Языки Интернет-программирования

Лекция 4. Ruby: Классы, Модули, Примеси

- Особенности передачи параметров в методы
- Объектная модель
- Организация модулей и примеси
- Отладка и тестирование

МГТУ им. Н.Э. Баумана, доц. каф. ИУ-6,
к.т.н. Самарев Роман Станиславович

samarev@acm.org

Особенности передачи параметров в методы 1/4

24.09.2019

- Simple attributes:

```
def calculate_value(x,y)
  x + y
end
```

- Default value:

```
def some_method(value='default', arr=[])
  puts value
  puts arr.length
end

some_method('something')
```

- https://en.wikibooks.org/wiki/Ruby_Programming/Syntax/Method_Calls

Особенности передачи параметров в методы 2/4

24.09.2019

- Variable Length Argument List

```
def calculate_value(x, y, *otherValues)
   puts otherValues
end

calculate_value(1, 2, 'a', 'b', 'c')
```

- Asterisk Operator (*)

```
arr = ['a', 'b', 'c']

calculate_value(*arr)  # calculate_value('a', 'b', 'c')
```

- https://en.wikibooks.org/wiki/Ruby_Programming/Syntax/Method_Calls

Особенности передачи параметров в методы 3/4

24.09.2019

- Hash Arguments

```
def foo(options = {})
  puts "#{options[:bar]} #{options[:buz]}"
end

foo(bar: 'bar', buz: 'buz')      # => 'bar buz‘
```

- Default values with hash arguments

```
def foo(options = {})
  options = {bar: 'bar'}.merge(options)
  puts "#{options[:bar]} #{options[:buz]}"
end

foo(buz: 'buz')            # => 'bar buz'
```

- http://ruby-doc.com/docs/ProgrammingRuby/html/tut_methods.html
- http://brainspec.com/blog/2012/10/08/keyword-arguments-ruby-2-0/

Особенности передачи параметров в методы 4/4

24.09.2019

- Keyword arguments in Ruby 2.0

```
def foo(str: "foo", num: 424242)
  [str, num]
end

foo(str: 'buz', num: 9) # => ['buz', 9]
foo(str: 'bar')         # => ['bar', 424242]
foo                     # => ['foo', 424242]
foo(bar: 'buz')         # => ArgumentError
```

- keyword arguments and options as a hash

```
def foo(str: "foo", num: 424242, **options)
  [str, num, options]
end

foo                 # => ['foo', 424242, {}]
foo(check: true) # => ['foo', 424242, {check: true}]
```

- http://ruby-doc.com/docs/ProgrammingRuby/html/tut_methods.html
- http://brainspec.com/blog/2012/10/08/keyword-arguments-ruby-2-0/

Safe Navigation Operator (&.)

24.09.2019

- Традиционный способ проверки вложений

```
account = Account.find(id)
if account && account.owner && account.owner.address
…
end

if account.try(:owner).try(:address)
…
end
```

- Ruby 2.3.0

```
if account&.owner&.address
…
end
```

- http://mitrev.net/ruby/2015/11/13/the-operator-in-ruby/

Классы

- Базовый класс Object
- То, что явно не указано, относится к Object
- Любые данные есть объект
- Имя любого класса есть константа, указывающая на экземпляр класса Class

