CRIMEA DIGITAL GROUP ACADEMY

ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

ЛЗ #4 - Классы и модули

Теория

Задание

Теория

Задание

Что такое класс

Класс - это шаблон для объектов. Объекты класса также называются его экземплярами.

```
class User

  ...

end
```

Как создать экземпляр класса

Для того, чтобы создать экземпляр класса нужно для начала описать его конструктор. Конструктор может быть неявным, если нет необходимости добавлять классу какое-то состояние

```ruby
class User
end

user = User.new
# => #<User:0x00007fb44a02cbb8>
```

Как создать экземпляр класса

Но конструктор также можно описать и используя переменные экземпляра дать ему глобальное состояние.

```
class User
  def initialize(name)
    @name = name
  end
end
user = User.new("Alex")
# => #<User:0x00007fb44a18ad70 @name="Alex">
```

Как создать экземпляр класса

Свойства (или атрибуты) класса недоступны для доступа извне без объявления дополнительной директивы `:attr_reader`

```
user = User.new("Alex")
user.name
# NoMethodError (undefined method `name' for #<User:0x00007fb44a18ad70
@name="Alex">)
```

Как читать файлы

```ruby
class User
  attr_reader :name
  def initialize(name)
    @name = name
  end
end

user = User.new("Alex")
user.name
# Alex
```

Как взаимодействовать со свойствами класса

Свойства (или атрибуты) класса недоступны для модификации извне без объявления дополнительной директивы :attr_writer

```
user = User.new("Alex")
user.name
# Alex
user.name = "Jack"
# NoMethodError (undefined method `name=' for #<User:0x00007fb44a18ad70
@name="Alex">)
```

Как взаимодействовать со свойствами класса

```ruby
class User
  attr_writer :name
  def initialize(name)
    @name = name
  end
end

user = User.new("Alex")
user.name = "Jack"
# Jack
```

Как взаимодействовать со свойствами класса

Если нужно читать свойства и модифицировать их то можно использовать шорткат `:attr_accessor`

Как взаимодействовать со свойствами класса

```ruby
class User
  attr_accessor :name
  def initialize(name)
    @name = name
  end
end

user = User.new("Alex")
user.name
# Alex
user.name = "Jack"
# Jack
```

У экземпляра класса могут быть свои методы (не фунции)

Чтобы объявить метод экземпляра нужно использовать конструкцию `def method_name ... end`

В методах экземпляра класса доступны все переменные экземпляра, а также локальные переменные объявленные в нем

Как взаимодействовать со свойствами класса

```ruby
class Car
  def initialize(model, fuel, consumption)
    @model = model
    @fuel = fuel
    @consumption = consumption
  end

  def power_reserve
    @fuel / @consumption
  end
end

car = Car.new("Audi", 100.0, 7.0)
car.power_reserve
# 14.285714285714286
```

Методы класса

Иногда нужно объявить метод класса. Когда экземпляр не нужен и класс несет в себе сугубо утилитарную функцию или если вам нужно дополнительные

```
class App
   def self.init
      # I'm like main function in other languages
   end
end

App.init
```

Наследование

В руби нет множественного наследования, но наследование от родителя есть в таком же виде как в других языках

```ruby
class Car
  def initialize(fuel, consumption)
    @fuel = fuel
    @consumption = consumption
  end

  def power_reserve
    @fuel / @consumption
  end
end
```

Наследование

```ruby
class Sedan < Car
  def initialize(fuel, consumption, actuator = 'front-wheel')
    @actuator = actuator
    super(fuel, consumption)
  end
end

class OffRoad < Car
  def initialize(fuel, consumption, actuator = 'all-wheel')
    @actuator = actuator
    super(fuel, consumption)
  end
end
```

Наследование

```ruby
off_road = OffRoad.new(120, 7)
#<OffRoad:0x00007fb44a1931f0 @actuator="all-wheel", @fuel=120, @consumption=7>
off_road.power_reserve
# => 17

sedan = Sedan.new(50, 6, 'rear-wheel')
#<Sedan:0x00007fb44a17a6a0 @actuator="rear-wheel", @fuel=50, @consumption=6>
sedan.power_reserve
# => 8
```

Методы класса

Для того чтобы обеспечить переиспользование методов, которые нельзя отнести к какой либо одной абстракции (то есть не подойдет использование наследования). Нужно использовать модули.

Модуль - это миксин, которых можно подключить в класс и использовать методы модуля, как методы экземпляра или как методы класса. В зависимости от выбранного типа подключения

Есть 3, но мы сфокусируемся на двух: `include` и `extend`

Методы класса

```ruby
module Resource
  def connection
    puts "Self is a #{self}"
  end
end

class PostsController
  include Resource
end

controller = PostsController.new
controller.connection
# Self is a #<PostsController:0x00007fb44a04f1b8>
```

Методы класса

```ruby
module Resource
  def connection
    puts "Self is a #{self}"
  end
end
class PostsController
  extend Resource
end

PostsController.connection
# => Self is a PostsController
PostsController.new.connection
# NoMethodError (undefined method `connection' for
#<PostsController:0x00007f946d181ec0>)
```

Методы класса

Иногда надо получить ссылку на метод, не вызвав его. Для того, чтобы отложить для использования в будущем.

```
class PostsController
  extend Resource

  def show(id)
    puts "ID is #{id}"
  end
end

controller = PostsController.new
method = controller.method(:show)
method.call(1)
```

Работа с файлами и папками

Для лабораторной работы также понадобится набор утилит, позволяющих сравнивать файлы. Для этого необходимо в начале файла подключить `gem "fileutils"` и использовать метод `compare_file` из класса `FileUtils`

```
require 'fileutils'

FileUtils.compare_file("a.txt", "b.txt")
```

Теория

Задание

SOFTWARE ENGINEERING

Задание #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 и взаимодействовать с пользователем согласно условиям задачи

Задание #2

Используйте гист

https://gist.github.com/AlxGolubev/b30af07fe3a0add200d1c693ac64133f

SOFTWARE ENGINEERING

Задание #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. `0. 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 экшене `index`)
  - `DELETE destroy` - должен запрашивать идентификатор поста, затем удалять его из массива постов

SOFTWARE ENGINEERING

Задание #2

- Нужно реализовать логику для `PostsController`
- Добавить `CommentsController` самостоятельно

В отчете по заданию расписать, понимание работы класса `Router`, причины, по которым используется `extend` для модуля `Resource`

В случае неправильного ввода (команды), ваша программа должна выдать соответствующее сообщение об ошибке, которое говорит клиенту, как ее исправить. Нельзя просто выводить `"Error!"` - это не поможет.

КОНТАКТЫ

Для быстрого входа
на сайт

courses@crimeadigital.ru

tel: 8 (800) 551 44 86

https://crimeadigital.ru