Programação Orientada a Objetos II

Assunto da aula

Nesta segunda parte da aula sobre Programação Orientada a Objetos será abordado como instanciar classes localizadas em outro arquivo de um projeto, os diferentes tipos de variáveis da linguagem Ruby, como inserir e recuperar atributos e como utilizar um método construtor.

Conteúdo

- Require
- Escopo de variáveis
- Atributos
- Construtores

Require

Na **Aula 5** deste curso, você aprendeu a utilizar require 'gem_name' para carregar

arquivos de uma gem e assim poder escrever códigos com funcionalidades desta biblioteca.

Com ele também é possível carregar arquivos ruby que foram escritos por você. Para exemplificar como isso acontece você criará um projeto onde alguns arquivos conterão apenas uma classe.

Projeto

1- Crie a pasta do projeto **Animal** executando

Default

```
mkdir animal
```

2- Dentro da pasta animal crie um arquivo chamado **animal.rb** com o código:

Default

```
class Animal
 def pular
   puts 'Toing! tóim! bóim! póim!'
 end

 def dormir
   puts 'ZzZzzz!'
 end
end
```

3- Agora crie um arquivo com o nome **app.rb** e adicione a ele o código:

Default

```
require './animal.rb'

animal = Animal.new

animal.pular
```

Perceba que no **require**, você especificou o arquivo que será procurado partindo do caminho relativo a **app.rb**.

Quando você não está trabalhando com gems e quer carregar um arquivo a partir do caminho onde o código será executado, utilize o **require_relative**.

4- Substitua o código de **app.rb** por

Default

```
require_relative 'animal'

animal = Animal.new
animal.pular
```

Se o arquivo **animal.rb** estivesse dentro de uma pasta chamada example, o caminho ficaria 'example/animal'.Como só existe um arquivo chamado **animal** não é preciso especificar a sua extensão

precisa especificar a sua extensão.

5- Agora, crie um arquivo chamado **cachorro.rb** com o código:

```
Default
1 class Cachorro < Animal
2  def latir
3    puts 'Au Au'
4  end
5 end
```

A classe **Cachorro** recebe como herança a classe **Animal**.

Porque o **require_relative** de **animal.rb** não está ai?

A resposta é simples:

Você irá inicializar a classe **Cachorro** dentro de **app.rb** e esse arquivo já faz um require_relative de **animal.rb**!

6- Para incluir a inicialização e execução dos métodos da classe **Cachorro** substitua o arquivo de **app.rb** por:

```
Default
1  require_relative 'animal'
2  require_relative 'cachorro'
3
4  puts '--Animal--'
5  animal = Animal.new
6  animal.pular
7
8  puts '--Cachorro--'
9  cachorro = Cachorro.new
10 cachorro.pular
11 cachorro.latir
```

7- Execute o programa e veja o resultado

```
Default
1 ruby app.rb
```

Escopo das variáveis

Agora que você sabe dividir um programa em vários arquivos é interessante aprender sobre os diferentes tipos de escopos de variáveis presentes no Ruby. Com este conhecimento você terá a capacidade de escolher qual utilizar dependendo da situação.

As variáveis se dividem em 04 tipos:

- Variável Local
- Variável Global

- Variável Global
- Variável de Classe
- Variável de Instância

A seguir, veja características de cada uma:

Variável Local

É declarada com a primeira letra de seu nome sendo uma letra minúscula ou sublinhado.

Pode ser **acessada apenas onde foi criada**. Por exemplo, se você definir uma variável local dentro de de uma classe ela estará disponível apenas dentro desta classe, se a definiu dentro de um método conseguirá acessá-la apenas dentro deste método e assim por diante.

Exemplo:

Default

```
class Bar
 def foo
   # Pode ser definida como local ou _local
   local = 'local é acessada apenas dentro deste metodo'
   print local
 end
end

bar = Bar.new
bar.foo
```

Variável Global

Declarada com o prefixo $.

Pode ser **acessada em qualquer lugar do programa**.

