

OOP,
metaprogramming,
blocks, iterators,
mix-ins, duck typing.
Code style

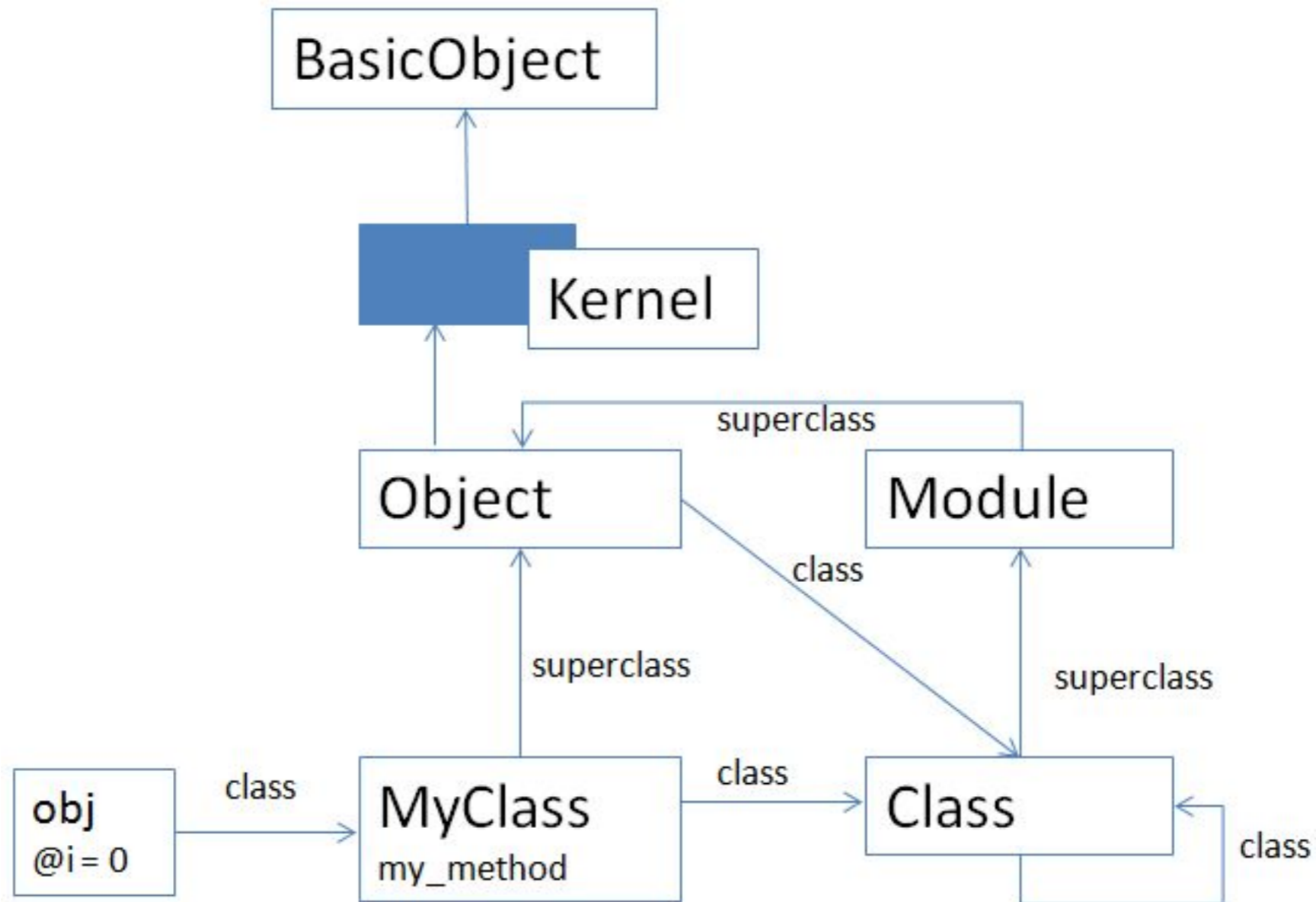
Anton Shemerey

- <https://github.com/shemerey>
- <https://twitter.com/shemerey>
- <https://www.facebook.com/shemerey>
- <https://www.linkedin.com/in/shemerey>
- shemerey@gmail.com
- etc.

Code style

- <https://github.com/bbatsov/ruby-style-guide>
- <https://github.com/bbatsov/rails-style-guide>
- Use two spaces per tab!
- explicit “return” - is evil

OOP - Everything is object, not quite but still...



