**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИИ**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
**«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»  
(ФГАОУ ВО СевГУ)**

Кафедра информационных технологий и компьютерных систем

Лабораторная работа №4  
дисциплина: Вычислительные системы  
тема: «Классы и модули в Ruby».

Выполнил: ст. группы: ИВТ-17-о  
Сердюк А.А.  
Проверил: Конюхов А.Ю.

Севастополь,

2020

**Цель работы:** ознакомиться с основами работы с классами и модулями в языке Ruby.

**Постановка задачи.** Написать два скрипта:

1. Переписать банкомат из ЛЗ #3 на работу с классами. Класс должен называться - CashMachine. Программа должна запускаться с помощью метода класса init, создавать экземпляр класса
2. Пользователь запускает программу и отвечает на вопрос, с каким ресурсом он хочет взаимодействовать. После чего он может передать тип запроса GET/POST/PUT/DELETE

- GET index - должен возвращать все посты из памяти и их индекс в массиве (прим.

Hello World \n 1. Hello (again))

- GET show - должен запрашивать идентификатор поста и показывать пост по

переданному идентификатору (как в index только 1 пост)

- POST create - должен запрашивать текст поста, добавлять его в массив постов и

возвращать в ответ идентификатор поста и сам пост

- PUT update - должен запрашивать идентификатор поста, потом новый текст поста и заменить его. В результате выводить пост (как в продакшене index)

- DELETE destroy - должен запрашивать идентификатор поста, затем удалять его из массива постов

- Нужно реализовать логику для PostsController

- Добавить CommentsController самостоятельно

**Результат выполнения:**

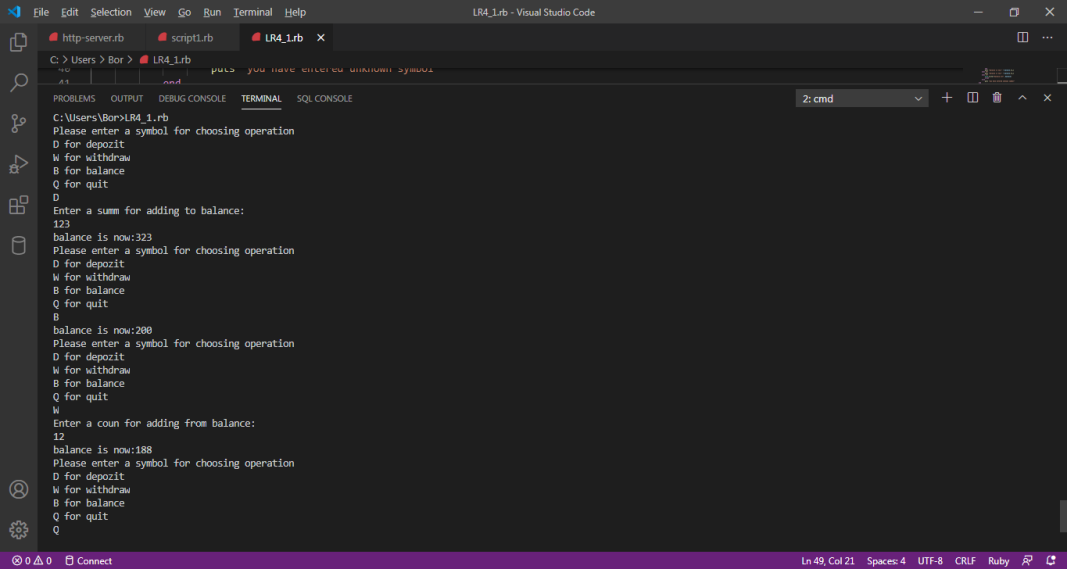


Рисунок 1 - Результат работы первого скрипта. Скрипт работает корректно.