Seu uso é **FORTEMENTE DESENCORAJADO** pois além de ser visível em qualquer lugar do código, também pode ser alterada em inúmeros locais ocasionando dificuldades no rastreamento de bugs.

Exemplo:

Default

```
class Bar
 def foo
   $global = 0
   puts $global
 end
end

class Baz
 def qux
   $global += 1
   puts $global
 end
```

```
end
end

bar = Bar.new
baz = Baz.new
bar.foo
baz.qux
baz.qux
puts $global
```


Variável de Classe

É declarada com o prefixo @@.

Pode ser acessada em qualquer lugar da classe onde foi declarada e

seu valor é **compartilhado** entre todas as **instâncias de sua classe**. Também
Exemplo:

Default

```
class User
 @@user_count = 0
 def add(name)
   puts "User #{name} adicionado"
   @@user_count += 1
   puts @@user_count
 end
end

first_user = User.new
first_user.add('João')

second_user = User.new
second_user.add('Mario')
```

Variável de Instância

Seu nome começa com o símbolo @.

Semelhante a variável de classe, tendo como única diferença o valor que **não é compartilhado** entre todas as **instâncias de sua classe**.

Exemplo:

Default

```
class User
 def add(name)
   @name = name
   puts "User adicionado"
   hello
 end

 def hello
   puts "Seja bem vindo, #{@name}!"
 end
end

user = User.new
user.add('João')
```

Atributos

O que são atributos?

Como você já sabe objetos possuem informações e comportamentos.

Na aula passada você viu a primeira parte deste conteúdo utilizando métodos para representar comportamentos. Agora é hora de aprender o restante adicionando e recuperando **informações** de um objeto.

Adicionando e Recuperando Informações

1- Crie um arquivo chamado **atributos.rb** com o código

Default

```ruby
class Dog
 def name
   @name
 end

 def name= name
   @name = name
 end
End

dog = Dog.new

dog.name = 'Marlon'

puts dog.name
```

O segundo recebe um valor e o atribui a variável @name.O primeiro método da classe **Dog** retorna o valor da variável de instância @name. Se a variável ainda não estiver definida, o resultado será nil.

Podemos dizer que o primeiro é para recuperar e o segundo para adicionar/alterar uma informação.

Declarar os métodos de um atributo pode ser vantajoso caso queira fazer algo além de definir o valor da variável de instância. De outra forma, existe uma maneira mais fácil de realizar esta operação.

2- Substitua o código de **atributos.rb** por

Default

```ruby
class Dog
 attr_accessor :name, :age
end


dog = Dog.new
```

```
dog = Dog.new

dog.name = 'Marlon'
puts dog.name

dog.age = '1 ano'
puts dog.age
```

O ruby disponibiliza um método chamado attr_accessor que cria os métodos var e var= para todos atributos declarados.

Contrutores

Outra questão importante é que toda vez que a instância de uma classe é criada, o Ruby procura por um método chamado initialize. Você pode criar este método para especificar valores padrões durante a construção da classe.

1- Crie um arquivo chamado **construtor.rb** com o seguinte código

Default

```
class Person
 def initialize(name, age)
   @name = name
   @age = age
 end

 def check
   puts "Instância da classe iniciada com os valores:"
   puts "Name = #{@name}"
   puts "Idade = #{@age}"
 end
end

person = Person.new('João', 12)
person.check
```

O número de parâmetros utilizados no método initialize é opcional.

2- É possível criar a instância com valores padrões do objeto e executar o método check em apenas uma instrução.

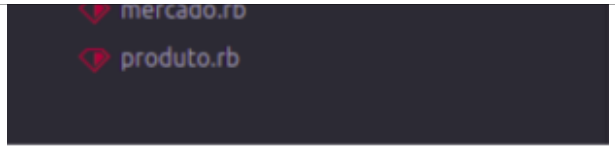
Substitua as duas últimas linhas do programa por:

Default

```
Person.new('João', 12).check
```

Missões especiais

Para exercitar o conhecimento adquirido nesta aula, crie um projeto chamado Compras com a seguinte estrutura:



Neste projeto você simulará o ato de escolher e comprar um produto em um mercado.