“hello” - this is object too

```
1
2 a = "hello" # instance
3 b = a.dup   # new object
4
5 # instance method (for object)
6 class << a
7   def to_s
8     "#{self} world"
9   end
10
11   def say_hello
12     self.to_s
13   end
14 end
15
16 a.to_s          # => "hello world"
17 b.to_s          # => "hello"
18 a.methods - b.methods # => [:say_hello]
19 a.class.methods - b.class.methods # => []
```

```
1
2 def a.to_s
3   "#{self} world"
4 end
5
6 def a.say_hello
7   self.to_s
8 end
9
```

42 - this is object too

- `42.instance_variables # => []`
- `42.instance_variable_set(:@hello, 'world')`
- `42.instance_variables # => [:@hello]`
- `42.object_id # => 85`

true - this is object too

- `true.instance_variables # => []`
- `true.instance_variable_set(:@false, true)`
- `true.instance_variables # => [:@false`
- `true.object_id # => 2`
- `false.object_id # => 0`
- `nil.object_id # => 4`

Class

```
3 class Point
4   def initialize(x, y)
5     @x, @y = x, y
6   end
7 end
8
9 if __FILE__ == $PROGRAM_NAME
10  puts Point.new(1, 10).inspect #<Point:0x0bf78 @x=1, @y=10>
11 end
```


attribute reader/writer

```
1 class Point
2   def initialize(x)
3     @x = x
4   end
5
6   def x
7     @x
8   end
9
10  def x=(x)
11    @x = x
12  end
13 end
14
```

```
1 class Point
2   attr_accessor :x
3
4   def initialize(x)
5     @x = x
6   end
7 end
8
```

```

1 module ReadWrite
2   def rw(*params)
3     params.each do |_
4       define_method attr do          # def x
5         instance_variable_get :"@#{attr}" # @x
6       end                             # end
7
8       define_method :("#{attr}=(val)" do # def x=(val)
9         instance_variable_set :"@#{attr}=", val # @x=val
10      end                               # end
11    end
12  end
13 end
14
15 class Object
16   extend ReadWrite
17 end
18
19 class Point
20   rw :x, :y
21
22   def initialize(x)
23     @x = x
24   end
25 end
26

```

```
# params = [:x, :y]
```

Class / Module

- `Module.ancestors`
 - `# => [Module, Object, Kernel, BasicObject]`
- `Class.ancestors`
 - `# => [Class, Module, Object, Kernel, BasicObject]`

class A; end

- new
- inheritance
- include class A - computer says no
- extend class A - computer says no
- computer says no - <http://bit.ly/gQX24>

module B; end

- new - computer says no
- inheritance - computer says no
- include B
- extend B
- computer says no - <http://bit.ly/gQX24>

include/extend

```
1 module M
2   def inst_method
3     "instance"
4   end
5 end
6
7
8 class Point
9   include M
10 end
11
12 if __FILE__.$PROGRAM_NAME
13   puts Point.new.inst_method
14 end
15
```

```
1 module M
2   def cl_method
3     "instance"
4   end
5 end
6
7
8 class Point
9   extend M
10 end
11
12 if __FILE__.$PROGRAM_NAME
13   puts Point.cl_method
14 end
15
```

```
1 module M
2   def self.included(base)
3     base.extend(ClassMethods)
4   end
5
6   # -- instance methods --
7
8   module ClassMethods
9     # -- class methods --
10    end
11  end
12
13
14  class Point
15    include M
16
17    def initialize(x)
18      @x = x
19    end
20  end
```

Classes and Objects

- More information: <http://bit.ly/YZBfmp>

Ruby Blocks

```
1 %w[first second third].each do |item|
2   item.capitalize
3 end
4
5 %w[first second third].each{|i| i.upcase }
6
7 %w[first second third].each <<= block
```