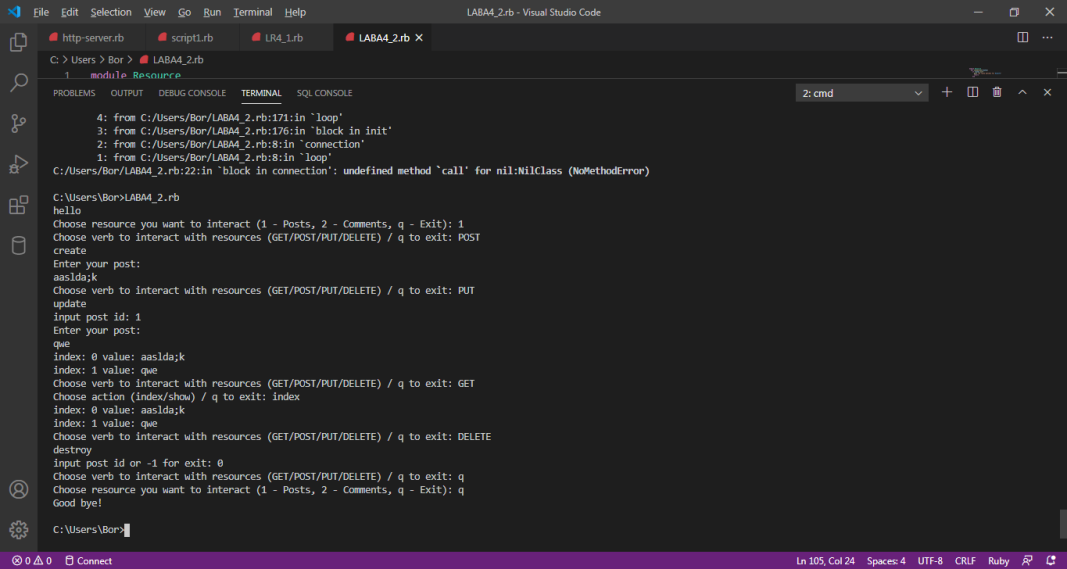


Рисунок 2 - Результат работы второго скрипта. Скрипт работает корректно.

**Код программы**

**L4(1).rd**

|  |
| --- |
| class CashMachine |
|  | # attr\_accessor :balance |
|  | # def initilalize(balance) |
|  | # @balance = balance |
|  | # end |
|  | def self.init |
|  | begin |
|  | if File.size("balance.txt")>0 |
|  | balance=File.read("balance.txt").to\_i |
|  | end |
|  | puts"Please enter a symbol for choosing operation" |
|  | puts "D for depozit" |
|  | puts "W for withdraw" |
|  | puts "B for balance" |
|  | puts "Q for quit" |
|  | $operatorF=gets.chomp.to\_s |
|  | case $operatorF |
|  | when "D" |
|  | puts "Enter a summ for adding to balance:" |
|  | depozit=gets.chomp.to\_i |
|  | if depozit>0 |
|  | balance=balance+depozit |
|  | else puts "you have entered wrong count" |
|  | end |
|  | puts "balance is now:" + balance.to\_s |
|  | when "W" |
|  | puts "Enter a coun for adding from balance:" |
|  | withdraw=gets.chomp.to\_i |
|  | if withdraw>0 and withdraw<balance |
|  | balance=balance-withdraw |
|  | else puts "balance cannot be below 0" |
|  | end |
|  | puts "balance is now:" + balance.to\_s |
|  | when "B" |
|  | puts "balance is now:" + balance.to\_s |
|  | when "Q" |
|  | File.write("balance.txt", balance) |
|  | break |
|  | else |
|  | puts "you have entered unknown symbol" |
|  | end |
|  |  |
|  | end while $operatorF |
|  | end |
|  | end |
|  |  |
|  |  |
|  | CashMachine2 = CashMachine.new(); |
|  | CashMachine.init |