Instruções do projeto:

1- No arquivo **produto.rb**, crie uma classe chamada **Produto** com os atributos: **nome** e **preço**.

2- No arquivo **mercado.rb** crie uma classe chamada **Mercado** que ao ser inicializada recebe como atributo um objeto da classe Produto.

3 – Dentro da classe, crie um método chamado **comprar** que imprime a seguinte frase **"Você comprou o produto #{@produto.nome} no valor de #{@produto.preco}"**

4- No arquivo **app.rb** crie uma instância da classe **Produto** e adicione valores aos atributos **nome** e **preco**. Depois, inicie uma instância da classe **Mercado** passando um objeto produto como atributo e para finalizar execute o método comprar.

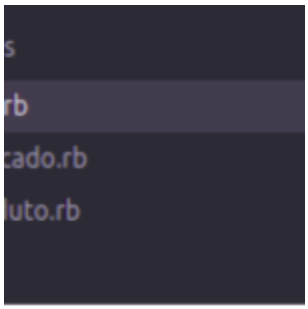
Aula passada

Código da missão

Default

```ruby
class Esportista
 def competir
   puts "Participando de uma competição"
 end
end

class JogadorDeFutebol < Esportista
 def correr
   puts "Correndo atrás da bola"
 end
end

class Maratonista < Esportista
 def correr
   puts "Percorrendo o circuito"
 end
end

esportistas = [JogadorDeFutebol.new, Maratonista.new]

esportistas.each do |esportista|
 esportista.competir
 esportista.correr
end
```

Anexos



DISCUSSION

André Costa 2 MESES AGO

——PRODUTO

```
class Product
def initialize(name, price)
@name = name
@price = price
end

attr_accessor :name, :price
end

product = Product.new(“mouse”, 123);
puts product.name
```

——MERCADO

```
class Market
def initialize(product)
@product = product
end

attr_accessor :product
```

```
def buy
puts “Você comprou o produto #{@product.name} no valor de #
{@product.price}”
end
end

——APP
require_relative “./produto.rb”
require_relative “./mercado.rb”

product = Product.new(“notebook”, 3500)
market = Market.new(product)

market.buy
```

REPLY

Fernando Galvão 10 MESES AGO

```
# produto.rb
class Product
attr_accessor :name, :price
end

# mercado.rb
class Market
def initialize(product, price)
@product = product
@price = price
end

def buy
puts “\n\tVocê comprou o produto #{@product} no valor de #{@price}”
puts “\tYou bought the #{@product} for #{@price}”
puts ”
end
end

# app.rb
require_relative ‘produto’
require_relative ‘mercado’

product = Product.new
product.name = ‘tomate’
product.price = 2.49

Market.new(product.name, product.price).buy
```

REPLY

JOAO.MOREIRA 11 MESES AGO

— App

```
require_relative ‘produto’
require_relative ‘mercado’

produto = Produto.new

puts ‘Digite o nome do produto’
produto.nome = gets.chomp

puts ‘Digite o valor do produto’
produto.preco = gets.chomp

mercado = Mercado.new(produto)
mercado.comprar
```

—mercado

```
class Mercado < Produto
def initialize(produto)
@produto = produto
end

def comprar
puts "Você comprou o produto #{@produto.nome} no valor de #
{@produto.preco}."
end
end
```

— produto

```
class Produto
attr_accessor :nome, :preco
end
```

REPLY

ALEXIS 1 ANO AGO

produto.rb

```
class Produto
attr_accessor :nome, :preco
end
```

mercado.rb

```
class Mercado
#as palavras produto e preco aqui são apenas parâmetros podendo
```

serem substituidas por quaisquer outras palavras.
def initialize(produto, preco)
@produto = produto #apenas parâmetros
@preco = preco#apenas parâmetros
end

def comprar
puts “Você comprou o produto #{@produto} no valor de #{@preco}”
end
end

app.rb
require_relative ‘produto’
require_relative ‘mercado’

produto = Produto.new

puts ‘Qual produto você vai comprar?’
produto.nome = gets.chomp

puts ‘Qual o valor do produto?’
produto.preco = gets.chomp

Mercado.new(produto.nome, produto.preco).comprar

REPLY

Vinícius Dadário de Freitas 1 ANO AGO

eu gostaria de saber se no final tem certificado?

