

# Introdução ao Ruby on Rails

by Jackson Pires



**O paradigma POO**

# Errata

# Errata

- `if(5>7); end`
- `if 5>7; end`
- `if (5>7) then; end`
- `if 5>7 then; end`

**O paradigma POO**

# O que é paradigma?

- “Paradigma é um conjunto de regras que estabelecem fronteiras e descrevem como resolver os problemas dentro destas fronteiras. ”

# O que é paradigma?

- “Paradigma é um conjunto de regras que estabelecem fronteiras e descrevem como resolver os problemas dentro destas fronteiras. ”
- “Os paradigmas influenciam nossa percepção; ajudam-nos a organizar e a coordenar a maneira como olhamos para o mundo...”

# O que é paradigma?

- “Paradigma é um conjunto de regras que estabelecem fronteiras e descrevem como resolver os problemas dentro destas fronteiras. ”
- “Os paradigmas influenciam nossa percepção; ajudam-nos a organizar e a coordenar a maneira como olhamos para o mundo...”
- "... Programação Orientada a Objetos (POO) é um paradigma para o desenvolvimento de software que baseia-se na utilização de componentes individuais (objetos) que colaboram para construir sistemas mais complexos. A colaboração entre os objetos é feita através do envio de mensagens".



# O que é paradigma?

- O paradigma de objetos baseia-se em alguns conceitos, como:
  - Classes
  - Objetos
  - Encapsulamento
  - Herança, dentre outros.

# **Classes, Objetos e Métodos**

# O que é uma Classe?

"Uma classe é um gabarito para a definição de objetos."



# O que é uma Classe?

"Uma classe é um gabarito para a definição de objetos."

Através da definição de uma classe, descreve-se que **propriedades/atributos** e **métodos/ações** o objeto terá.



# Definindo uma classe no Ruby

```
#lobo.rb  
class Lobo  
  ...  
end
```

# Instanciando uma classe (Objeto)

```
#lobo.rb
class Lobo
end

-----

#app.rb / irb
require_relative 'lobo'
lobo_1 = Lobo.new
lobo_2 = Lobo.new
```

# Definindo uma ação/método em uma classe

```
#lobo.rb  
class Lobo  
  def uivar  
    puts "auuu!"  
  end  
end
```

# Definindo uma ação/método em uma classe

```
#lobo.rb
class Lobo
  def uivar
    puts "auuu!"
  end
end

-----

#app.rb / irb
require_relative 'lobo'
lobo_1 = Lobo.new
lobo_1.uivar
```



# Usando parâmetros uma ação/método

```
#lobo.rb
class Lobo
  def uivar(forca=3)= puts "a#{'u'*forca}!"
end
-----
#app.rb / irb
require_relative 'lobo'
lobo_1 = Lobo.new
lobo_1.uivar(7)
```

# Encapsulamento

# Encapsulando métodos

```
#lobo.rb
class Lobo
  def uivar(forca=3)
    puts "a#{'u'*forca}!"
    grunir
  end

  private

  def grunir = puts "arrgggg!!!"
end
-----
#app.rb / irb
require_relative 'lobo'
lobo_1 = Lobo.new
lobo_1.uivar
```

# Encapsulando métodos

```
#lobo.rb
class Lobo
  def uivar(forca=3)
    puts "a#{'u'*forca}!"
    grunir
  end

  private

  def grunir = puts "arrggggg!!!"
end
-----
#app.rb / irb
require_relative 'lobo'
lobo_1 = Lobo.new
lobo_1.public_methods
lobo_1.public_methods.include(:uivar)
lobo_1.public_methods.include(:grunir)
```

The background is a solid red color with a complex, abstract pattern of overlapping, sharp, geometric shapes. These shapes, which include various triangles and polygons, are rendered in slightly different shades of red, creating a sense of depth and movement. The overall effect is a dynamic, crystalline texture.

# Herança

# Herdando classes

```
#animal.rb
class Animal
  def correr = puts "correndo..."
end
-----
#lobo.rb
require_relative 'animal'
class Lobo < Animal
  ...
end
-----
#app.rb / irb
require_relative 'lobo'
lobo_1 = Lobo.new
lobo_1.corre
lobo_1.uiva
```

# Módulos

# Módulos

Módulos Ruby são similares a classes em relação ao fato de que também **armazenam uma coleção de métodos, constantes, outras definições de módulos e classes.**

Diferente das classes, **você não pode criar objetos baseados em módulos** nem pode criar módulos que herdam desse módulo.



# Módulos

```
#bichos.rb
module Bichos
  MAIS_COMUNS = ['cachorro', 'gato']

  class Animal
    def corre = puts "correndo..."
  end

  class Lobo < Animal
    ...
  end
end
-----
```

```
#app.rb / irb
require_relative 'bichos'
include Bichos
Bichos::MAIS_COMUNS
lobo_1 = Lobo.new
lobo_1.corre
lobo_1.uiva
```

# Exercício

# Exercício

Crie um módulo **Matematica** que contenha a classe **Calculadora**, e essa classe deve possuir os métodos **somar**, **subtrair**, **multiplicar** e **dividir** que aceitam dois parâmetros a fim de realizar a operação em questão e mostrar através de um `puts`. Em seguida, crie uma arquivo `app.rb` que use esse módulo e classe.