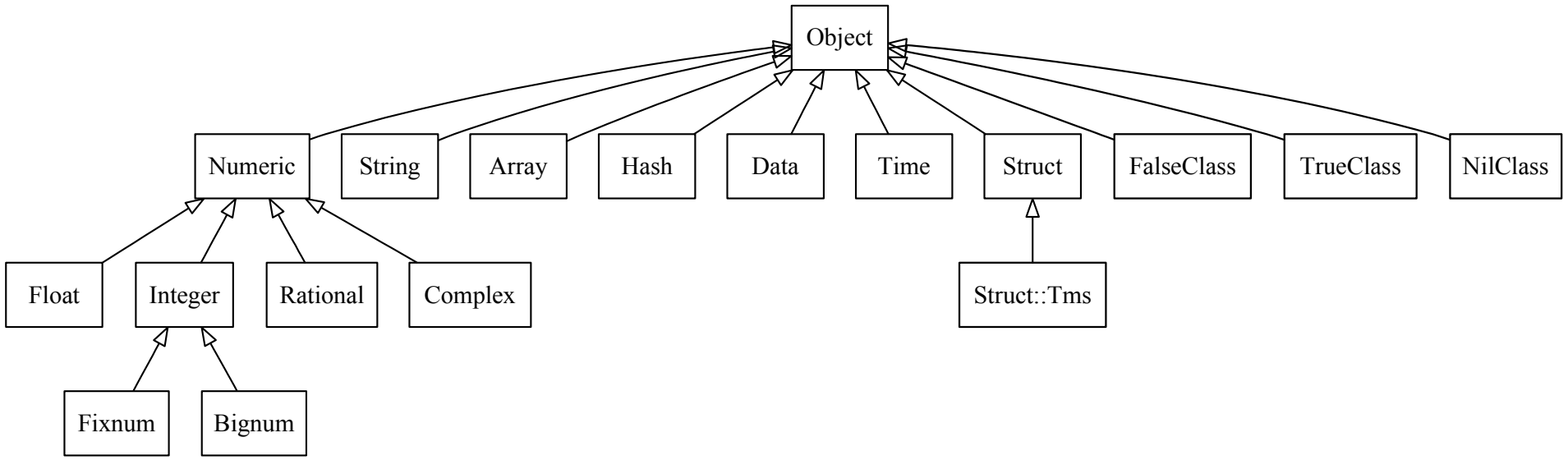
Именование переменных

- Переменные класса @@name_for_class
- Переменные экземпляра @name_for_inst
- Константы CONST_NAME или Const_name

Система типов

24.09.2019

```
anc_desc = {}
ObjectSpace.each_object(Class).select {|x| x < Object}.each {|c| anc_desc[c.name] = c.superclass.name}
File.open('result.dot', 'w') do |file|
        file.puts %Q(digraph "Ruby #{RUBY_VERSION}" {\n)
        file.puts %Q(node [shape=box];\n edge [arrowtail="empty", dir=back];\n)
         anc_desc.each.select {|k, v| k && v}
            .sort_by {|desc, anc| anc+desc}
            .each {|desc, anc| file.puts %Q("#{anc}" -> "#{desc}";\n)}
        #               anc_desc.each {|desc, anc|  file.puts %Q("#{anc}" -> "#{desc}";\n)}
        file.puts '}';
end
system "dot -Tsvg result.dot -o ruby.svg"
```



Пояснения к программе

24.09.2019

```
anc_desc = {}
p classes = ObjectSpace.each_object(Class) # все объекты Class
p children_cl = classes.select { |x| x < Object } # только потомки Object
# формируем хэш имя класса – имя предка
children_cl.each { |c| anc_desc[c.name] = c.superclass.name }
p anc_desc # выводим полученный хэш

File.open('result.dot', 'w') do |file| # формируем файл для graphviz
    file.puts %Q(digraph "Ruby #{RUBY_VERSION}" {\n)
    file.puts %Q(node [shape=box];\n edge [arrowtail="empty", dir=back];\n)
    anc_desc.each.select {|k, v| k && v}
                .sort_by{|desc, anc| anc + desc}.each do |desc, anc|
       file.puts %Q("#{anc}" -> "#{desc}";\n)
    end
    #           anc_desc.each {|desc, anc|  file.puts %Q("#{anc}" ->
    "#{desc}";\n)}   # сортируем, если это необходимо
    file.puts '}';
end
system 'dot -Tsvg result.dot -o ruby.svg' # генерируем svg
```

Конструкторы

24.09.2019

```
class ColoredRectangle
    def initialize(r, g, b, s1, s2)  # вызывается автоматически из new
     @r, @g, @b, @s1, @s2 = r, g, b, s1, s2
    end

    def self.white_rect(s1, s2) # альтернативный “конструктор”
     new(0xff, 0xff, 0xff, s1, s2)
    end
    def ColoredRectangle.red_square(s)   # альтернативный “конструктор”
     new(0xff, 0, 0, s, s)
    end

    def inspect; "#{@r} #{@g} #{@b} #{@s1} #{@s2}" end
end
a = ColoredRectangle.new(0x88, 0xaa, 0xff, 20, 30)
b = ColoredRectangle.white_rect(15, 25)
c = ColoredRectangle.red_square(40)
p a, b, c
```