REPLY

Thiago 1 ANO AGO

class Produto
attr_accessor :nome, :preco
end
————————————-

class Mercado
def initialize(produto)
@produto = produto
end
def comprar
puts “Você comprou o produto #{@produto.nome} no valor de #{@produto.preco}.”

```
end
end
————————————-

require_relative 'produto'
require_relative 'mercado'

p1 = Produto.new
p1.nome, p1.preco = 'produto 01', 15
m01 = Mercado.new(p1)
m01.comprar
```

REPLY

Airton Negromonte 2 ANOS AGO

Por favor alguém saberia me explicar a linha 6 do programa atributos.rb? Nessa linha está digitada assim: def name= name mas se eu digitar assim: def name = name, ou seja se o sinal de "=" estiver separado de o programa dá erro.
Não entendi o erro:
atributos.rb:8: syntax error, unexpected '=', expecting ';' or '\n'
def name = name
^

REPLY

Juliana 2 ANOS AGO

```
class Produto
attr_accessor :nome, :preco

def initialize(nome, preco)
@nome = nome
@preco = preco
end
end

class Mercado < Produto

def initialize(produto)
@produto = produto
end

def comprar
puts "Você comprou o produto #{@produto.nome} no valor de #
{@produto.preco}."
end
```

end
end

require_relative 'produto'
require_relative 'mercado'
produto = Produto.new('Arroz', 5.00)
mercado = Mercado.new(produto)
puts mercado.comprar

REPLY

Juliana 2 ANOS AGO

class Produto

attr_accessor :nome, :preco

def initialize(nome, preco)
@nome = nome
@preco = preco
end
end

class Mercado < Produto

def initialize(produto)
@produto = produto
end

def comprar
puts "Você comprou o produto #{@produto.nome} no valor de #{@produto.preco}."
end

end

require_relative 'produto'
require_relative 'mercado'

produto = Produto.new('Arroz', 5.00)
mercado = Mercado.new(produto)

puts mercado.comprar

REPLY

Gustavo Coelho 2 ANOS AGO

#MISSÃO

#app.rb

require_relative “produto”
require_relative “mercado”

produto = Produto.new

produto.nome = “Produto A”
produto.preco = 25.00

merc = Mercado.new(produto)

merc.comprar

#produto.rb

class Produto
attr_accessor :nome, :preco
end

#mercado.rb

require_relative “produto”
class Mercado
def initialize(pro = Produto.new)
@produto = pro
end
def comprar
puts “Você comprou o produto #{@produto.nome} no valor de #{@produto.preco}.”
end
end

REPLY

Ideilson 2 ANOS AGO

===app.rb===
require_relative ‘produto’
require_relative ‘mercado’

produto = Produto.new
produto.nome = “Leite”
produto.preco = 12.5

mercado = Mercado.new(produto)
mercado.comprar
===mercado.rb===
class Mercado
def initialize(produto)

@produto = produto
end

def comprar
puts “Você comprou o produto #{@produto.nome} no valor de #{@produto.preco}”
end
end
===produto.rb===
class Produto
attr_accessor :nome, :preco
end