**L4(2).rb**

|  |
| --- |
| module Resource |
|  | def connection(routes) |
|  | if routes.nil? |
|  | puts "No route matches for #{self}" |
|  | return |
|  | end |
|  |  |
|  | loop do |
|  | print 'Choose verb to interact with resources (GET/POST/PUT/DELETE) / q to exit: ' |
|  | verb = gets.chomp |
|  | break if verb == 'q' |
|  |  |
|  | action = nil |
|  |  |
|  | if verb == 'GET' |
|  | print 'Choose action (index/show) / q to exit: ' |
|  | action = gets.chomp |
|  | break if action == 'q' |
|  | end |
|  |  |
|  |  |
|  | action.nil? ? routes[verb].call : routes[verb][action].call |
|  | end |
|  | end |
|  | end |
|  |  |
|  | class PostsController |
|  | extend Resource |
|  | def initialize |
|  | @posts = [] |
|  | end |
|  |  |
|  | def index |
|  | if @posts.empty? |
|  | puts 'No post found' |
|  | return |
|  | end |
|  | @posts.each.with\_index{ |value, key| |
|  | puts "index: #{key} value: #{value}" |
|  | } |
|  | end |
|  |  |
|  | def show |
|  | puts 'show' |
|  | loop do |
|  | print 'enter post id: ' |
|  | i\_post = gets.to\_i |
|  | if @posts.size > i\_post && (i\_post >= 0) |
|  | puts "Post with id #{i\_post}\n#{@posts[i\_post]}" |
|  | break |
|  | end |
|  | if i\_post == -1 |
|  | break |
|  | end |
|  | puts "Error! Post with id #{i\_post} not found\nTry again or -1 for exit" |
|  | end |
|  | end |
|  |  |
|  | def create |
|  | loop do |
|  | puts "create\nEnter your post:" |
|  | p\_text = gets |
|  | if p\_text.size > 1 |
|  | @posts.push(p\_text) |
|  | break |
|  | end |
|  | puts "You not input a comment! Try again" |
|  | end |
|  | end |
|  |  |
|  | def update |
|  | puts 'update' |
|  | print 'input post id: ' |
|  | i\_post = gets.to\_i |
|  | i\_data = nil |
|  | loop do |
|  | puts 'Enter your post:' |
|  | i\_data = gets |
|  | if !i\_data.empty? |
|  | break |
|  | end |
|  | puts "You not input comment! Try again" |
|  | end |
|  | @posts[i\_post] = i\_data |
|  | self.index |
|  | end |
|  |  |
|  | def destroy |
|  | puts 'destroy' |
|  | loop do |
|  | print 'input post id or -1 for exit: ' |
|  | i\_post = gets.to\_i |
|  | if i\_post == -1 |
|  | break |
|  | end |
|  | if @posts.size > i\_post |
|  | @posts.delete\_at(i\_post) |
|  | return |
|  | else puts "This post not exist\n Enter equal post id or -1" |
|  | end |
|  | end |
|  | end |
|  | end |
|  |  |
|  | class CommentsController |
|  | extend Resource |
|  | def initialize |
|  | @comments = {} # Ключом будет id от поста, а массив комментов - собственно коммы |
|  | end |
|  |  |
|  | def index |
|  | if @comments.empty? |
|  | puts 'Comments not found' |
|  | return |
|  | end |
|  | @comments.each{ |key, value| |
|  | puts "post id: #{key}\n" |
|  | value.each.with\_index {|vl, i| |
|  | puts "id comment: #{i} comment text: #{vl}" |
|  | } |
|  | } |
|  | end |
|  |  |
|  | def show |
|  | print 'Input post id: ' |
|  | id\_post = gets.to\_i |
|  | if @comments.key?(id\_post) |
|  | @comments[id\_post].each.with\_index{|value, id| |
|  | puts "comment id: #{id} comment text: #{value}" |
|  | } |
|  | else puts "No comments found" |
|  | end |
|  | end |
|  |  |
|  | def create |
|  | print 'Input post id: ' |
|  | id\_post = gets.to\_i |
|  | puts "input comment: " |
|  | comment\_text = gets |
|  | @comments.store(id\_post, [comment\_text]) |
|  | end |
|  |  |
|  | def update |
|  | print 'Input post id: ' |
|  | id\_post = gets.to\_i |
|  | puts "Input comment:" |
|  | comment\_text = gets |
|  | if @comments.key?(id\_post) |
|  | @comments[id\_post].push(comment\_text) |
|  | else |
|  | @comments.store(id\_post, [comment\_text]) |
|  | end |
|  | self.index |
|  | end |
|  |  |
|  | def destroy |
|  | print 'Input post id: ' |
|  | id\_post = gets.to\_i |
|  | @comments.delete(id\_post) |
|  | end |
|  | end |
|  |  |
|  | class Router |
|  | def initialize |
|  | @routes = {} |
|  | end |
|  |  |
|  | def init |
|  | resources(PostsController, 'posts') |
|  | resources(CommentsController, 'comments') |
|  | loop do |
|  | print 'Choose resource you want to interact (1 - Posts, 2 - Comments, q - Exit): ' |
|  | choise = gets.chomp |
|  |  |
|  | PostsController.connection(@routes['posts']) if choise == '1' |
|  | CommentsController.connection(@routes['comments']) if choise == '2' |
|  | break if choise == 'q' |
|  | end |
|  |  |
|  | puts 'Good bye!' |
|  | end |
|  |  |
|  | def resources(klass, keyword) |
|  | controller = klass.new |
|  | @routes[keyword] = { |
|  | 'GET' => { |
|  | 'index' => controller.method(:index), |
|  | 'show' => controller.method(:show) |
|  | }, |
|  | 'POST' => controller.method(:create), |
|  | 'PUT' => controller.method(:update), |
|  | 'DELETE' => controller.method(:destroy) |
|  | } |
|  | end |
|  | end |
|  |  |
|  | load 'router.rb' |
|  | puts 'hello' |
|  |  |
|  | router = Router.new |
|  |  |
|  | router.init |

**Вывод:** В ходе проведения работы, было проведено ознакомление с базовыми возможностями языка Ruby по работе с классами и модулями.