|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.03 «Прикладная информатика»**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 7 |

**Название:**

Ruby. Классы, модули, примеси.

**Дисциплина:** Языки Интернет-программирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-34Б |  | 30.10.2022 | С. А. Рахманов |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Д. В. Малахов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2022

**Цель работы:** получение навыков программирования на языке Ruby с подключением файлов и взаимодействием с ними, работа со средствами проверки соответствия стиля программирования.

**Задание:**

Все консольные приложения Ruby следует реализовывать в виде трех

отдельных файлов:

1. основная программа;

2. программа для взаимодействия с пользователем через консоль;

3. программа для автоматического тестирования на основе MiniTest::Unit

или RSpec. Везде, где это возможно, данные для проверки должны

формироваться автоматически по правилам, указанным в задании.

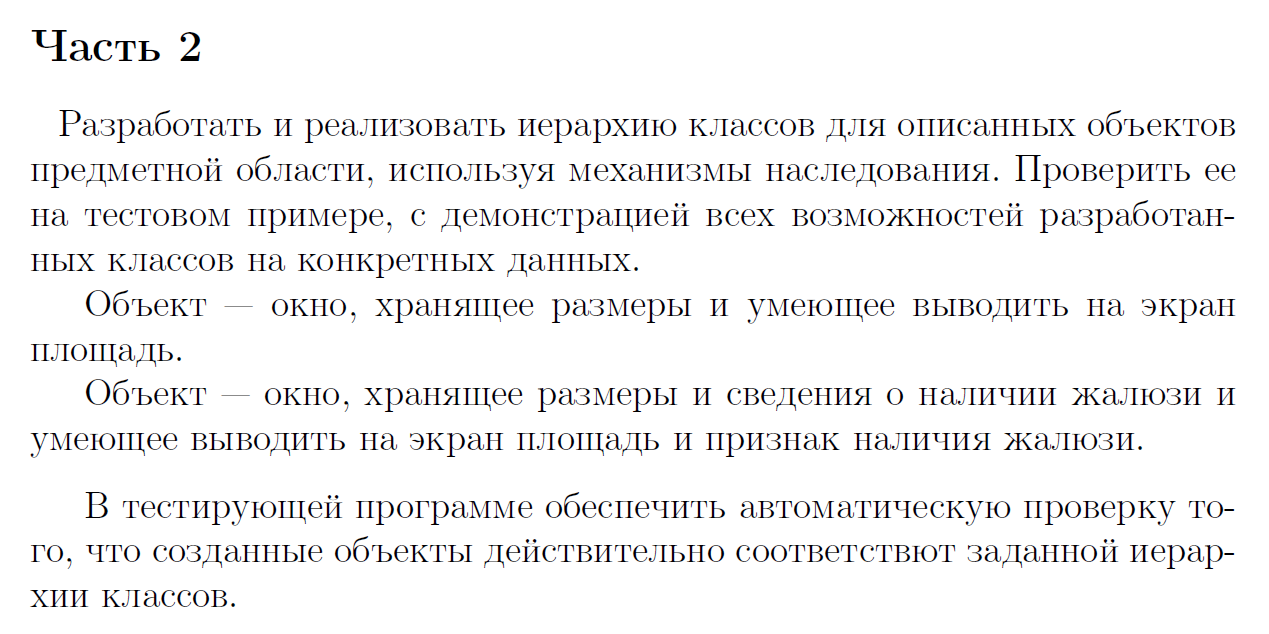
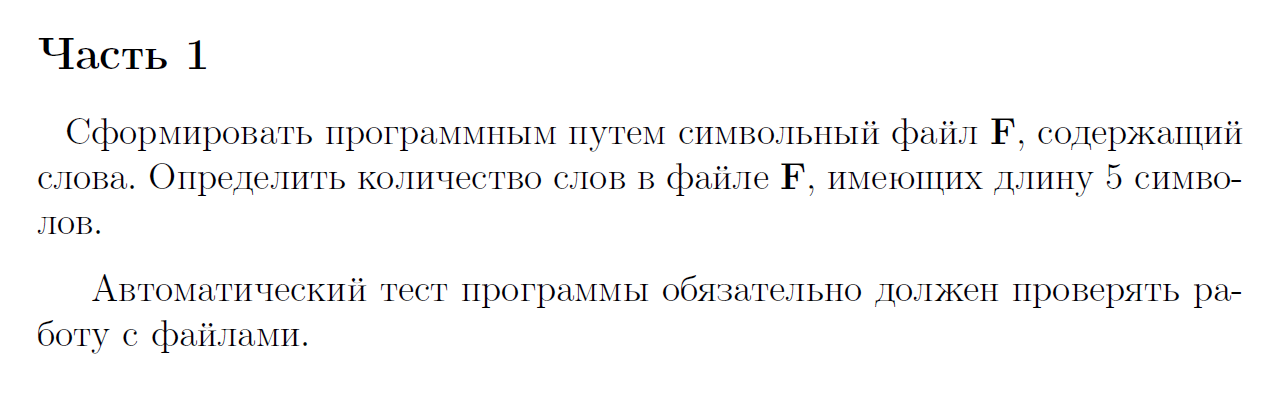
При реализации программ везде, где это возможно, следует избегать

использования циклов for, do, while. Вместо них используйте методы из

примеси Enumerable.