REPLY

Frankyston Lins 2 ANOS AGO

https://gist.github.com/frankyston/d217446929eee4f436a7fa95036de5

REPLY

Eric Coutinho 2 ANOS AGO

Minhas respostas vão tudo na mesma linha!! Não sei o por quê

REPLY

Eric Coutinho 2 ANOS AGO

class Market
def initialize(produto)
@produto = produto
end

def buy
puts “Você comprou o produto #{@produto.name} no valor de #{@produto.price}”
end
end

class Product
attr_accessor :name, :price
def initialize(name, price)
@name = name
@price = price

```
end
end

require('./produto.rb')
require('./mercado.rb')

puts 'Qual produto deseja comprar?'
puts ' '
puts '1 – Amazfit Smart Watch – Valor: R$600'
puts '2 – Mi Band – Valor: R$120'
puts '3 – Redmi Note 8 – Valor: R$1020'
puts '4 – Mi Notebook Air – Valor: R$5000'
puts ' '
print 'Opção: '
op = gets.chomp.to_i

case op
when 1
produto = Product.new('Amazfit Smart Watch', 'R$600')
when 2
produto = Product.new('Mi Band ', 'R$120')
when 3
produto = Product.new('Redmi Note 8 ', 'R$1020')
when 4
produto = Product.new('Mi Notebook Air ', 'R$5000')
else
puts 'Compre alguma coisa! Preciso pagar o aluguel!'
op = false
end
if (op)
market = Market.new(produto).buy
end
```

REPLY

Eric Coutinho 2 ANOS AGO

mercado.rb

```
class Market
def initialize(produto)
@produto = produto
end

def buy
puts "Você comprou o produto #{@produto.name} no valor de #
```

```
{@produto.price}”
end
end
```

Produto.rb

```
class Product
attr_accessor :name, :price

def initialize(name, price)
@name = name
@price = price
end
end
```

App.rb

```
require(‘./produto.rb’)
require(‘./mercado.rb’)

puts ‘Qual produto deseja comprar?’
puts ‘ ‘
puts ‘1 – Amazfit Smart Watch – Valor: R$600’
puts ‘2 – Mi Band – Valor: R$120’
puts ‘3 – Redmi Note 8 – Valor: R$1020’
puts ‘4 – Mi Notebook Air – Valor: R$5000’
puts ‘ ‘
print ‘Opção: ‘
op = gets.chomp.to_i

case op
when 1
produto = Product.new(‘Amazfit Smart Watch’, ‘R$600’)
when 2
produto = Product.new(‘Mi Band ‘, ‘R$120’)
when 3
produto = Product.new(‘Redmi Note 8 ‘, ‘R$1020’)
when 4
produto = Product.new(‘Mi Notebook Air ‘, ‘R$5000’)
else
puts ‘Compre alguma coisa! Preciso pagar o aluguel!’
op = false
end
if (op)
market = Market.new(produto).buy
end
```

end

REPLY

Flávio S Ferreira 2 ANOS AGO

produto.rb

```
class Produto
attr_accessor :nome, :preco
def initialize(nome, preco)
@nome = nome
@preco = preco
end
end
```

mercado.rb

```
class Mercado
def initialize(produto)
@produto = produto
end

def comprar
puts “Você comprou o produto #{@produto.nome} no valor de #{@produto.preco}.”
end
end
```

app.rb

```
require_relative ‘produto’
require_relative ‘mercado’

arroz = Produto.new(‘Arroz’, 2.99)
feijao = Produto.new(‘Feijão’, 4.99)

Mercado.new(arroz).comprar
Mercado.new(feijao).comprar
```

REPLY

Edward 3 ANOS AGO

#Classe produto

```
class Produto
attr_accessor :nome, :preco

def initialize(nome_produto, preco_produto)
@nome = nome
@preco = preco
```

```
end
end

#Classe Mercado
class Mercado < Produto
def initialize (produto)
@produto = produto
end

def comprar
unless @produto.nome == '' || @produto.preco == 0
puts "Você comprou o produto #{@produto.nome.capitalize} no valor de R$: #{@produto.preco}"
else
puts 'Desculpe não oferecemos produto de graça ou produtos sem nome!'
end
end
end

#App
require_relative 'produto.rb'
require_relative 'mercado.rb'

puts '- Bem-vindo a vendinha do Edward -'
print 'Digite o nome do produto que deseja comprar: '
nome_produto = gets.chomp
print 'Agora digite seu preço: '
preco_produto = gets.chomp.to_f

produto_novo = Produto.new(nome_produto, preco_produto)
meu_mercado = Mercado.new(produto_novo)
meu_mercado.comprar
```

