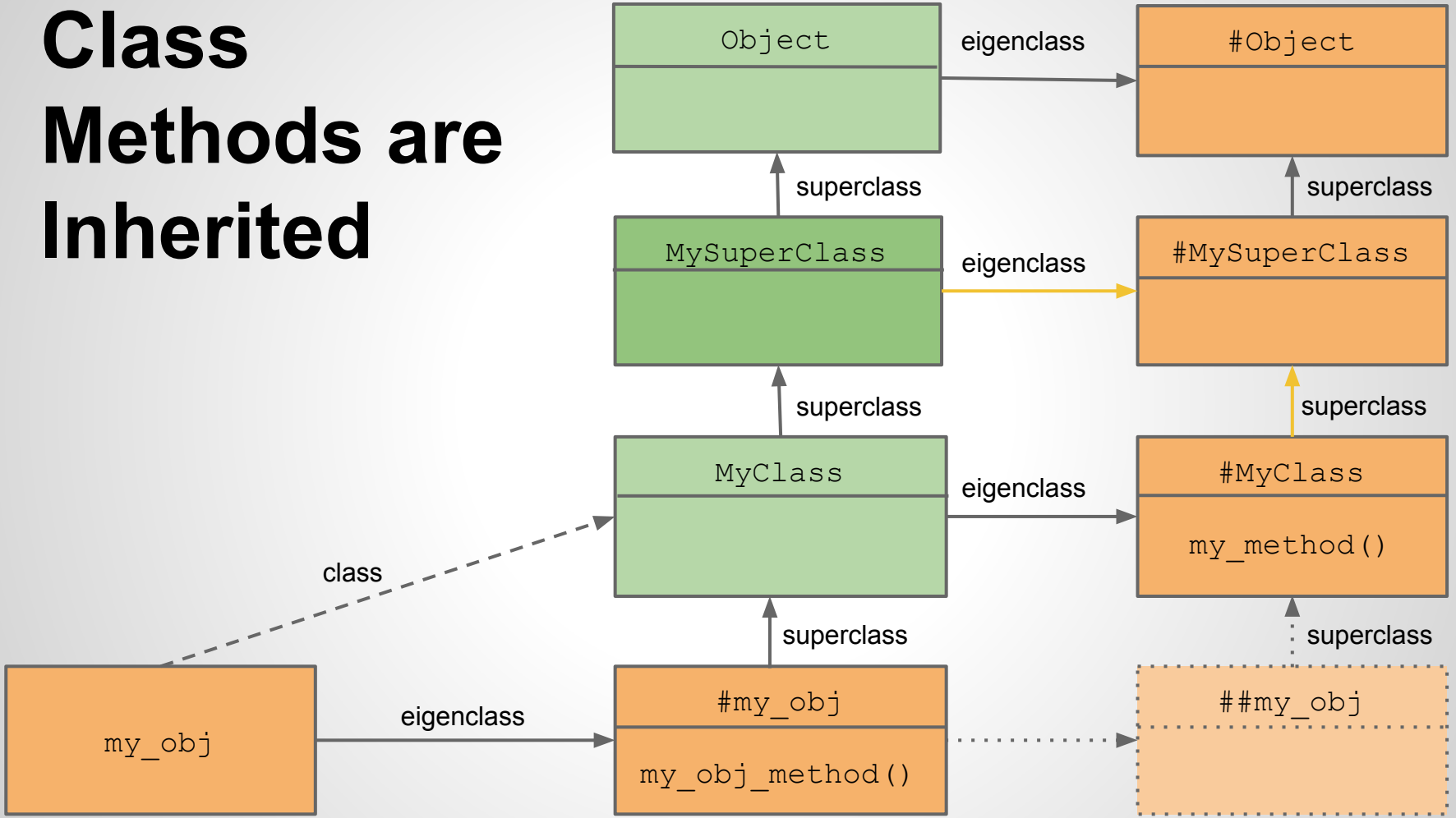
Eigenclasses of Classes



Class Methods are Inherited



Class Methods are Inherited



Jou ma se `class`

`class_eval` and `instance_eval`

`obj.class/instance_eval` do

	`self`	Class Scope
`class_eval`	the class	the class
`instance_eval`	receiver (`obj`)	eigenclass

