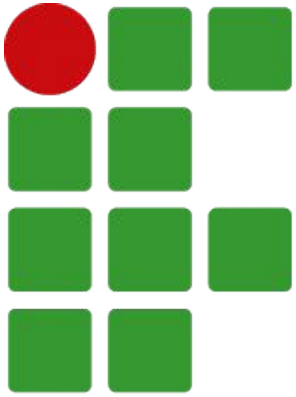
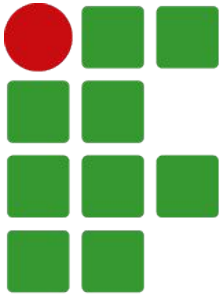


INSTITUTO FEDERAL
Paraná
Campus Paranaguá

Orientação a Objetos em Ruby

Desenvolvimento Web III

Prof. Diego Stiehl



INSTITUTO FEDERAL
Paraná
Campus Paranaguá

Classes, Métodos, Atributos, Blocos, ...

Orientação a Objetos em Ruby

INSTITUTO FEDERAL

Classe

- Para definir uma classe em Ruby
  - Palavra reservada `class`
- Pode ser feito em qualquer arquivo .rb
  - Ou no console (IRB/Pry)
- Exemplo:

```ruby
class Pessoa
  # Tudo que pertence à classe
end
```

INSTITUTO FEDERAL

Métodos

- São definidos de forma idêntica às funções
  - Porém, no corpo de uma classe
- Métodos estáticos utilizam o `self.nome`
- Exemplo:

```ruby
class Pessoa
  def dizer_ola
    puts "Olá, jovem!"
  end

  def self.dizer_quem_sou_eu
    puts "eu sou a CLASSE pessoa"
  end
end
```

INSTITUTO FEDERAL

Instanciação

- Para instanciar um objeto de uma classe
  - Utilizar método **new**
    - Parênteses opcionais, como sempre
- Exemplos:

```
pessoa = Pessoa.new
veiculo = Veiculo.new(:carro, 'Fusca', 1978)
animal = Animal.new :cavalo, :macho, 'Pé de Pano'
```

- Calma!
  - Já falaremos dos construtores

INSTITUTO FEDERAL

Classes Abertas

- Lembram do conceito de classes abertas?
- Podemos definir, a qualquer momento:
  - Um novo método para uma classe QUALQUER
    - Vale para todos os objetos que venham a ser criados daquela classe
  - Um novo método para um objeto QUALQUER
    - Vale para um objeto específico, não se repetindo nos demais
- Podemos SOBRESCREVER métodos

INSTITUTO FEDERAL

Classes Abertas - Exemplos

- Novo método em objeto

```ruby
pessoa = Pessoa.new
def pessoa.escrever_mensagem texto
  puts "A pessoa disse: #{texto}"
end
pessoa.escrever_mensagem 'Hmmmmmmmm...'
```

- Novo método em classe

```ruby
class String
  def dizer_uma_coisa coisa
    puts coisa
  end
end
'Diego'.dizer_uma_coisa 'olá'
'TESTE'.dizer_uma_coisa 'blabla'
```

INSTITUTO FEDERAL

Construtor

- Método construtor utiliza a sintaxe:

```
def initialize(param1, param2, param3)
```

- Exemplo:

```
class Pessoa
  def initialize mensagem
    puts "Pessoa foi criada e disse: #{mensagem}"
  end
end

pessoa = Pessoa.new 'Olá mundo!'
```

INSTITUTO FEDERAL

Atributos

- Atributos passam a existir quando setados
  - Sintaxe com @ (arroba)
- Exemplos:

```
@nome = 'Diego'
@idade = 29
@caes = %w{Fritz Franz Berlin}
```

- Visibilidade padrão é privada ao objeto

INSTITUTO FEDERAL

Construtor com Atributos

```ruby
class Pessoa
  def initialize nome, sobrenome, idade
    @nome = nome
    @sobrenome = sobrenome
    @idade = idade
  end
end
pessoa = Pessoa.new 'Diego', 'Maradona', 65
```

