Конспект RubySchool.us [2]

Урок 15

Атрибуты (свойства класса):

```
attr_reader
attr_writer
attr_accessor
```

```
class Some
  attr_writer :foo
end

bar = Some.new
bar.foo = 12
puts bar.foo
```

```
class Song
  attr_accessor :name, :duration

  def initialize name, duration
    @name = name
    @duration = duration
  end
end

song1 = Song.new 'Obla-di, obla-da', 6
puts song1.name
```

```
class Airport
  attr_reader :name

  def initialize name
    @name = name
  end
end

class Airplane
  attr_reader :model

  def initialize model
    @model = model
  end
end
```

1 аэропорт содержит много самолётов (*):

Задача Аэропорты и самолёты в них:

```
# 1. создать 2 аэропорта с 3 самолётами в каждом
# 2. вывести на экран название аэропортов и какие в нём самолёты

class Airport
  attr_reader :name, :planes

  def initialize(name)
    @name = name
    @planes = []
  end
```

```
  def add_plane plane
    @planes << plane
  end
end

class Plane
  attr_reader :model

  def initialize(model)
    @model = model
  end
end

# массив для хранения аэропортов
airports = []

# создадим 2 аэропорта
airport1 = Airport.new "SVO"
airport2 = Airport.new "VKO"

# запишем их в массив аэропортов
airports << airport1
airports << airport2

# add plane in airport1
plane1 = Plane.new "Airbus"
plane2 = Plane.new "Boeing"
plane3 = Plane.new "IL-76"

airport1.add_plane plane1
airport1.add_plane plane2
airport1.add_plane plane3

# add plane in airport2
plane2_1 = Plane.new "DoooDdoo"
plane2_2 = Plane.new "Foofo"
plane2_3 = Plane.new "FafoooFf"
plane2_4 = Plane.new "Boooooou"

airport2.add_plane plane2_1
airport2.add_plane plane2_2
airport2.add_plane plane2_3
airport2.add_plane plane2_4

# выведем информацию об аэропортах и самолётах в них:
airports.each do |airport|
  puts "#{airport.name}"

  puts "Planes in this airport:"
  airport.planes.each do |plane|
    puts "#{plane.model}"
  end

  puts "========================="
end
```

Вариант задачи Аэропорт-Самолёты со Страной:

```
class Country
  attr_reader :name, :airports

  def initialize(name)
    @name = name
    @airports = []
  end
```

```
  end

  def add_airport airport
    @airports << airport
  end
end

class Airport
  attr_reader :name, :planes

  def initialize(name)
    @name = name
    @planes = []
  end

  def add_plane plane
    @planes << plane
  end
end

class Plane
  attr_reader :model

  def initialize(model)
    @model = model
  end
end

# массив стран
countries = []

# создадим 2 страны
russia = Country.new "Russia"
germany = Country.new "Germany"

# добавим в массив стран
countries << russia
countries << germany

# создадим 2 аэропорта
airport1 = Airport.new "MOSCOW"
airport2 = Airport.new "BERLIN"

# добавим аэропорты в страну
russia.add_airport airport1
germany.add_airport airport2

# add plane in airport1
plane1 = Plane.new "Airbus"
plane2 = Plane.new "Boeing"
plane3 = Plane.new "IL-76"

airport1.add_plane plane1
airport1.add_plane plane2
airport1.add_plane plane3

# add plane in airport2
plane2_1 = Plane.new "DoooDdoo"
plane2_2 = Plane.new "Foofo"
plane2_3 = Plane.new "FafoooFf"
plane2_4 = Plane.new "Booooou"

airport2.add_plane plane2_1
airport2.add_plane plane2_2
airport2.add_plane plane2_3
airport2.add_plane plane2_4
```

```
# выведем информацию о странах, аэропортах и самолётах в них:

countries.each do |country|
  puts "Airports From #{country.name}:"
  country.airports.each do |airport|
    puts "- Airport #{airport.name}"

      puts "-- Planes in this airport:"
      airport.planes.each do |plane|
        puts "-- #{plane.model}"
      end
  end
  puts "=================="
end
```