`end`

No Surprises

```ruby
# instance method
MyClass.class_eval do
  def bye
    puts :bye
  end
end

# class method
MyClass.class_eval do
  def self.later
    puts :later
  end
end
```

```ruby
m = MyClass.new

# singleton method
m.instance_eval do
  def hello
    puts :hello
  end
end

# singleton method
m.instance_eval do
  def self.hi
    puts :hi
  end
end
```

`class_eval` and `instance_eval`

`obj.class/instance_eval` do

	`self`	Class Scope
`class_eval`	the class	the class
`instance_eval`	receiver (`obj`)	eigenclass

`end`


... THE CLASS ?
WTF ?
imgflip.com

`class_eval` and `instance_eval`

`obj.class/instance_eval` do

	`self`	Class Scope
`class_eval`	the class class: that class instance: the eigenclass	the class class: that class instance: the eigenclass
`instance_eval`	receiver (`obj`)	eigenclass

`end`

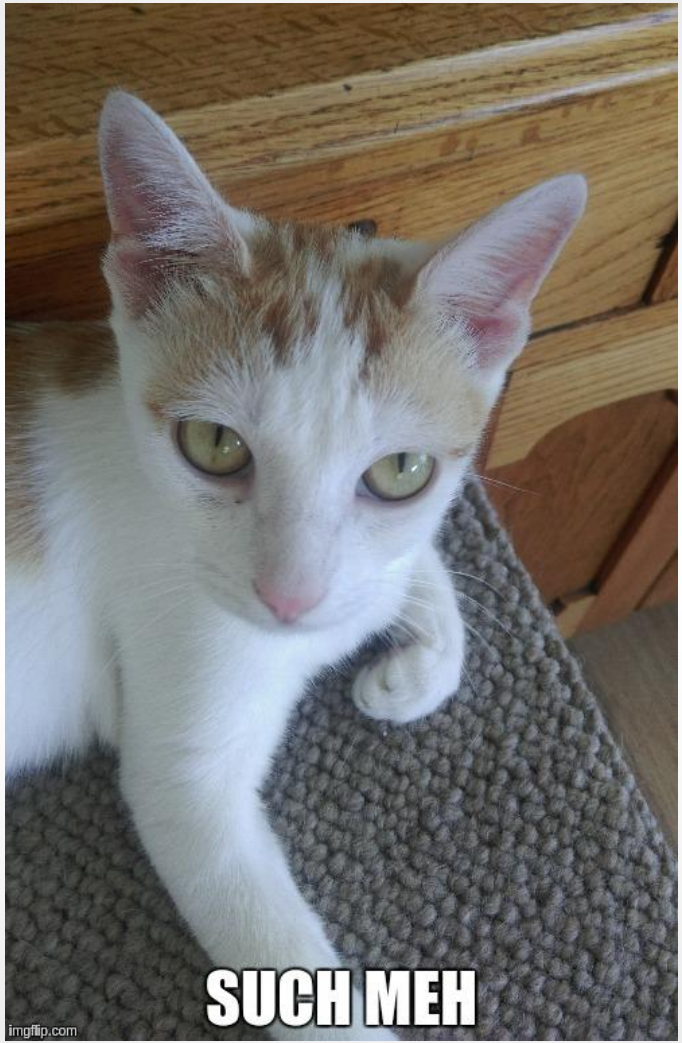
`include` vs `extend`

- `include MyModule`: Includes the ***instance methods*** of `MyModule` in the receiver ***class***

- `extend MyModule`: Includes the ***instance methods*** of `MyModule` in the ***Eigenclass*** of the receiver

include vs extend

```
class MyClass
  extend MyModule
end

class MyClass
  class << self
    include MyModule
  end
end
```


SUCH MEH
imgflip.com

`include` in Eigenclass

```
class MyClass
  blah blah blah
  ...
  class << self
    blah blah
    blah blah
    ...
    include MyModule
    ...
    blah
  end
  blah
end
```

include vs extend

```ruby
module MyModule
  def self.my_module_method
    puts :my_module_method
  end

  def my_method
    puts :my_method
  end
end

class MyClass
  include MyModule # instance methods => instance methods
  extend MyModule # instance methods => class methods
end
```

Including Class Methods

```ruby
module MyModule
  def my_module_method
    puts :my_module_method
  end
end

class MyClass
  extend MyModule
end
```

More Reading