INSTITUTO FEDERAL

Getter / Setter

- Getters e setters “tradicionais” em Ruby:

```ruby
class Pessoa
  def initialize nome
    @nome = nome
  end
  def nome
    @nome
  end
  def nome=(nome)
    @nome = nome
  end
end

pessoa = Pessoa.new 'Diego'
puts pessoa.nome
pessoa.nome = 'Luiza'
```

Syntax Sugar

INSTITUTO FEDERAL

Acessores

- Forma curta de fornecer acesso aos atributos:
  - Acessores
- Sintaxe (getter/setter):

```
attr_accessor :atributo1, :atributo2, :atributo3
```

- Duas variantes:
  - `attr_reader`: acesso somente leitura
  - `attr_writer`: acesso somente escrita

INSTITUTO FEDERAL

Acessores - Exemplo

```ruby
class Pessoa
  attr_accessor :nome, :sobrenome #leitura e escrita
  attr_reader :idade              #somente leitura
  attr_writer :peso               #somnete escrita
end

pessoa = Pessoa.new

pessoa.nome = "Diego"             #OK
pessoa.sobrenome = "Stiehl"       #OK
pessoa.idade = 19                 #erro
pessoa.peso = 80.54               #OK

puts pessoa.nome                  #OK
puts pessoa.idade                 #OK (mas == nil)
puts pessoa.peso                  #erro
```

INSTITUTO FEDERAL

self

- O self referencia o próprio objeto
  - Permite acessar elementos internos
- Exemplo:

```
self.chama_outro_metodo parametro
```

INSTITUTO FEDERAL

Métodos Especiais

- Como vimos, em Ruby temos métodos com nomes especiais e funções específicas
- Exemplos de métodos comuns
  - `<<` ← Significado de uma inclusão
  - `+` ← Significado de soma
  - `-` ← Significado de subtração
- Podemos sobrescrevê-los
  - Ou escrevê-los em classe que não possuem
    - Definimos nosso próprio comportamento

INSTITUTO FEDERAL

Sobrescrevendo << para adicionar filhos a uma Pessoa

```ruby
class Pessoa
  attr_accessor :nome, :filhos, :pai
  def initialize nome
    @nome = nome
    @filhos = []
  end
  def <<(pessoa)
    @filhos << pessoa
    pessoa.pai = self
    self
  end
end

diego = Pessoa.new 'Diego'
fritz = Pessoa.new 'Fritz'
franz = Pessoa.new 'Franz'
diego << fritz << franz

for filho in diego.filhos
  puts filho.nome
end
puts franz.pai.nome
```

Modificadores de Acesso

- Opções: público, privado e protegido
- Código a partir de palavra reservada assume o modificador
- Exemplo:

```ruby
class Pessoa
 attr_accessor :nome
 def initialize nome
   @nome = nome
 end

 protected
   def metodo_qualquer
   end

 private
   def outro_metodo
   end
end
```

public (implícito)

INSTITUTO FEDERAL

Herança

- Estende o comportamento de uma classe
  - Acaba sendo subutilizada
    - Culpa das classes abertas e outros recursos que veremos mais à frente
- Sintaxe:

```
class MinhaClasse < ClasseMae
```

Herança - Exemplo

```ruby
class ArrayMelhorado < Array
  def bagunca
    reverse
  end
  def << elemento
    puts "#{elemento} adicionado no vetor"
    super
  end
end

x = ArrayMelhorado.new
x << 1 << 2 << 'AB' << 55.58
x.bagunca
x << 100
puts x.class
puts x.class.superclass
```

Método **reverse** existente em **Array#reverse**

Ao final, chama método **<<** de Array e retorna seu retorno

INSTITUTO FEDERAL

Blocks / Programação Funcional

- Block é uma técnica que permite delegação (terceirização) de trechos de código
  - Muito utilizado em Ruby
- Vêm da programação funcional
  - "Paradigma de programação que trata a computação como uma avaliação de funções matemáticas e que evita estados ou dados mutáveis"