• Атрибуты

- Не обязательно декларировать явно

```
class Person
    def name
     @name                  # возвращает значение @name
    end

    def name=(x)
     @name = x              # инициализирует @name
    end

    def age
     @age                   # возвращает @age
    end
    # ...
end
```

Атрибуты
Специальные методы

24.09.2019

- Методы генерации атрибутов и методов доступа

attr (aSymbol, writable=false) – чтение [и запись]
attr_reader([aSymbol]+) – метод для чтения
attr_writer([aSymbol]+) – метод для записи
attr_accessor([aSymbol]+) – reader/writer

```
class Person
    attr :name, true # Создаются @name, name, name=
    attr :age        # Создаются @age, age
end
```

Атрибуты класса
Пример использования

24.09.2019

```
class Metal
    @@current_temp = 70 # переменная, общая для всех экземпляров!
    attr_accessor :atomic_number

    def Metal.current_temp=(x)
     @@current_temp = x
    end

    def Metal.current_temp
     @@current_temp
    end

    def liquid?
     @@current_temp >= @melting
    end

    def initialize(atnum, melt)
     @atomic_number = atnum  # атомный номер элемента
     @melting = melt                          # температура плавления
    end
end
```

Атрибуты класса
Продолжение примера

24.09.2019

```
 # создаём 3 объекта класса Metal
aluminum = Metal.new(13, 1236)
copper = Metal.new(29, 1982)
gold = Metal.new(79, 1948)
# устанавливаем общую температуру
Metal.current_temp = 1600
# смотрим, кто расплавился
puts aluminum.liquid?  # true
puts copper.liquid?    # false
puts gold.liquid?      # false
# повышаем общую температуру
Metal.current_temp = 2100
# смотрим, кто теперь расплавился
puts aluminum.liquid?  # true
puts copper.liquid?    # true
puts gold.liquid?      # true
```

Классы
Пространство объектов

24.09.2019

```
class Person
  attr_accessor :fname, :lname
  def initialize(fname, lname)
    @fname = fname
    @lname = lname
  end
  def to_s
    @lname + ', ' + @fname
  end
  def self.find_by_fname(fname)
    # получаем список всех объектов указанного класса
    ObjectSpace.each_object(Person) { |o|  return o if o.fname == fname }
    nil
  end
end
```

Классы
Пространство объектов

24.09.2019

```
Person.new('Yukihiro', 'Matsumoto')
Person.new('David', 'Thomas')
Person.new('David', 'Black')
Person.new('Bruce', 'Tate')

# Find matz!
puts Person.find_by_fname('Yukihiro')
```

- Внимание! Пример поиска не эффективен из-за полного перебора each_object!

Классы
Struct

24.09.2019

- Класс для конструирования классов без методов

```
Student = Struct.new(:name, :group)

stud = Student.new( 'Иванов И.И.', 'ИУ6-11' )

puts Student.class          # -> Class
puts stud.class             # -> Student
puts "Студент: #{stud.name}\tГруппа: #{stud.group}"
        #->Студент: Иванов И.И.    Группа: ИУ6-11
```

- http://www.ruby-doc.org/core-2.5.0/Struct.html

Наследование

24.09.2019

```
class Person
    attr_accessor :name, :age, :gender
    def initialize(name, age, gender)
      @name, @age, @ gender = name, age, gender
    end
    # ...
end
```

- **Только одиночное наследование!**

```
class Student < Person
    attr_accessor :idnum, :hours
    def initialize(name, age, gender, idnum, hours)
      super(name, age, gender)
      @idnum = idnum
      @hours = hours
    end
    # ...
end
```

