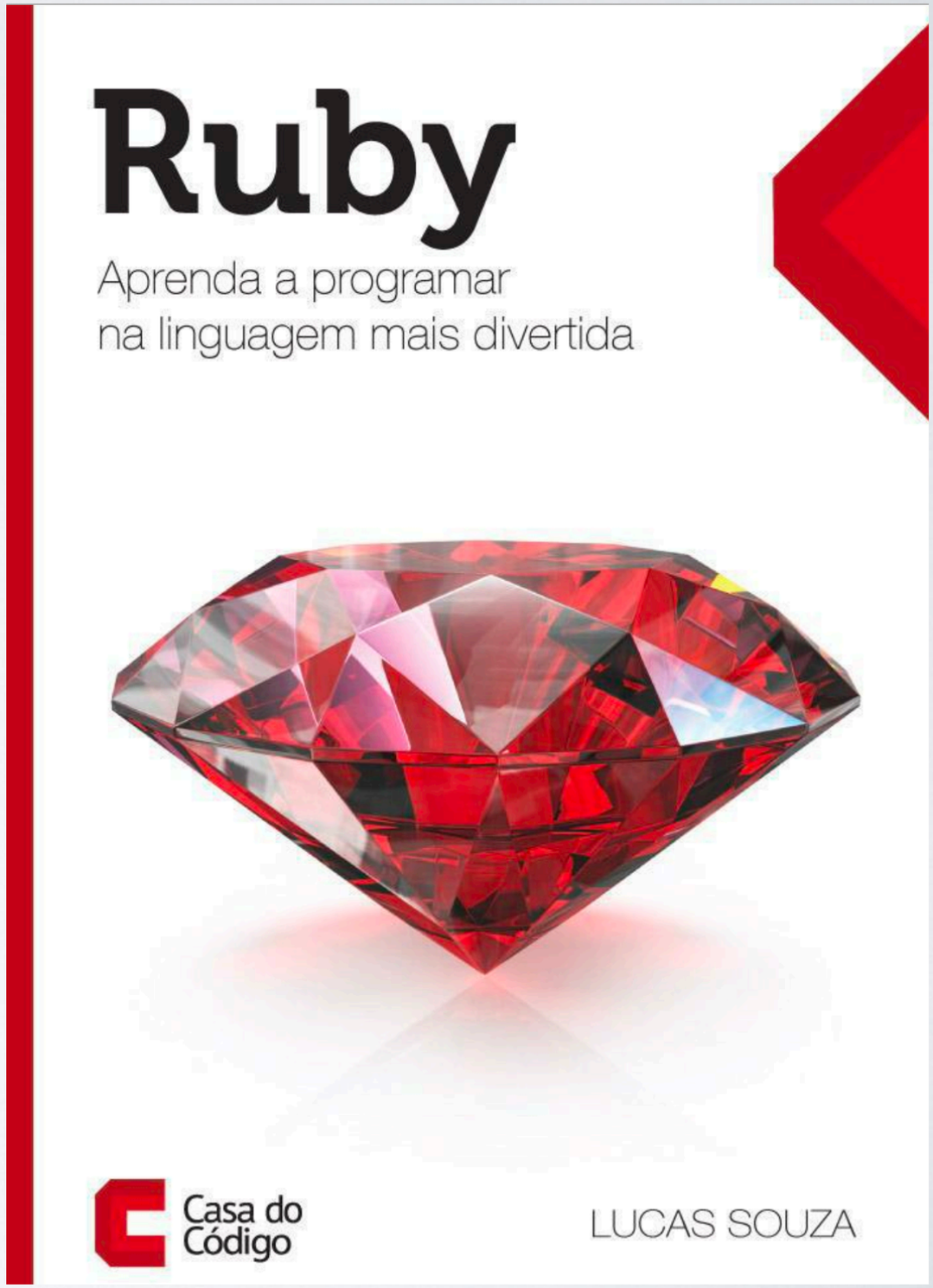
PROGRAMAÇÃO ORIENTADA A OBJETOS

Rafael Vieira Coelho

(rafaelvc2@gmail.com)



DEFININDO MÉTODOS

- Precisamos usar a palavra reservada **def**

```ruby
def hello(name)
  "Hello, #{name}"
end
```

- E no momento de chamar a função:

```ruby
hello("World")
# => "Hello, World"
```

PARÂMETROS PADRÃO

- Podemos ter valores padrão para quando não são passados:

```ruby
def make_animal_sound(sound = 'Cuack')
    puts sound
end
```

```ruby
make_animal_sound('Mooo') # Mooo
make_animal_sound         # Cuack
```

PARÂMETROS PADRÃO

- Podemos ter valores padrão para quando não são passados:

```
def make_animal_sound(sound = 'Cuack', volume = 11)
    play_sound(sound, volume)
end

make_animal_sound('Mooo') # Spinal Tap cow

make_animal_sound(volume: 1) # Duck whisper
```

NÚMERO VARIÁVEL DE PARÂMETROS

- Quando não sabemos quantos valores são, usamos o * antes do nome do parâmetro:

```ruby
def welcome_guests(*guests)
    guests.each { |guest| puts "Welcome #{guest}!" }
end
```

```ruby
welcome_guests('Tom')     # Welcome Tom!
welcome_guests('Rob', 'Sally', 'Lucas') # Welcome Rob!
                                        # Welcome Sally!
                                        # Welcome Lucas!
```

A APLICAÇÃO EXEMPLO

Existem vários assuntos que poderíamos abordar e neste livro construiremos uma aplicação para controlar uma loja de livros. Controlaremos o seu estoque, quais são os clientes da loja, faremos algumas vendas e guardaremos todas estas informações em disco, utilizando a API de arquivos do Ruby.