Все тексты программ должны быть проверены на соответствие стилю

программирования Ruby при помощи rubocop.ru или reek.



**Тексты программ**

**Часть 1**

*interface.rb*

# frozen\_string\_literal: true

require './main'

loop do

print 'Start? [y/n]'

start = gets.chop

if start == 'y'

print "words with 5 letters in file = #{calc\_words5(true)}\n"

else

exit

end

end

*main.rb*

# frozen\_string\_literal: true

def generate\_word

(1..(rand(10))).map { rand(97..122).chr }.join

end

def generate\_str

str = ''

rand(1..10).times do

str += "#{generate\_word} "

end

str += "\n"

end

def generate\_text

text = ''

rand(1..10).times do

text += generate\_str

end

text

end

def generate\_file(text)

f = File.open('F.txt', 'w')

f.write(text)

f.close

end

def calc\_words5(file\_creation)

if file\_creation

some\_text = generate\_text

generate\_file(some\_text)

end

f = File.open('F.txt', 'r')

arr = []

f.each { |str| arr << str.split } # ; p str.chop

f.close

arr.flatten!

arr.each.inject(0) { |acc, word| word.length == 5 ? acc + 1 : acc }

end

*test.rb*

# frozen\_string\_literal: true

require 'minitest/autorun'

require './main'

# Test Class

class TestTree < Minitest::Test

# first test

def test\_1

assert\_equal(true, File.exist?('F.txt'), "File don't exist")

end

# second test

def test\_2

data = ['Hello my name is John', 'Hello, how are you? Everything is okay or not',

"If no block is given, \n a new Enumerator is returned that includes the index "]

data.each do |d|

generate\_file(d)

f = File.open('F.txt', 'r')

assert\_equal(d, f.read, 'Incorrect generate of file')

end

end

# third test

def test\_3

data = ['Hello my name is John', 'Hello, how are you? Everything is okay or not',

"If no block is given, \n a new Enumerator is returned that includes the index "]

ans = [1, 0, 2]

3.times do |ind|

generate\_file(data[ind])

assert\_equal(ans[ind], calc\_words5(false), 'Incorrect words number')

end

end

end

**Часть 2**

*interface.rb*

# frozen\_string\_literal: true

require\_relative 'main'

print "Choose 'window' or 'blind' >>> "

obj = gets.chomp

if obj.downcase == 'window'

print 'Enter length and width >>> '

len, wid = gets.split(' ')

my\_window = Window.new(len.to\_i, wid.to\_i)

puts "\nSquare: #{my\_window.square}"

else

print 'Enter length, width and blindness >>> '

len, wid, bld = gets.split(' ')

my\_blind = Blind.new(len.to\_i, wid.to\_i, bld)

puts "\nSquare: #{my\_blind.square}"

puts "Is Blind: #{my\_blind.blind?}"

end

*main.rb*

# frozen\_string\_literal: true

# Class for an ordinary window

class Window

def initialize(leng, width)

@len = leng

@wid = width

end

def square

@len \* @wid

end

end

# Class for an professional window

class Blind < Window

def initialize(leng, width, bld)

@bld = true?(bld)

super(leng, width)

end

def true?(str)

str.downcase == 'true'

end

def blind?

@bld ? true : false

end

end

*test.rb*

# frozen\_string\_literal: true

require 'minitest/autorun'

require\_relative 'main'

# Class Test

class Test < Minitest::Test

def setup

@window = Window.new(10, 5)

@blind1 = Blind.new(15, 2, 'false')

end

def test\_1

assert\_equal(true, @window.is\_a?(Window))

assert\_equal(true, @blind1.is\_a?(Blind))

end

def test\_2

assert\_equal(false, @window.is\_a?(Blind))

assert\_equal(true, @blind1.is\_a?(Window))

end

def test\_3

assert\_equal(false, @blind1.instance\_of?(Window))

end

def test\_4

res = [Blind, Window, Object, Minitest::Expectations, Kernel, BasicObject]