Yield



Enumerable

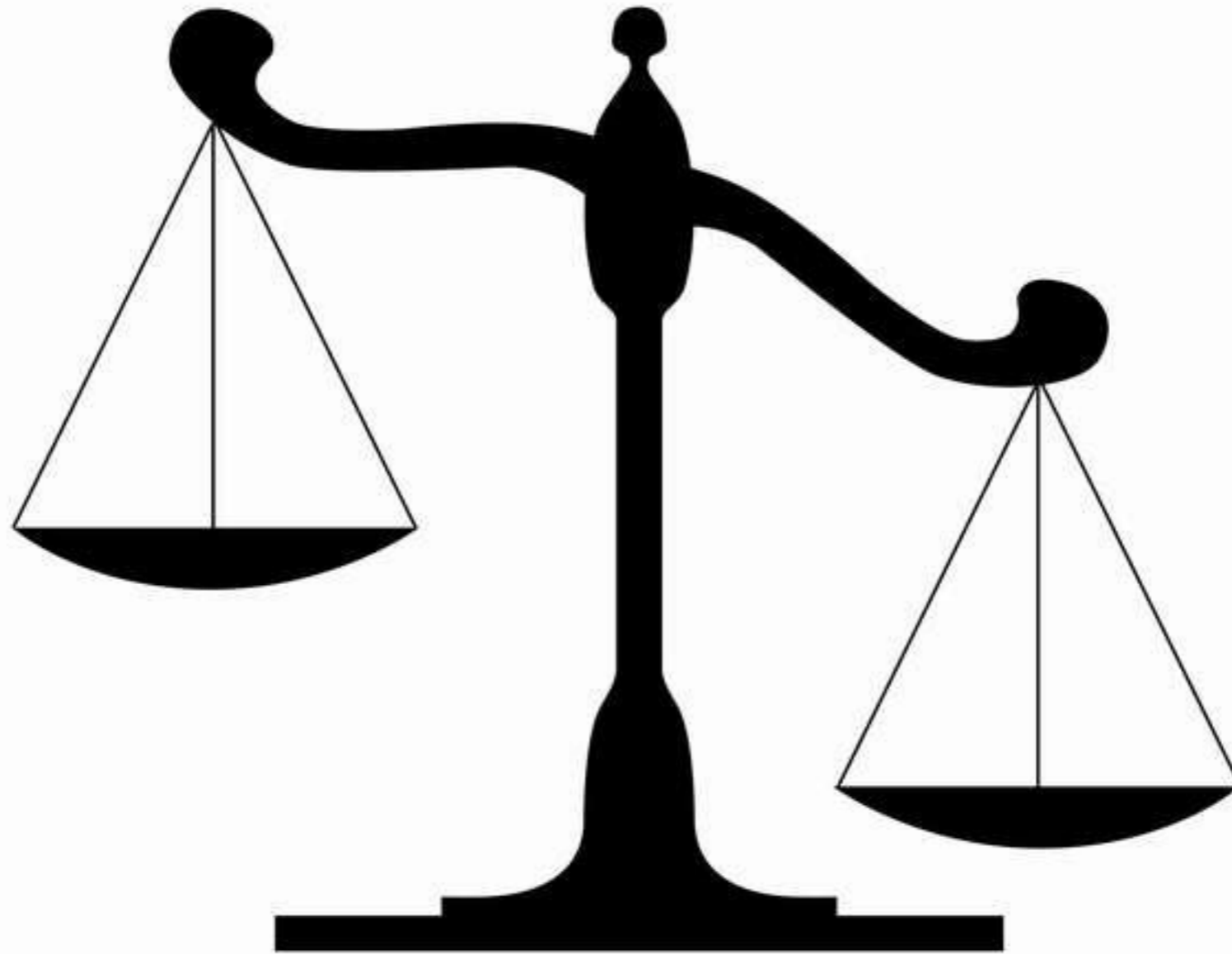


<http://ru.wikibooks.org/wiki/Ruby/Справочник/Enumerable>

```
1 require "ostruct"
2
3 Point = Struct.new(:x, :y) do
4   def distance
5     x**2 + y**2)
6   end
7 end
8
9 class PointList
10  include Enumerable <<<<<<<
11  # -- code omitted --
12  def each
13    for item in @data
14      yield item
15    end
16  end
17 end
18
19 if __FILE__ == $PROGRAM_NAME
20  list = PointList.new << Point.new(1,3) << Point.new(2,1) << Point.new(2,2)
21  list.sort_by(&:distance)
22  list.sort_by {|e| e.distance }
23 end
```

Comparable

1, $\bar{0}$, -1



Comparable

```
1 require "ostruct"
2
3 Point = Struct.new(:x, :y) do
4   include Comparable <<<<<<
5
6   def distance
7     Math.sqrt(x**2 + y**2)
8   end
9
10  def <=>(other)
11    self.distance <=> other.distance
12  end
13 end
14
15 if __FILE__ == $PROGRAM_NAME
16   puts Point.new(1,3) == Point.new(2,1) # false
17   puts Point.new(1,3) == Point.new(3,1) # true
18   puts Point.new(2,3) >= Point.new(1,1) # true
19   puts Point.new(2,3) <= Point.new(1,1) # false
20   puts Point.new(2,2).between? Point.new(2,1), Point.new(2,3) #true
21 end
```

Mad Ruby



#to_proc (&)

```
1 class Symbol
2   def to_proc
3     Proc.new { |*args| args.shift.__send__(self, *args) }
4   end
5 end
```

```
19 if __FILE__ == $PROGRAM_NAME
20   list = PointList.new << Point.new(1,3) << Point.new(2,1..2)
21   list.sort_by(&:distance)
22   list.sort_by {|e| e.distance }
23 end
```

```
1 [1, 2, 3].map(&:to_s) # => ["1", "2", "3"]
2 [3, 4].inject(&:*) # => 60
```

Advanced Examples

```
1 File.open('nginx.log', 'r') do |file|
2   while line = file.gets
3     puts line
4   end
5 end
7 # -- possible implementation --
8 class File
9   def open(file, option)
10    if block_given?
11      f = fopen(file, option)
12      yield f
13      f.close
14    end
15  end
16 end
```

```
1 ActiveRecord::Base.transaction do
2   # -- code omitted --
3 end
4
5 # -- possible implementation --
module ActiveRecord
7  class Base
8    def transaction
9      # -- start transaction
10     yield
11     # -- commit transaction
12     rescue ActiveRecord::Rollback
13     # -- roll back transaction
14     ensure
15     # -- close transaction
16   end
17 end
18 end
```

The End