INSTITUTO FEDERAL

Exemplo - Sem Block

- Somar o saldo total das contas de um banco

```ruby
class Banco
  def initialize(contas)
    @contas = contas
  end

  def status
    saldo = 0
    @contas.each do |conta|
      saldo += conta
    end
    saldo
  end
end
banco = Banco.new([200, 300, 400])
puts banco.status
```

INSTITUTO FEDERAL

Exemplo

- Um dia preciso que cada iteração mostre o saldo da conta

```
@contas.each do |conta|
  saldo += conta
  puts saldo # essa linha é nova
end
```

- Funciona
  - E se algum dia eu precisar mudar a saída?
    - Preciso modificar dentro da classe Conta

INSTITUTO FEDERAL

Exemplo

- Blocks permitem terceirizar a saída para quem chamou a função `status`
- Sintaxe de **chamada**:

Parâmetro “parcial” passado do método status para cá

```
puts banco.status do |saldo_parcial|
  puts saldo_parcial
end
```

Bloco

- `banco.status` segue retornando o total
  - Mas conseguimos capturar cada iteração
    - E fazer o que quiser

INSTITUTO FEDERAL

Refatorando

```ruby
class Banco
  # initialize...
  def status
    saldo = 0
    @contas.each do |conta|
      saldo += conta
      yield(saldo) if block_given?
    end
    saldo
  end
end

banco = Banco.new([200, 300, 400])
status = banco.status do |saldo_parcial|
  puts "Acumulado: #{saldo_parcial}"
end
puts status
```

Foi passado um bloco?

Chama (executa) código do bloco, passando parâmetro `saldo`

INSTITUTO FEDERAL

Exemplo

- Outra sintaxe para definição do mesmo bloco
  - Indicada se você não precisar do que mais que uma linha

```
banco.status { |saldo_parcial| puts saldo_parcial }
```

Outro Exemplo - Array de String

- Obter Array derivado, mudando strings

```
objetos = %w{cadeira mesa toalha cabide cuia}

def objetos.refaz
  novo = []
  self.each do |o|
    novo << yield(o)
  end
  novo
end

objetos_no_plural = objetos.refaz do |o|
  o + 's'
end

objetos_em_maiusculas = objetos.refaz { |o| o.upcase }

p objetos_no_plural
p objetos_em_maiusculas
```

Exemplo

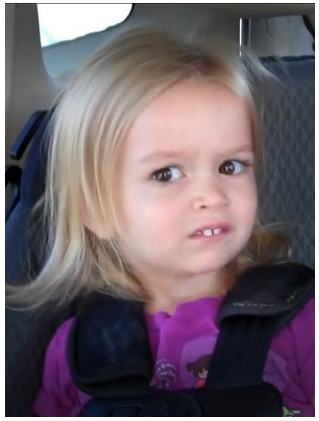
- Na verdade a classe Array já tem um método que faz exatamente isto
  - Método `map`

```
objetos = %w{cadeira mesa toalha cabide cuia}

objetos_no_plural = objetos.map do |o|
  o + 's'
end

objetos_em_maiusculas = objetos.map { |o| o.upcase }

p objetos_no_plural
p objetos_em_maiusculas
```



INSTITUTO FEDERAL

Outro Exemplo - sort

- Ordenar um vetor de Livros pelos atributos

```ruby
class Livro
  attr_accessor :titulo, :autor, :ano
  def initialize titulo, autor, ano
    @titulo = titulo
    @autor = autor
    @ano = ano
  end
end

livros = []
livros << Livro.new("Senhor dos Anéis", "Tolkien", 1964)
livros << Livro.new("Harry Potter", "Rowling", 1997)
livros << Livro.new("A Torre Negra", "King", 1973)

livros
  .sort_by { |livro| livro.titulo }
  .each { |livro| puts "#{livro.titulo} - #{livro.autor} (#{livro.ano})" }
```