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

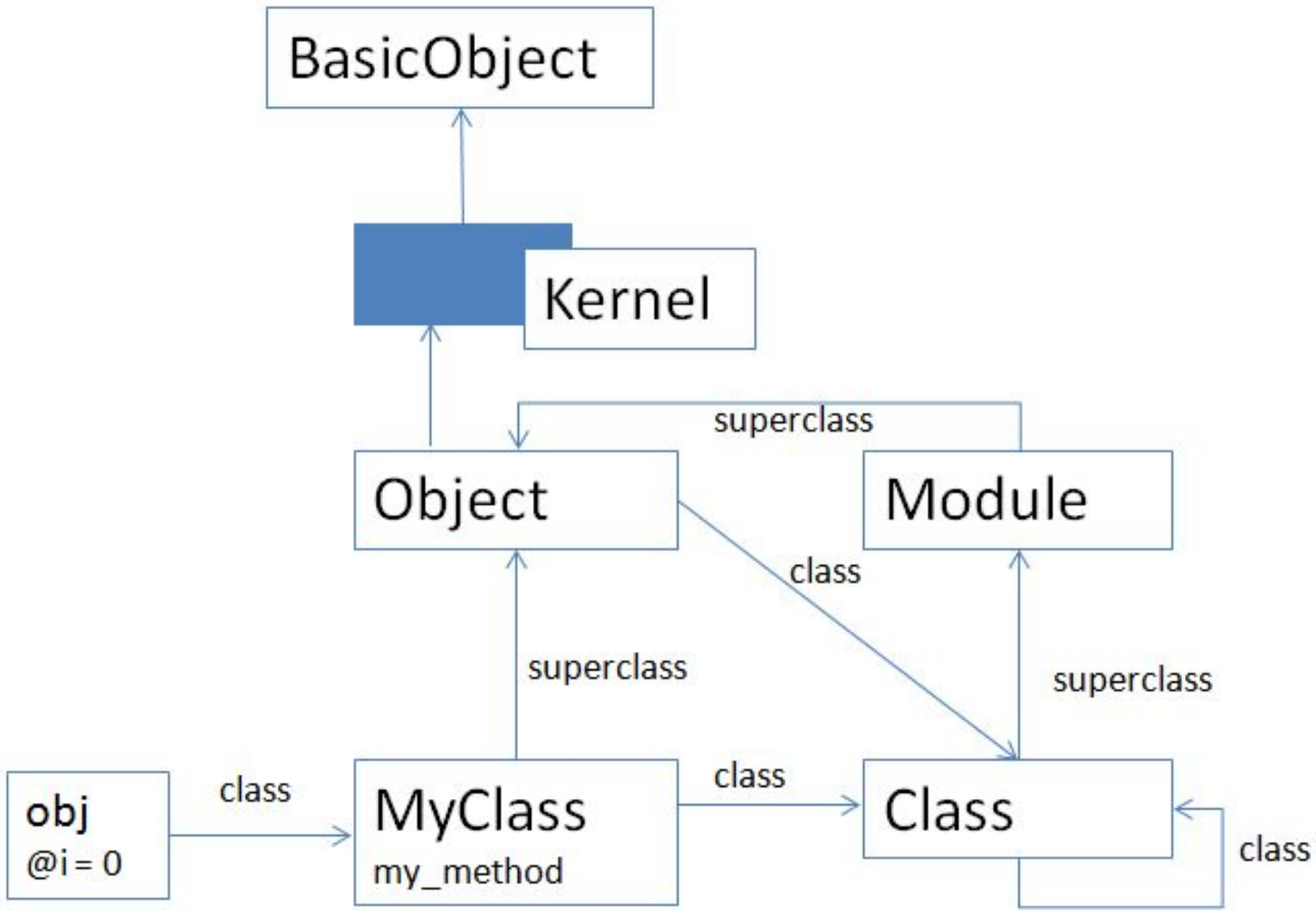
Anton Shemerey

- 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

```ruby

a = "hello" # instance
b = a.dup   # new object

# instance method (for object)
class << a
  def to_s
    "#{self} world"
  end

  def say_hello
    self.to_s
  end
end

a.to_s                          # => "hello world"
b.to_s                          # => "hello"
a.methods - b.methods           # => [:say_hello]
a.class.methods - b.class.methods # => []
```

```ruby

def a.to_s
  "#{self} world"
end

def a.say_hello
  self.to_s
end