CLASSE LIVRO

Para criar classes usando Ruby, basta usar a palavra reservada `class` e delimitar o final da sua classe com a palavra reservada `end`.

```
1 class Livro


2 end
```

INSTANCIAÇÃO DE OBJETOS

O que criamos foi uma abstração de um `Livro` ou seja, um *template* que dirá do que um livro é composto. A partir da classe, que é o template, precisamos criar um novo livro, onde poderemos dizer quais são suas características, como título, isbn, autor e assim por diante. Para criarmos cada um, usaremos a palavra `new`:

```
1 teste_e_design = Livro.new("Mauricio Aniche", 247,
"123454")
2 web_design_responsivo = Livro.new("Tárcio Zemel", 189,
"452565")
```

CONSTRUTOR

Quando criamos uma classe, ganhamos automaticamente um método `initialize` *default*, para que o objeto possa ter suas informações inicializadas. Como queremos passar dados para inicializar nosso objeto de uma maneira diferente, no caso passando o nome do autor, número de páginas e ISBN, devemos implementar nosso próprio método `initialize` em nossa classe `Livro`:

```
1 class Livro


2     def initialize(autor, numero_de_paginas, isbn)

3     end


4 end
```

Ruby não suporta sobrecarga de métodos para o initialize.

VALORES PADRÃO

- Caso não seja passado o número isbn, será usado 1

```
1 class Livro

2       def initialize(autor, numero_de_paginas, isbn = "1")

3       end


4 end
```

EXEMPLO COMPLETO

```ruby
class Livro

    def initialize(autor, numero_paginas, isbn="1")
        @autor = autor
        @numero_paginas = numero_paginas
        @isbn = isbn
    end

    def mostra_dados()
        puts "ISBN: #{@isbn}"
        puts "Autor: #{@autor}"
        puts "Páginas: #{@numero_paginas}"
    end
end

livro = Livro.new('Rafael Vieira Coelho', 147, '869.930872')
livro.mostra_dados()
```

```
ISBN: 869.930872
Autor: Rafael Vieira Coelho
Páginas: 147
[Finished in 0.1s]
```

CODIFICAÇÃO PADRÃO

Os arquivos que contém as classes criadas em nosso sistema, ficarão em arquivos `.rb` dentro de um diretório de sua preferência. Porém é importante ressaltar que arquivos `.rb` possuem um encoding US-ASCII por padrão. Caso seu código contenha qualquer caractere que não for compatível com o ASCII, a interpretador Ruby será finalizado e acusará o erro: `invalid multibyte char (US-ASCII)`.

Se você quiser alterar o encoding padrão do arquivo `.rb`, basta adicionar a seguinte linha do arquivo:

```
1 # encoding: utf-8
```

VISIBILIDADE DOS ATRIBUTOS

- Em Ruby, todas as variáveis de instância são criadas de forma privada (sem acesso externo)

- Se quisermos acessá-las, temos recursos da própria linguagem que veremos na sequência.

MODOS DE VER OS ATRIBUTOS

```ruby
class Livro

    def initialize(autor, numero_paginas, isbn="1")
        @autor = autor
        @numero_paginas = numero_paginas
        @isbn = isbn
    end

    def to_s
        puts "ISBN: #{@isbn}"
        puts "Autor: #{@autor}"
        puts "Páginas: #{@numero_paginas}"
    end
end

livro = Livro.new('Rafael Vieira Coelho', 147, '869.930872')

puts livro
```

```
ISBN: 869.930872
Autor: Rafael Vieira Coelho
Páginas: 147
#<Livro:0x00007fb5c6874eb0>
[Finished in 0.1s]
```

MODOS DE VER OS ATRIBUTOS

```ruby
class Livro

    def initialize(autor, numero_paginas, isbn="1")
        @autor = autor
        @numero_paginas = numero_paginas
        @isbn = isbn
    end

    def to_s
        puts "ISBN: #{@isbn}"
        puts "Autor: #{@autor}"
        puts "Páginas: #{@numero_paginas}"
    end
end

livro = Livro.new('Rafael Vieira Coelho', 147, '869.930872')

p livro

```

```
#<Livro:0x00007fcf96080e40 @autor="Rafael Vieira Coelho",
@numero_paginas=147, @isbn="869.930872">
[Finished in 0.1s]
```

GET E SET

```ruby
class Livro

    def initialize(autor, numero_paginas, isbn="1")
        @autor = autor
        @numero_paginas = numero_paginas
        @isbn = isbn
    end

    def to_s
        puts "ISBN: #{@isbn}"
        puts "Autor: #{@autor}"
        puts "Páginas: #{@numero_paginas}"
    end

    def autor
        @autor
    end

    def autor=(autor)
        @autor = autor
    end
end

livro = Livro.new('Rafael Vieira Coelho', 147, '869.930872')

puts livro.autor
livro.autor = 'Stephen King'
puts livro.autor
```

```
Rafael Vieira Coelho
Stephen King
[Finished in 0.1s]
```

EXERCÍCIO

- Crie uma classe que representa Alunos (os atributos devem ser 3 notas, nome).

- A classe deve ter um método que retorna a média final do aluno.