Информация о классе
Доступ к метаданным

24.09.2019

```
s = 'Hello'
n = 237
sc = s.class          # String
nc = n.class          # Fixnum

n = 9876543356210
p n.instance_of? Bignum     # true
p n.instance_of? Integer    # false
p n.kind_of? Bignum         # true
p n.kind_of? Integer        # true
p n.is_a? Bignum            # true
p n.is_a? Integer           # true
p n.is_a? Numeric           # true
p n.is_a? Object            # true
p n.is_a? String            # false
p n.is_a? Array             # false
```

```
# Проверка отношений
#                наследования классов
p Integer < Numeric         # true
p Integer < Object          # true
p Object == Array           # false
p IO >= File                # true

p Float < Integer           # nil
```

#kind_of? #is_a? – true => объект является экземпляром указанного класса или его предков

#instance_of? – true => объект является экземпляром строго указанно класса

Управление доступом

- public – общедоступные (по-умолчанию)
- protected – только для класса и потомков
- private – только для методов класса

Управление доступом

24.09.2019

```
class Bank
   # определяем методы
   def open_safe
    # ...
   end
   def close_safe
    # ...
   end
   # а теперь делаем их закрытыми
   private :open_safe, :close_safe

   def make_withdrawal(amount)
    if access_allowed
     open_safe
     get_cash(amount)
     close_safe
    end
   end

   # Остальные методы закрытые
      private
       def get_cash
        # ...
       end
       def access_allowed
        # ...
       end
end


bank = Bank.new
bank.make_withdrawal (100000)
```

Управление доступом


```
class Person
    def initialize(name, age)
      @name, @age = name, age
    end
    def <=>(other)
      age <=> other.age
    end
    attr_reader :name, :age  # определим атрибуты
    protected :age           # а теперь закроем возраст
end

p1 = Person.new('fred', 31)
p2 = Person.new('agnes', 43)
compare = (p1 <=> p2)              # -1
x = p1.age                         # Ошибка!
```

«Замораживание» объекта методом freeze

24.09.2019

```
str = 'Тест '
str.freeze
begin
  str << 'не пройден! ' # Попытка модифицировать.
rescue => err
  puts "#{err.class} #{err}"
end

arr = [1, 2, 3]
arr.freeze
begin
  arr << 4 # Попытка модифицировать.
rescue => err
  puts "#{err.class} #{err}"
end

# TypeError: can't modify frozen string
# TypeError: can't modify frozen array
```

Некорректное «замораживание»


```
p str = 'counter-'
str.freeze
p str += 'intuitive' # "counter-intuitive"


p arr = [8, 6, 7]
arr.freeze
p arr += [5, 3, 0, 9] # [8, 6, 7, 5, 3, 0, 9]
```

метод += создаёт новый объект!

Константы и «замороженные объекты»

24.09.2019

```
Str = 'test'
Str = 'test2'          # => warning: already initialized constant
p Str.gsub! '2', '*'  # => "test*"
```

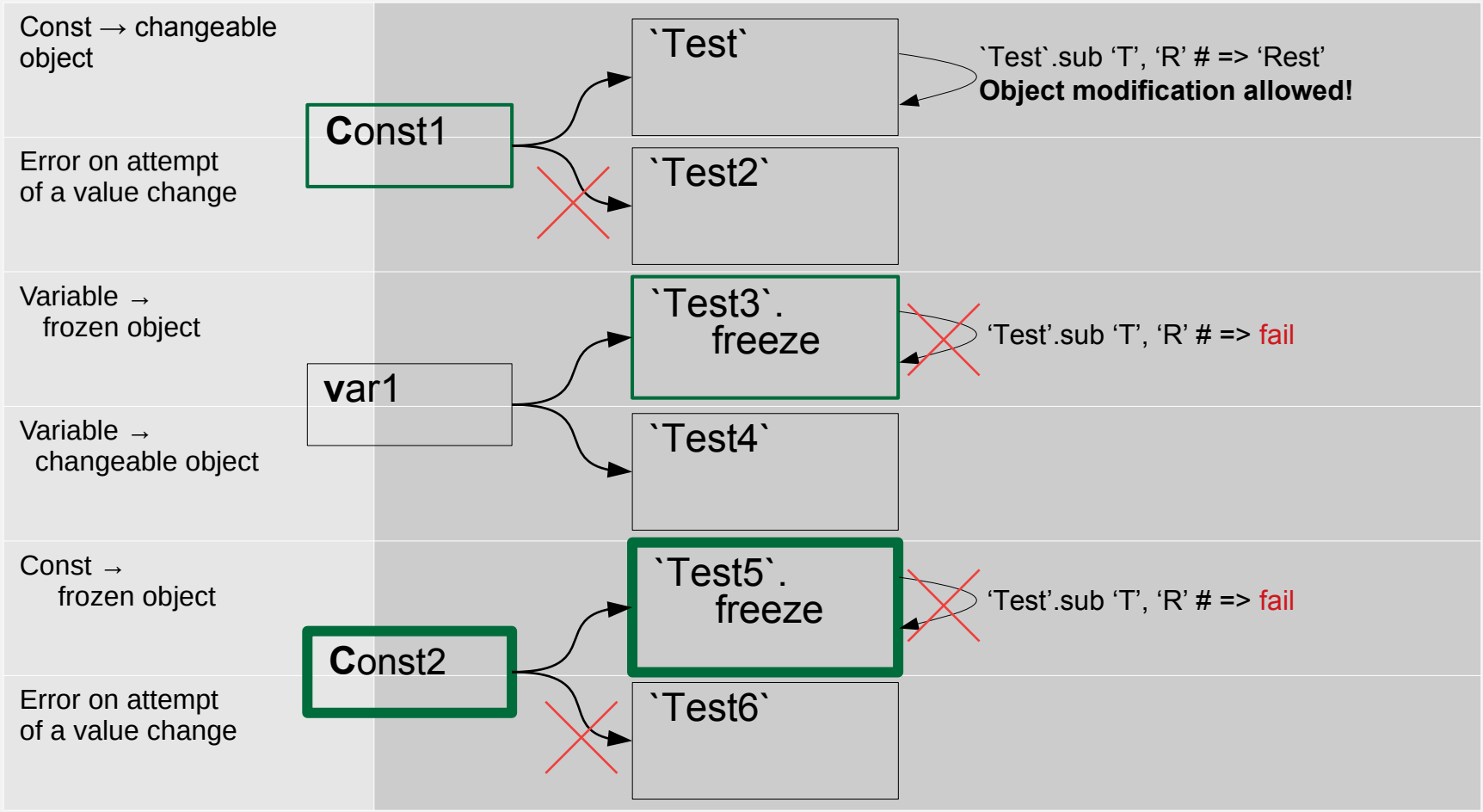
- Константа – постоянная ссылка на объект (но без контроля изменения объекта)

```
Str.freeze
p Str.chomp! '*' # => can't modify frozen String (RuntimeError)
```

- На «замороженный объект» может ссылаться любая переменная, включая константы

Constants, variables, freeze...

24.09.2019



Распределённое объявление и переопределение классов

24.09.2019

```
puts Time.now # => 2012-09-12 21:37:34 +0400
```

- Переопределим в своём файле метод Time#to_s

```
class Time
    # код класса находится в стандартной библиотеке!
    def to_s
            strftime '%H:%M:%S %d.%m.%Y'
    end
end

p Time.now       # Time.now().inspect -> 2012-09-12 21:45:08 +0400
puts Time.now    # Time.now().to_s  -> 21:45:08 12.09.2012
```

- **“monkey patch” – бездумное динамическое изменение кода!**

Переопределение методов экземпляров

24.09.2019

```
t1 = Time.now()
t2 = t1.clone # полное копирование объекта

class << t1
    def to_s
        strftime '%H:%M:%S %d.%m.%Y'
    end
end

puts t1 # 21:49:49 12.09.2012
puts t2 # 2012-09-12 21:49:49 +0400
```

Видимость атрибутов

24.09.2019

```
@str = 'Hello!' # атрибут экземпляра
def test_print
  puts @str         # проверяем доступность
  @@str2 = 'test' # атрибут уровня класса
end
# запускаем метод
test_print

# метод для фильтрации вывода
def filter_output ar
   #выводим только известные нам имена
   filter = [ '@str', '@@str2', 'test_print', "#{__method__}"]

   ar.map { |x| x.to_s }.each do |text|
         puts "found #{text}" if filter.include? text
   end
end
```

```
---------- Class Object ----------------
found @@str2
found test_print
found filter_output
---------- object ----------------
found test_print
found filter_output
---------- object ----------------
found @str
found test_print
found filter_output
```

```
# печатаем атрибуты и методы класса Object
puts '---------- Class Object ----------------'
filter_output Object.class_variables
filter_output Object.methods
filter_output Object.private_methods

# получаем все объекты-потомки Object
ObjectSpace.each_object(Object) do |o|
  # исключаем потомков и оставляем только
  # экземпляры Object
  if o.instance_of? Object
    puts '---------- object ----------------'
    # выводим атрибуты и методы экземпляра
    filter_output o.instance_variables +
                  o.methods +
                  o.private_methods
  end
end
```

Определение методов на русском языке

24.09.2019

```
#coding: utf-8
class Numeric
    def квадрат # добавляем метод
       self * self
    end
end

# любое число теперь имеет этот метод
puts 10.квадрат
puts 20.квадрат
puts 3.14.квадрат
puts Math::PI.квадрат
```

Модули
Класс Module

24.09.2019

- Разграничивают пространство имён
- Обеспечивают возможность использование примесей
- Не может иметь экземпляры!
- Может содержать методы, константы и классы
- Не имеет наследования

```
module Mod
  include Math
  CONST = 1
  def meth
    #  ...
  end
end
puts Mod.class                  # => Module
puts Mod.constants              # => [:CONST, :PI, :E]
puts Mod.instance_methods       # => [:meth]
puts Mod.const_get(:PI)         # => 3.141592653589793
```

- http://www.ruby-doc.org/core-2.4.0/Module.html

Примеси (mixins)

24.09.2019

- Механизм, позволяющий реализовать аналог множественного наследования

```
module Sum # объявляем модули Sum и Mul
    def sum;   inject  { |s, element|  s + element }  end
end
module Mul;
    def mul;    inject { |s, element|  s * element }  end
end

class Array # добавляем стандартному классу наши модули!
  include Sum
  include Mul
end

# вызываем новый метод для нового массива
[1, 2, 3, 4, 5].sum #=> 15
[1, 2, 3, 4, 5].mul  #=> 120
```

- http://rails.vsevteme.ru/2009/02/28/samorazvitie/5-metaprogramming-patterns-18-kyu-primesi

Примеси

24.09.2019

```
# coding: utf-8
require 'set'
# расширяем библиотечный модуль Enumerable
module Enumerable
  def sum
    inject { |m, element|  m + element }
  end
end
# и тогда мы получаем sum для всех контейнеров:
puts Set[5, 3, 1].sum                # => 9
puts (('a'..'z').sum)                # => 'abcdefghijklmnopqrstuvwxyz'
puts ({ 1 => 'a', 2 => 'b'}.sum)  # => [1, "a", 2, "b"]
```

Enumerator vs Enumerable

24.09.2019

- Enumerator – класс
  (может быть создан как объект)
  - Методы:
    #each, #next, #next_values,
    #peek, #peek_values, #take….

- Enumerable – модуль
  (примешивается к другим классам)
  - Методы:
    #all?, #any?, #chunk, #collect, #map, #count, #cycle, #entries,
    #detect, #find, #find_all, #find_index, #first, #reduce, #inject,
    #sort….

class Enumerator
Пример использования

24.09.2019

```
strings = ObjectSpace.each_object(Numeric)
# strings содержит объект Enumerator

# обходим все значения
begin
   while (str = strings.next)
      puts str
   end
rescue StopIteration => e
   p 'result: ' + e.to_s
end

# То же, но короче
strings.each { |i| puts i }
```

class Enumerator
Создание нового объекта

24.09.2019

```
fib = Enumerator.new do |y|
  a = b = 1
  loop do
    y << a # y.yield a
    a, b = b, a + b
  end
  puts 'end' # Цикл бесконечный. Никогда не выводится!
end

p fib.take(10) # => [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]

test_enum = Enumerator.new do |y|
  y << 1
  y << 2
  y << 3
end
test_enum.each { |i| puts i } # => 1 2 3
```

- http://www.ruby-doc.org/core-2.5.0/Enumerator.html

class Enumerator
метод take_while

24.09.2019

- вычисления суммы ряда 1/1+1/2+1/3+1/4... с заданной точностью:

```
list = Enumerator.new do |yielder|
  sum, prev, counter = 0.0, 1.0, 1.0
  loop do
    yielder.yield sum, prev, counter

    prev = sum
    sum += 1.0 / counter
    counter += 1
  end
end

puts '*' * 80

puts list.take_while { |sum, prev| (prev - sum).abs > 1e-4 }
```

module Enumerable
Пример использования


```
class Fib
  include Enumerable
  def each
    a = b = 1
    loop do
      yield a
      a, b = b, a + b
    end
  end
end

p Fib.new.find { |i| i > 10 } # найти первое число больше 10 => 13
p Fib.new.take(10) # => [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

- Модуль Enumerable включен в Array, Hash, String

- http://ruby-doc.org/core-2.5.0/Enumerable.html

Динамическое выполнение кода

- Метод Kernel#eval

```
str = 'puts 123'
eval str # 123

str = '1 + 2 * 3'
puts eval str # 7

str = 'def a; 1 + 2 * 3 end'
eval str
puts a() # 7
```

- http://www.ruby-doc.org/core-2.5.0/Kernel.html#method-i-eval

Выполнение кода из внешнего файла

24.09.2019

- Файл ext.rb

```
def print_ext
  puts "Hi!"
end
puts 'This is external module'
```

- Файл main.rb

```
# вариант 1
code = File.read 'ext.rb'
eval code
print_ext

#This is external module
#Hi!

# вариант 2
Thread.start do
   $SAFE = 4 # ограничивает уровень выполнения программы
   load('ext.rb', true)
end
```

- http://www.ruby-doc.org/docs/ProgrammingRuby/html/taint.html

Тестирование
Построение дерева

24.09.2019

```
class Tree
  attr_accessor :left, :right, :data
  protected :left, :right, :data

  def initialize(x = nil)
    @left, @right, @data = nil, nil, x
  end

  def insert(x)
    if @data.nil?
      @data = x
    elsif x <= @data
      @left ? @left.insert(x) : @left = Tree.new(x)
    else
      @right ? @right.insert(x) : @right = Tree.new(x)
    end
  end
```

Тестирование
Продолжение

24.09.2019

```
 def inorder
    @left.inorder { |y| yield y } unless @left.nil?
    yield @data
    @right.inorder { |y| yield y } unless @right.nil?
  end

  def inbackorder
    @right.inbackorder { |y| yield y } unless @right.nil?
    yield @data
    @left.inbackorder { |y| yield y } unless @left.nil?
  end
end
```

Тестирование с помощью тестовых наборов

24.09.2019

```
require './tree.rb'

items = [35, 1, 24, 2, -4, 3, 25, 4, 94, 5, 0, 6, 14, 7]
# или items = gets.split.map(&:to_i)
# или items = Array.new(14){ rand(100)-50 }

tree = Tree.new

items.each { |x| tree.insert(x) }

puts tree.inorder { |x| print "#{x} "}
puts tree.inbackorder { |x| print "#{x} "}
```

Тестирование
Класс MiniTest::Unit::TestCase

24.09.2019

- Структура теста

```
require 'minitest/autorun'                    # устаревшая альтернатива - 'test/unit'
class Test… < MiniTest::Unit::TestCase        # ранее Test::Unit::TestCase

  def setup # вызывается перед выполнением каждого теста
  end

  def test_name1 # тест1
    assert_equal( val1, val2 )
  end
  …
  def test_nameN # тестN
    assert_equal( val1, val2 )
  end

  def teardown # вызывается после выполнения каждого теста
  end
end
```

Тестирование
TestTree Unit TestCase

24.09.2019

```
require './tree.rb'
require 'minitest/autorun'

class TestTree < MiniTest::Unit::TestCase
  def setup # вызывается перед выполнением каждого теста
    #@items = [35, 1, 24, 2, -4, 3, 25, 4, 94, 5, 0, 6, 14, 7]
    @items = Array.new(100) { rand(200) – 100 }
    #@items = 100.times.map { Random.rand(200) - 100 }

    @tree = Tree.new
    @items.each { |x| @tree.insert(x) }
    @result = [ ]
  end

  def test_1
    @tree.inorder { |x| print "#{x} "; @result << x } # формируем результат
    assert_equal(@items.sort, @result)               # сравниваем с эталоном
  end

  def test_2
    @tree.inbackorder { |x| print "#{x} "; @result << x } # формируем результат
    assert_equal(@items.sort_by { |x| -x }, @result )     # сравниваем с эталоном
  end
end
```

• Minitest::Test Утверждения

24.09.2019

assert(test, message = nil) refute(test, message = nil)	True if boolean
assert_equal(expected, actual, [message]) refute_equal(expected, actual, [message])	True if expected == actual
assert_match(pattern, string, [message]) refute_match(pattern, string, [message])	True if string =~ pattern
assert_nil(object, [message]) refute_nil(object, [message])	True if object == nil
assert_in_delta(expected_float, actual_float, delta, [message])	True if (actual_float - expected_float).abs <= delta
assert_instance_of(class, object, [message])	True if object.class == class
assert_raise(Exception,...) {block}	True if the block raises (or doesn't) one of the listed exceptions.

- http://docs.seattlerb.org/minitest/Minitest/Assertions.html

Отладка Ruby-программ

24.09.2019

- Отладочная печать сообщений **p, puts**
- Интерактивный ruby – **irb**
- **pry** - An IRB alternative and runtime developer console

- Интегрированная среда разработки
  - Eclipse IDE + Ruby DLTK
  - Atom IDE
  - Sublime text
  - Netbeans
  - KDevelop
  - JetBrains RubyMine
  - ...

Отладка Ruby-программ
pry

24.09.2019

- An IRB alternative and runtime developer console
- gem install pry

```
# test.rb
require 'pry'

class A
  def hello() puts "hello world!" end
end

a = A.new

# start a REPL session ( от англ. read-eval-print loop — цикл «чтение — вычисление — вывод»)
binding.pry      # Точка останова и ожидания команд консоли
# program resumes here (after pry session)
puts "program resumes here."
```

- https://github.com/pry/pry

Статическая проверка кода *«RuboCop is an analyzer and formatter»*

24.09.2019

- https://github.com/bbatsov/rubocop

- Установка:
  `gem install rubocop`

- Использование:
  `rubocop ruby_file.rb`
  - проверить и выдать отчёт

  или
  `rubocop -a ruby_file.rb`
  - автоматически применить стилистические исправления

- https://github.com/bbatsov/ruby-style-guide

Статическая проверка кода «*Reek - Code smell detector for Ruby*»

24.09.2019

- https://github.com/troessner/reek
- Установка:
  `gem install reek`

- Использование:
  `reek ruby_file.rb`
  - проверить и выдать отчёт

  или
  `reek --no-documentation ruby_file.rb`


```ruby
# Smelly class
class Smelly
  # UncommunicativeMethodName: Smelly#x has the name 'x'
  def x
    y = 10 # Uncommunicative: Smelly#x has the variable name 'y'
  end
end
```

Ruby Version Manager (RVM)

- https://www.rvm.io/
- Менеджер версий Ruby
  - Установка нескольких версий в пространстве пользователя
  - MRI/YARV (ruby) / JRuby (jruby) / Rubinius (rbx)
  - Переключение между версиями
  - Возможность работы нескольких пользователей с разными версиями одновременно

```
rvm install 2.6.2
rmv list
rvm use
...
```

Литература

24.09.2019

- Основы языка программирования Ruby : учебное пособие / Р. С. Самарев. — Москва : Издательство МГТУ им. Н. Э. Баумана, 2015. — 98, [2] с. : ил.
- Фултон Х. Программирование на языке Ruby.–М.:ДМК Пресс, 2007.-688 с.:ил.
- Д. Флэнаган, Ю. Мацумото. Язык программирования Ruby.–СПб.; Питер, 2011
- D. Thomas, C.Fowler, A. Hunt. Programming Ruby 1.9 & 2.0. The Pragmatic Programmers' Guide. (The Facets of Ruby) 4th Edition - Texas.Dallas: The Pragmatic Programmers, 2013 .- 888 p.

- http://ru.wikibooks.org/wiki/Ruby
- http://en.wikibooks.org/wiki/Ruby_Programming