Задача: альбом с песнями

```
# add album and 3 song

class Album
  attr_reader :name, :songs

  def initialize(name)
    @name = name
    @songs = []
  end

  def add_song song
    @songs << song
  end

end

class Song
  attr_reader :name, :duration

  def initialize(name, duration)
    @name = name
    @duration = duration
  end

end

# add massive with albums
albums = []

abba = Album.new 'Abba'

albums << abba

song1 = Song.new "Foooo Barrrr!", "1:44"
song2 = Song.new "Baz Baz Baz!", "4:30"
song3 = Song.new "Yahooou!", "2:32"

abba.add_song song1
abba.add_song song2
abba.add_song song3

albums.each do |album|
  puts "Album: #{album.name}", ""
  album.songs.each do |song|
    puts "Song: #{song.name} -- duration: #{song.duration}"
  end
end
```

Наследование (inheritance):

```
class Parent
  def foo
    puts "foo"
  end
end

class Child < Parent
end
```

Наследование от суперкласса метода initialize - через super:

```
class Animal
  def initialize(name)
    @name = name
  end

  def jump
    puts "#{@name} is jumping..."
  end
end

class Cat < Animal

  def initialize
    super 'Murzik'
  end

  def say_meow
    puts "#{@name} Meowww!"
  end
end

class Dog < Animal

  def initialize
    super 'Sharik'
  end

  def say_gav
    puts "#{@name} Gavvv!"
  end
end

cat = Cat.new
dog = Dog.new

cat.jump
cat.say_meow

dog.jump
dog.say_gav
```

> изучи ссылку Classes, Objects, and Variables

Урок 16

Существует 3 уровня доступа к методам. Методы бывают:

- открытый (public) интерфейс — общий интерфейс для всех пользователей данного класса;
- защищённый (protected) интерфейс — внутренний интерфейс для всех наследников данного класса;
- закрытый (private) интерфейс — интерфейс, доступный только изнутри данного класса.

class API

Private - можно вызывать только из методов класса, но не отдельно (ограничивает доступ только внутри класса):

```
class Animal

  attr_reader :name

  def initialize(name)
    @name = name
  end

  def jump
    eat
    puts "#{name}: I am jumping..."
    sleep
  end

  private

  def eat
    puts "#{name}: I am eating..."
  end

  def sleep
    puts "I am sleeping..."
  end
end

cat = Animal.new "Murzik"
cat.jump
# cat.eat # будет ошибка
# cat.sleep # тоже ошибка
```

Статические методы - self

```
class Man
  def say_hi
    puts "Hi!"
  end
end

man = Man.new
man.say_hi

# or

class Man
  def self.say_hi # !!!
    puts "Hi!"
  end
end

Man.say_hi
```

yield - вызов

```
def run_5_times
  5.times do
    yield
  end
end

run_5_times { puts "Foooo!!!" }
```

Можно передать параметр:

```
def run_5_times
  x = 0
  while x < 5
    yield x
    x += 1
  end
end

run_5_times { |i| puts "Foo! Index: #{i}" }
```

Можно передать несколько параметров:

```
def run_5_times
  x = 0
  while x < 5
    yield x, 55
    x += 1
  end
end

run_5_times { |i, v| puts "Foo! Index: #{i}. Value: #{v}" }
```

Лямбда (lambda) - это указатель на функцию:

```
x = lambda { #some code }

x = lambda do
  #some code
end

x.call # вызов
```

Параметры в lambda-выражениях:

```
x = lambda { |a| puts "#{a}" }

x.call 55
```

```
x = lambda { |a, b| puts "#{a+b}" }

x.call 3, 22
```