```

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

```ruby
class Point
  def initialize(x)
    @x = x
  end

  def x
    @x
  end

  def x=(x)
    @x = x
  end
end
```

```ruby
class Point
  attr_accessor :x

  def initialize(x
  @x = x
  end
end
```

```
1  module ReadWrite
2    def rw(*params)                                  # params = [:x, :y]
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
```

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
```

```ruby
module M
  def self.included(base)
    base.extend(ClassMethods)
end

# -- instance methods --

module ClassMethods
  # -- class methods --
  end
end


class Point
  include M

  def initialize(x)
    @x = x
  end
end
```

Classes and Objects

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

Ruby Blocks

```ruby
%w[first second third].each do |item|
    item.capitalize
end

%w[first second third].each{ |i| i.upcase }
```

6
7 %w[first second third].each <<= block

Yield

```ruby
require "ostruct"
Point =  Struct.new(:x, :y)

class PointList
  def initialize
   @data = []
  end

  def <<(other)
    @data << other
  end

  def each
    for item in @data
      yield item  <<<<<<<<<<<<
    end
  end
end

if __FILE__ == $PROGRAM_NAME
  list = PointList.new << Point.new(1,3) << Point.new(2,1)
  list.each do |point|
    puts point  #<struct Point x=1, y=3>, #<struct Point x=2, y=1>
  end
end
```

```ruby
  def each(block = Proc.new)
    for item in @data
      block.call(item)
    end
  end
```

```ruby
  def each(&block)
    for item in @data
      block.call(item)
    end
  end
```

Enumerable

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

```ruby
require "ostruct"

Point = Struct.new(:x, :y) do
  def distance
    Math.sqrt(x**2 + y**2)
  end
end

class PointList
  include Enumerable
  # -- code omitted --

  def each
    for item in @data
      yield item
    end
  end
end

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

Comparable
1, 0, -1

Comparable

```ruby
require "ostruct"

Point =  Struct.new(:x, :y) do
  include Comparable <<<<<<
  def distance
    Math.sqrt(x**2 + y**2)
  end

  def <=>(other)
    self.distance <=> other.distance
  end
end

if __FILE__ == $PROGRAM_NAME
  puts Point.new(1,3) == Point.new(2,1) # false
  puts Point.new(1,3) == Point.new(3,1) # true
  puts Point.new(2,3) >= Point.new(1,1) # true
  puts Point.new(2,3) <= Point.new(1,1) # false
  puts Point.new(2,2).between? Point.new(2,1), Point.new(2,3) #true
end
```

Mad Ruby

#to_proc (&)

```ruby
class Symbol
  def to_proc
    Proc.new { |*args| args.shift.__send__(self, *args) }
  end
end
```

```ruby
if __FILE__ == $PROGRAM_NAME
  list = PointList.new << Point.new(1,3) << Point.new(2,1.2,2)
list.sort_by(&:distance)
list.sort_by {|e| e.distance }
end
```

```ruby
[1, 2, 3].map(&:to_s) # => ["1", "2", "3"]
[3, 4.5].inject(&:*) # => 60
```

Advanced Examples

```ruby
File.open('nginx.log', 'r') do |file|
  while line = file.gets
    puts line
  end
end
```

```ruby
# -- possible implementation --
class File
  def open(file, option)
    if block_given?
      f = fopen(file, option)
      yield f
      f.close
    end
  end
end
```

```ruby
ActiveRecord::Base.transaction do
  # -- code omitted --
end

# -- possible implementation --
module ActiveRecord
  class Base
    def transaction
      # -- start transaction
      yield
      # -- commit transaction
      rescue ActiveRecord::Rollback
      # -- roll back transaction
      ensure
      # -- close transaction
    end
  end
end
```

The End