REPLY

Gabriel 2 ANOS AGO

Mto obrigado, eu n tinha conseguido fazer esse

REPLY

Edward 3 ANOS AGO

#Correção no initilize do produto, estava errado a nomenclatura na atribuição para os atributos da classe
def initialize(nome_produto, preco_produto)

```
@nome = nome_produto
@preco = preco_produto
end
```

REPLY

Leonardo Scorza 3 ANOS AGO

Boa o/

REPLY

Alessandro Moreira 3 ANOS AGO

Entendi bem a diferença do require e require_relative, ficou facim! Mas o path_load, tamém pode ser utilizado pelo ruby para encontrar o arquivo que esta sendo referenciado e assim resolveria casos como a importação do arquivo em uso para herança. Estou certo ?

REPLY

Denilson Silva 3 ANOS AGO

produto.rb

```
class Produto < Mercado
attr_accessor :nome, :preco
end
```

mercado.rb

```
class Mercado
def comprar(nome, preco)
puts "Voce comprou o produto #{nome} no valor #{preco}"
end
end
```

app.rb

```
require_relative 'mercado'
require_relative 'produto'

produto = Produto.new
produto.nome = "Galaxy S9+"
produto.preco = 234
produto.comprar(produto.nome, produto.preco)
```

REPLY

Yuri Tavares 3 ANOS AGO

Yuri Tavares 3 ANOS AGO

App. Rb

require_relative ‘product’
require_relative ‘market’

product = Product.new(‘Manga’, 12.40)
market = Market.new(product)

market.buy

——————————

market.rb

class Market
def initialize(product)
@product = product
end

def buy
puts “Você comprou o produto #{@product.name} no valor de #{@product.price}”
end
end

——————————

product.rb

class Product
attr_accessor :name, :price

def initialize(name, price)
@name = name
@price = price
end
end

REPLY

Iago Silva Vieira 3 ANOS AGO

Abaixo está meu arquivo APP, alguém tem uma idéia melhor para criar classes de forma dinâmica? Como as variáveis são de instancia, essa foi a única forma que pensei de criar vários produtos.

require_relative “produto.rb”
require_relative “mercado.rb”

puts “Compre os seus produtos informando o nome e o valor deles.”

```
continue = “Sim”
cont = 0
total = 0
array = []

loop do

if(continue==”Sim”)

array.push(“p#{cont}”)

array[cont] = Produto.new

puts “\nInforme o nome do produto #{cont}: ”
array[cont].nome = gets.chomp.to_s

puts “\nInforme o valor do Produto #{cont}: ”
array[cont].valor = gets.chomp.to_f

puts “\nDeseja comprar outro produto? Sim/Não?”
continue = gets.chomp.to_s

if(continue == “Não”)

puts “\nAbaixo segue um resumo da sua conta: \n”

while cont >= 0

m = Mercado.new(array[cont]).comprar
total += array[cont].valor
cont -= 1
end
puts “\nTotal da compra R$ #{total} ———————”
end

cont += 1
else
puts “\nVocê digitou a opção errada, digite Sim/Não.”
continue = gets.chomp.to_s
end
break if continue == “Não”
end
```

REPLY

Neemias 3 ANOS AGO

Ótimas aulas Leonardo! Você ensina muito bem!

REPLY

GUILHERME ANGELO MAZZA 3 ANOS AGO

Seria possivel a liberação das aulas com 3 dias ? tenho um tempo um pouco mais curto pra apreder e o curso é extremamente bom !!!

REPLY

Leonardo Scorza 3 ANOS AGO

E ai Guilherme, como vai?
Fico muito feliz que você esteja gostando do curso 🙂
Sobre o prazo, não podemos modifica-lo (ele está ligado a muitas variáveis), mas aproveita que no blog tem muito coisa boa sobre ruby e sobre rails \o/

REPLY

ASK QUESTION