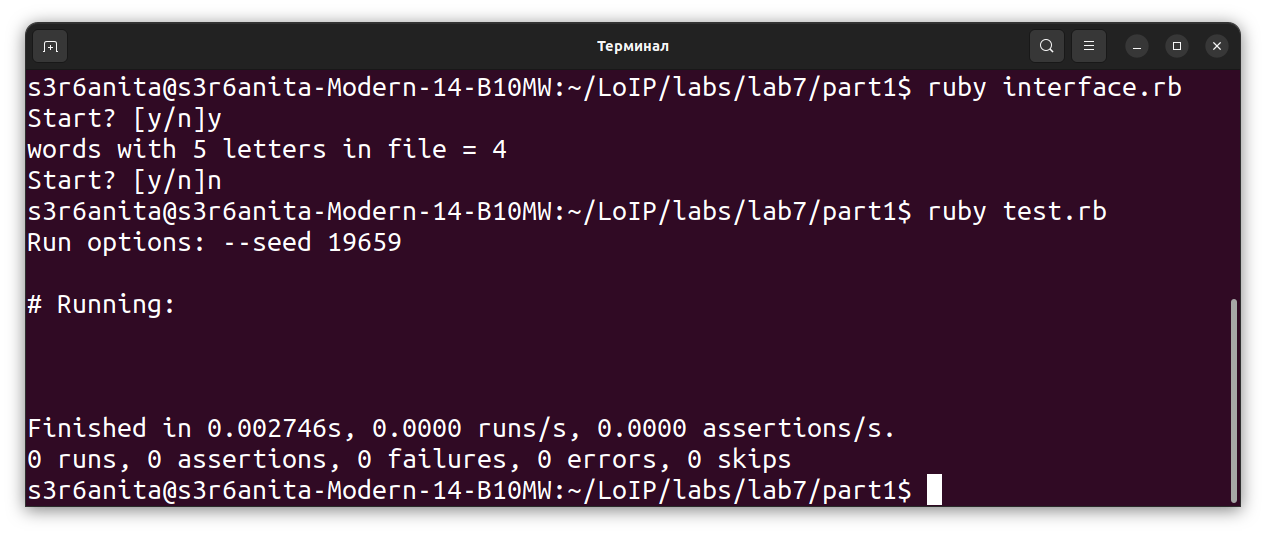
assert\_equal(res, @blind1.class.ancestors)

end

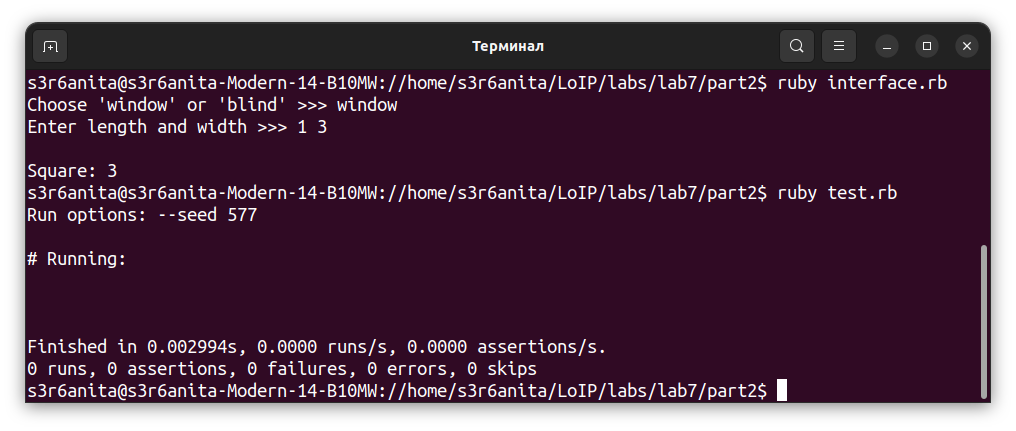
end

**Результаты выполнения**

**Часть 1**

****

**Часть 2**

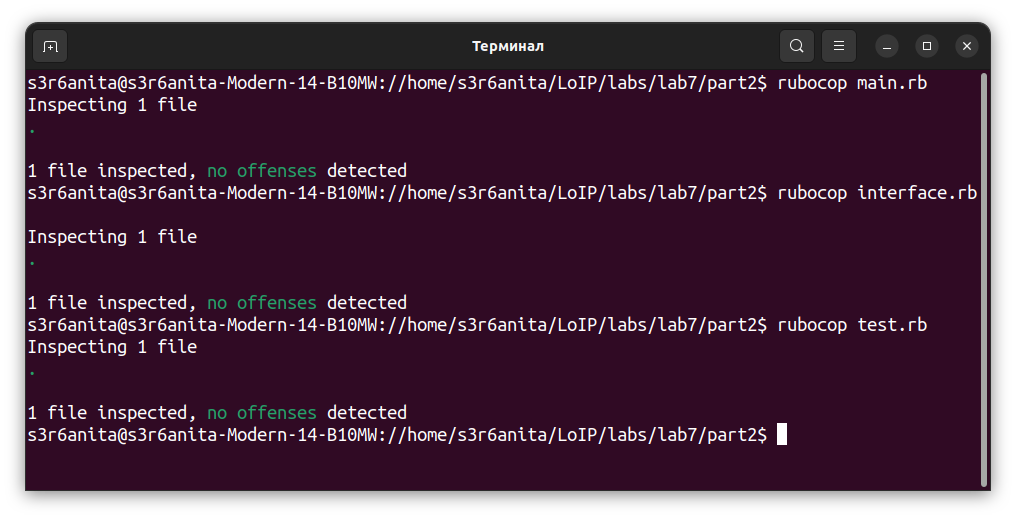
****

**Результаты проверки анализатором rubocop**

**Часть 1**

****

**Часть 2**

****

**Вывод:** получил навыки программирования на языке Ruby с подключением файлов и взаимодействием с ними, также получил опыт работы со средствами проверки соответствия стиля программирования.