Работа с массивами:

```
say_hi = lambda { puts "Hi!" }
say_bye = lambda { puts "Bye!" }

week = [ say_hi, say_hi, say_hi, say_hi, say_hi, say_bye, say_bye ]

week.each do |f|
  f.call
end
```

return, но можно не писать:

```
sub_10 = lambda do |x|
  return x - 10
end

a = sub_10.call 1000
puts a
```

```
sub_10 = lambda { |x| return x - 10 }
```

lambda и работа с хешем:

```
# прибавляет 10
# прибавляет 20
# отнимает 5

add_10 = lambda { |x| x+10 }
add_20 = lambda { |x| x+20 }
sub_5 = lambda { |x| x-5 }

# если до 300 - прибавлять 10
# если до 600 - прибавлять 20
# если больше 600 - вычитать 5

balance = 1000

hh = { 111 => add_10, 222 => add_10, 333 => add_20, 444 => add_20, 555 => add_20, 666 => sub_5, 777 => sub_5 }

while true
  x = rand(100..999)
  puts "Combination: #{x}"

  if hh[x]
    f = hh[x]
    balance = f.call balance
    puts "Lambda called."
  else
    balance = sub_5.call balance
  end

  puts "Balance: #{balance}"
  puts "Press Enter to continue..."
  gets
end
```

Module - модули

> module = namespace = пространство имён

```
module Foo
  class Bar
    #somecode
  end
end

baz = Foo::Bar.new
```

Пример:

```
module Humans

  class Manager
    def say_hi
      puts "Hello!"
    end
  end

  class Hipster
    def say_hi
      puts "Yooooouuuu! Hi!"
    end
  end

  class Alisha
    def say_hi
      puts "Nihau!"
    end
  end

end

hipster = Humans::Hipster.new
hipster.say_hi
```

Встраивание модуля в класс

```
class Animal
  def say
    # some code
  end
end

class Cat
  include Animal
  def eat
    #some code
  end
end
```

Mixins

```
module A
   def a1
   end
   def a2
   end
end
module B
   def b1
   end
   def b2
   end
end

class Sample
include A
include B
   def s1
   end
end

samp = Sample.new
samp.a1
samp.a2
samp.b1
samp.b2
samp.s1
```

Подключение модуля из другого файла:

```
require './foo.rb'
```

Типы переменных:

```
CONSTANT = 3.14 # константа
$global_var # глобальная переменная
@instance_var # переменная экземпляра
@@class_var # переменная класса (сохраняет своё значение во всех экземплярах класса)
local_var # переменная доступная внутри метода
```

```
class Song
  @@plays = 0

  def play
    @@plays += 1
  end

  def total_plays
    puts @@plays
  end
end

song1 = Song.new
song2 = Song.new
song3 = Song.new

song1.play
song2.play
song3.play
song3.total_plays
```

Урок 17

Метапрограммирование

Функция send - отправить в какую-то функцию какие-то аргументы.

```
def mm
  #some code
end

# варианты вызова метода:
mm

# metaprogram:
send :mm
# or
send "mm"
```

Параметры в send:

```
def mm param
  puts "#{param}"
end

send :mm, 343
```

Передача хеша в send в виде параметра

```
def mm hash
  hash.each { |key, value| puts "#{key} -- #{value}"}
end

send :mm, :foo => 22, :bar => 44
```

```
class Foo
  attr_accessor :name

  def initialize
    send("name=", "Mike") # аналог @name = "Mike"
  end
end

bar = Foo.new
puts bar.name
```

```
class Foo
  attr_accessor :x, :y

  def initialize hash
    hash.each do |key, value|
      send "#{key}=", value
    end
  end

end

s = Foo.new :x => 1, :y => 5

puts s.x
puts s.y
```

```
class Foo

  attr_accessor :name, :age, :country

  def initialize hash
    hash.each do |key, value|
      send("#{key}=", value)
    end
  end

end

bar = Foo.new :name => "Mike", :age => 60, :country => "USA"

p bar
```

method_missing - метод о несуществовании метода

```
class Foo
  def method_missing name
    puts "Метода #{name} не существует"
  end
end

bar = Foo.new
bar.fjsdahuefd
```

Ещё пример:

```
class Albu
  def initialize(actions)
    @actions = actions
  end

  def method_missing name
    value = @actions[name]
    puts "If you want to #{name}, you must call #{value}."
  end
end

a = Albu.new :cook => "Joe", :take_a_ride => "Jessie", :die => "Gus"
a.cook
a.take_a_ride
a.die
```

> ИЗУЧИ ССЫЛКУ http://www.tutorialspoint.com/ruby/ruby_modules.htm

Немного хитростей и приёмов:

Лежит ли число в определённом диапазоне чисел:

```
a = 12
a.between?(10, 22)
```

Зацикленный вывод:

```
a = ["a", "b", "c"]
a.cycle { |x| puts x }
```

ruby - разбить длинную строку можно указав обратный слэш:

```
puts ' ... ' \
' ... '
```

> rubymine - заменить имя переменной сразу везде: выделить и нажать shift+f6

Обнаружение в массиве (метод detect - первый, select - подмассив найденных):

```
arr = [1, 2, 4, 6, 12, 2]

arr.detect { |x| x == 4 }
arr.detect { |x| x < 7 }
arr.select { |x| x < 7 }
```

Метод .map

```
arr.map { |x| x*2 }
```

define_method

```
send :define_method, "aaa" do
  puts "Hello, I`m new method"
end

aaa

# аналог
# def aaa
#   ...
# end
```

```
print "Name of method to define: "
method_name = gets.strip

send :define_method, method_name do
  puts "Hello, I`m new method"
end

send method_name
```

> Преимущество такого подхода в том, что метод вызывается по его имени.

```
def left
  puts "Robot goes left"
end

def right
  puts "Robot goes right"
end

print "Enter your method call: "
my_method = gets.strip

send my_method
```

В параметрах к методу хеш должен ставиться последним, т.к. мы не знаем количество элементов хеша.

split - строку в массив:

```
a = "January, 10, Bob, Alisha, Mike, 44"
p a.split(",")
```

Разберём текст по буквам:

```
a = "Hello How are you"
p a.split("")
```

По словам:

```
a = "Hello How are you"
p a.split(" ")
```

Количество уникальных букв:

```
a = "Hello How are you"
p a.split("").uniq.size
```

Работа с файлами:

```
# чтение из файла
input = File.open("test.txt", "r")
puts input.read


# запись в файл
output = File.open("output.txt", "w")

output.write "Foo Bar Baz!"
output.close
```

Режимы работы с файлами:

- **r** - режим **"только для чтения"**: указатель файла находится в начале файла. Это режим "по-умолчанию". Если файла не существует - выпадет ошибка.
- **w** - режим **"только для записи"**: перезаписывает файл, если файл существует. Если файл не существует, создаёт новый файл для записи.
- **r+** - режим **"чтения и записи"**: указатель файла будет в начале файла. При осуществлении записи будет затирать уже существующие строки в начале файла. Если файла не существует - выпадет ошибка.
- **w+** - режим **"чтения и записи"**: перезаписывает существующий файл, если файл существует. Если файл не существует, создаёт новый файл для чтения и записи.
- **a** - append, добавить в конец файла. Режим **"только для записи"**. Указатель файла находится в конце файла, если файл существует. То есть файл находится в режиме добавления. Если файл не существует, он создаёт новый файл для записи.
- **a+** - режим **"чтения и записи"**. Указатель файла находится в конце файла, если файл существует. Файл открывается в режиме добавления. Если файл не существует, он создаёт новый файл для чтения и записи.

Про режим a+

text.txt:

```
Foo!
Bar!
Baz!
```

app.rb

```
file = File.open("text.txt", "a+") # откроем файл для добавления и чтения
puts file.read # прочитаем содержимое файла

file.write "Karamba!" # добавим в файл

file.rewind
puts file.read # прочитаем ещё раз файл

file.close # закроем файл
```

Чтение файла строка за строкой:

```
# чтение файла строка за строкой:
input = File.open("test.txt", "r")

n = 1

while line = input.gets
  puts "#{n} #{line}"
  n += 1
end

# or
while line = input.gets
  puts line
end
```

Решение задачи с чтением файла и подсчётом суммы:

```task.txt Январь,100 Февраль,20 Март,30 Апрель,20 Май,300 Июнь,100 Июль,22 Август,50 Сентябрь,40 Октябрь,50 Ноябрь,80 Декабрь,0

```ruby
input = File.open("task.txt", "r")

total = 0

while line = input.gets
  total += line.split(",")[1].to_i
end

input.close

puts ""
puts "Total: #{total}"
```

Переименование файла:

```
File.rename("file.txt", "file_foo.txt")
```

chmod в Ruby

```
File.chmod(0644, "/foo.txt", "out")
```

```
puts Dir.pwd
```

cd в Ruby:

```
Dir.chdir '/home/alex/'
```

Массив всех каталогов и файлов по заданному пути:

```
Dir.entries '/foo/bar/'
```

> ДЗ: написать программу, которая ищет файл на диске

Урок 18

Задача с password:
```

```
 # прочитать строки из файла password.txt
input = File.open("password.txt", "r")
output = File.open("password_2.txt", "w")

# вывести на экран пароли длиной 6 символов
# записать это в отдельный файл

while (line = input.gets)
  line.strip!
  if line.size == 6
    puts line
    output.write "#{line}\n"
  end
end

input.close
output.close
```

```
 # вывести на экран пароли длиной 6 символов
# записать это в отдельный файл

input = File.open("password_2.txt", "r")

# Enter your password
print "Enter your password: "
password = gets.strip

# Your password is weak / not weak
while (line = input.gets)
  line.strip!

  if line == password
    puts "Your password is weak!"
    exit
  end
end

puts "Your password is hard!"

input.close
```

GET - получить (браузер) POST - отправить (браузер)

> Изучи ссылку: RFC HTTP http://lib.ru/WEBMASTER/rfc2068/

Использование библиотеки Net::HTTP

> Изучи ссылку: Class: Net::HTTP (Ruby 2.4.1) https://ruby-doc.org/stdlib-2.4.1/libdoc/net/http/rdoc/Net/HTTP.html

Загрузить страницу:

```
 require 'net/http'

page = Net::HTTP.get('krdprog.ru', '/index.html')
puts page
```

Вариант 2:

```
 require 'net/http'
require 'uri'

uri = URI.parse "http://krdprog.ru/index.html"
response = Net::HTTP.get uri

puts response
```

Метод post_form

```
require 'net/http'
require 'uri'

Net::HTTP.post_form URI('http://www.example.com/search.cgi'),
{ "q" => "ruby", "max" => "50" }
```

> Class: Net::HTTP (Ruby 2.4.1) post_form https://ruby-doc.org/stdlib-2.4.1/libdoc/net/http/rdoc/Net/HTTP.html#method-c-post_form

```
require 'net/http'
require 'uri'

uri = URI.parse "https://duckduckgo.com/index.html"
response = Net::HTTP.post_form uri, :x => "ruby"

puts response
```

Урок 19

Sinatra

```
gem install sinatra
```

first program:

```
require "sinatra"

get '/' do
  "Hello!"
end
```

Создадим каталог views

```
mkdir views
```

Создадим файл views/index.erb

```
touch views/index.erb
```

Поправим наш файл:

```
require "sinatra"

get '/' do
  erb :index
end
```

Пример с POST:

app.rb

```
require "sinatra"

get '/' do
  erb :index
end

post '/' do
  @login = params[:login]
  erb :index
end
```

views/index.erb

```
<h1>Foo Bar Baz</h1>

<p>You typed: <%= @login %></p>

<form action="/" method="POST">
  <input type="text" name="login">
  <input type="password" name="pass">
  <input type="submit">
</form>
```

Расширим:

app.rb

```
require "sinatra"

get '/' do
  erb :index
end

post '/' do
  @login = params[:login]
  @password = params[:pass]

  if @login == 'admin' && @password == '12345'
    erb :welcome
  else
    @if_stop = "STOP!"
    erb :index
  end
end

get '/contacts' do
  "Contacts: +7 000 000-00-00"
end
```

views/index.erb

```
<h1>Foo Bar Baz</h1>

<p>You typed: <%= @login %></p>

<p><%= @if_stop %></p>

<form action="/" method="POST">
  <input type="text" name="login">
  <input type="password" name="pass">
  <input type="submit">
</form>
```

views/welcome.erb

```
<h1>Welcome!</h1>
<p>You login in system.</p>

<p><a href="/contacts">Contacts</a></p>
```

> Отдельная область программирования scraping - парсинг информации

Продолжение конспекта: Урок 20-25 - https://github.com/krdprog/rubyschool-notes/blob/master/rubyschool-notes-03.md

Содержание конспекта: