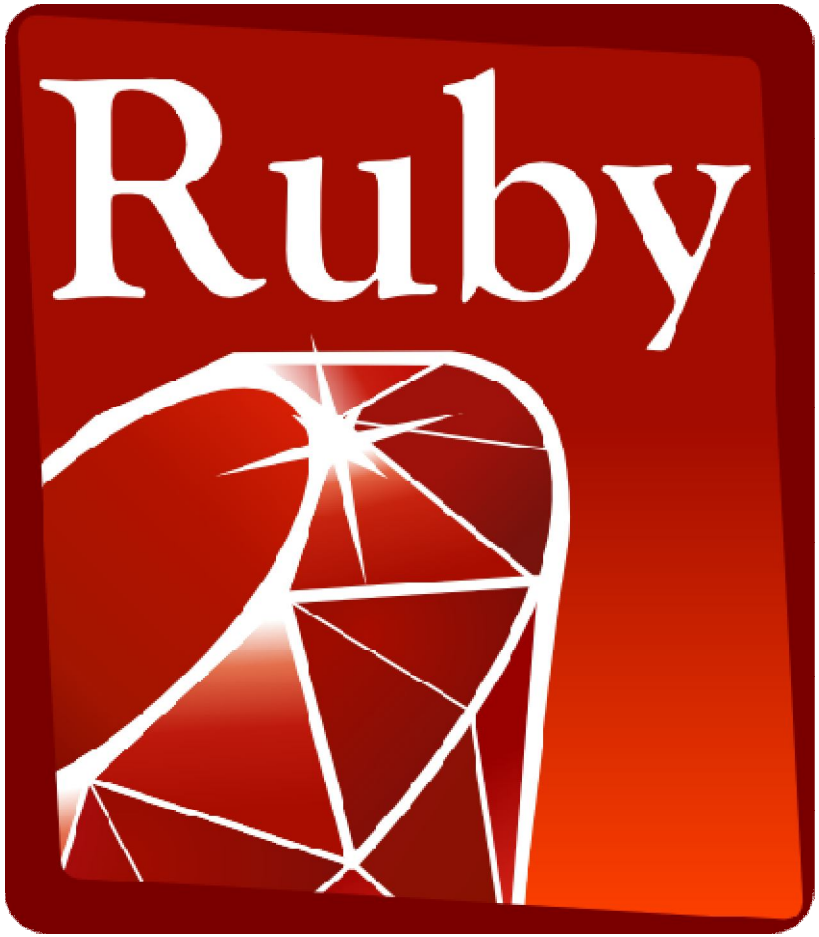
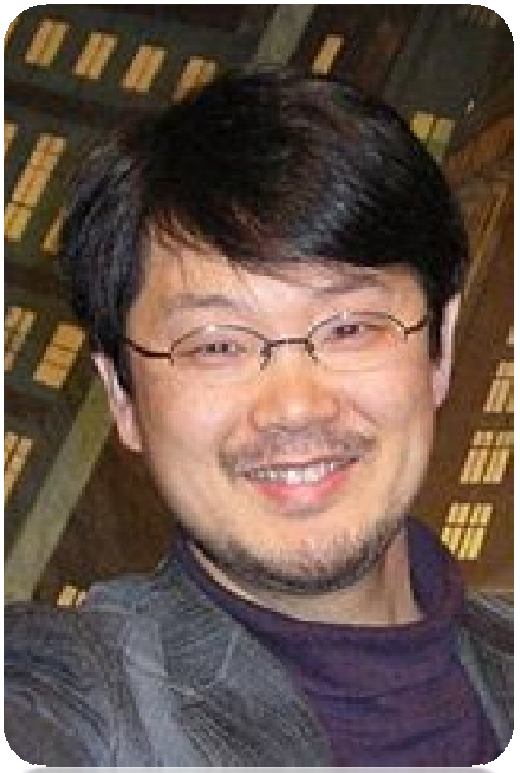
Linguagem de Programação

- Helder
- Lucas S.
- Silas
- Valdemar

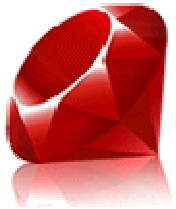


Perl, Smalltalk, Eiffel, Ada e Lisp.

“Mais poderosa do que Perl, e mais orientada a objetos do que Python.”



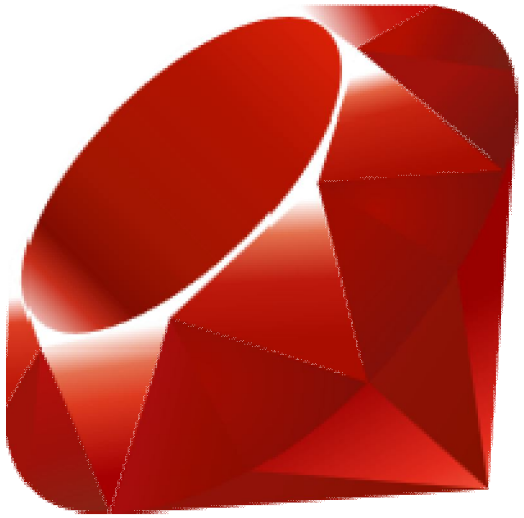
Yukihiro Matsumoto “Matz“



Ruby
A Programmer's Best Friend

- “Coral” ou “Ruby”?

- Totalmente livre. Não somente livre de custos, mas também livre para utilizar, copiar, modificar e distribuir.



- Linguagem de script.
- Equilibra a programação funcional com a programação imperativa.

- Uma linguagem criada para que todas as pessoas pudessem programar de uma forma mais fácil e divertida.

- “Tento tornar o Ruby natural, não simples”

Histórias de Sucesso

- *Simulação*
  - *NASA Langley Research Center*
- *Negocio*
  - Toronto Rehab
- *Robótica*
  - MORPH
- *Redes*
  - Open Domain Server
- *Telefonica*
  - Lucent
- *Administração de Sistemas*
  - Level 3 Communications
- *Aplicações Web*
  - Basecamp
  - 43 Things
  - A List Apart
  - Blue Sequence

Ruby On Rails

Linguagem de programação

Direcionada à aplicativos da Web

Ruby Game Scripting System (RGSS)

- Extensão da biblioteca original do Ruby
- Códigos/funções são voltados para a criação de jogos
- Ruby pode não funcionar no interpretador de RGSS
- Principal utilização:
  - RPG Maker

Legibilidade/ Redigibilidade

- Sintaxe Limpa, simples, exata.
- Leitura natural e fácil escrita.
- *“O Ruby é simples na aparência, mas muito complexo no interior”.* Yukihiro Matsumoto
- - linguagem de computador + linguagem natural

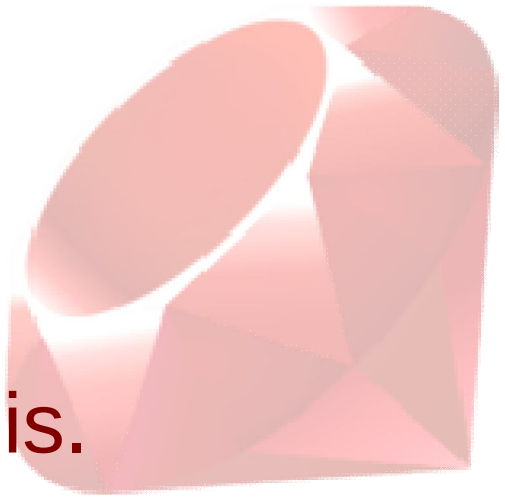
5.Times{print “Olá Mundo!”}
Cinco vezes imprima Olá Mundo!

Legibilidade/ Redigibilidade

“When we discovered Ruby, we realized that we’d found what we’d been looking for. In Ruby you can concentrate on solving the problem at hand, instead of struggling with compiler and language issues. That’s how it can help you become a better programmer: by giving you the chance to spend your time creating solutions for your users, not for the compiler.”

Dave Thomas, Andy Hunt. Autores do livro Programming Ruby

Confiabilidade



- Verificação de tipos de variáveis.
- Capacidade de tratamento de exceções, tal como o Java ou Python, facilitando o tratamento de erros.

Portabilidade

- O Ruby é altamente portável
- Desenvolvido principalmente em ambiente GNU/Linux
- Mas trabalha em muitos tipos de ambientes:
  - UNIX, Mac OS X, Windows, DOS, BeOS, OS/2, etc.

Eficiência

- Linguagem interpretada:
  - A execução é muito mais lenta comparada a execução de programas compilados.
- Tipagem Dinâmica:
  - Tende a ser mais lenta em tempo de execução, pois deve-se verificar o tipo de cada interação.

Palavras Reservadas

- Palavras reservadas não podem ser usadas como nomes de classes, variáveis ou qualquer outra coisa

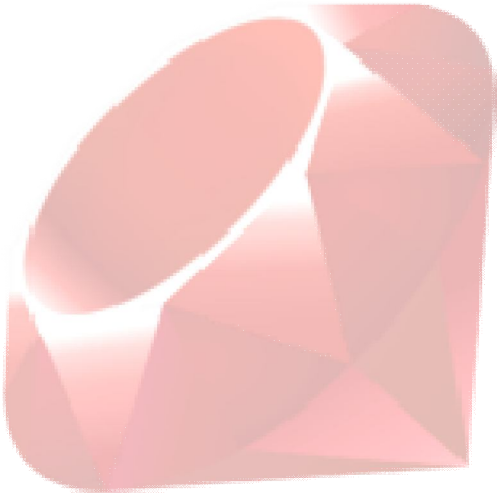
  As palavras reservadas são:

BEGIN	**class**	**ensure**	**nil**	**self**	**when**
END	**def**	**false**	**not**	**super**	**while**
alias	**defined**	**for**	**or**	**then**	**yield**
and	**do**	**if**	**redo**	**true**	
begin	**else**	**in**	**rescue**	**undef**	
break	**elsif**	**module**	**retry**	**unless**	
case	**end**	**next**	**return**	**until**	

Tradução da Linguagem

- Ruby é uma linguagem interpretada.
- Sem necessidade de um bytecode.
- Apenas um interpretador é necessário para traduzir o código.

Tipagem em Ruby



- C é uma linguagem fracamente tipada:
  - char a= 'a', b='b';
  - float c = a + b;
- Ele mostra que podermos somar dois chars e atribuirmos a um float.
- No entanto, Ruby é **dinamicamente, implicitamente e fortemente** tipada.

Tipagem em Ruby

- **Dinamicamente tipada**
- Quer dizer que a cada interação, o tipo é verificado

Teste.rb

```
1  x = 100
2  4.times{ x = x*100 ; puts "x pertence a classe #{x.class} #{x} "}
```

```
x pertence a classe Fixnum 10000
x pertence a classe Fixnum 1000000
x pertence a classe Fixnum 100000000
x pertence a classe Bignum 10000000000
[Finished in 1.0s]
```

Tipagem em Ruby

- **Implicitamente tipada**
  - x = 100 não precisamos declarar o tipo de x. Ou seja, não foi necessário fazer algo como:
  - Fixnum x = 100.
  - tipo de cada variável em tempo de execução.
- **Fortemente tipada**
  - todas as variáveis devem fazer parte de uma classe

    a = 100

    b = "Ruby“

    a + b

    TypeError: String can't be coerced into Fixnum

Gerenciamento (alocação/liberação) de Memória

- Coletor de lixo
  - Gerencia a memória
  - Libera memória ocupada fora de uso
  - Evita aborrecimento, erros e pode melhorar o desempenho,

- Desvantagens
  - Menor controle sobre a gerência de memória
  - É difícil saber o momento exato e a ordem em que os objetos são destruídos.

Passagem de Parâmetros

- Todos os parâmetros são passados por cópia da referência.

```
1   x = 10
2
3   def incrementa(a)
4       a += 1
5   end
6
7   incrementa(x)
8   puts x
9
10
```

```
10
[Finished in 0.2s]
```

Passagem de Parâmetros

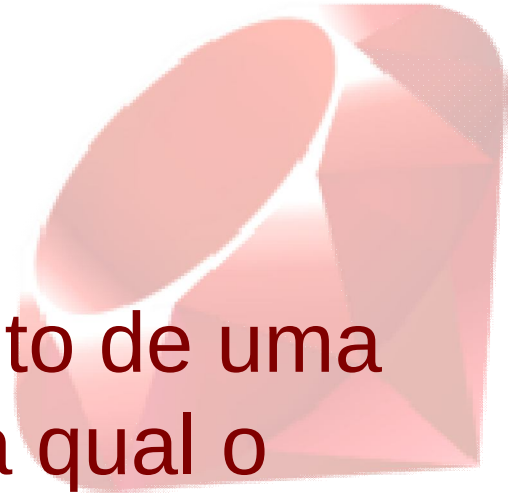
```ruby
class Teste
    attr_accessor :valor
    def initialize(x)
    @valor = x
    end
end

teste = Teste.new(10)
puts "#{teste.valor}"

def ModificaTeste(objeto)
    objeto.valor = 123
end

ModificaTeste(teste)
puts "#{teste.valor}"
```

```
10
123
[Finished in 0.2s]
```

```ruby
x = "Ola Mundo"

def AlteraString(s)
    s << "--Alterado--"
end

AlteraString(x)

puts x

```

```
Ola Mundo--Alterado--
[Finished in 0.2s]
```

Curto-Circuito



- Uma avaliação em curto-circuito de uma expressão é uma avaliação na qual o resultado é determinado sem avaliar todos os operandos e/ou operadores.
- Todos os operados lógicos de Ruby são avaliados em curto-circuito.

Curto-Circuito

```ruby
a = 10
b = 15
if a<b && (a+=1)<b
end
puts "a = #{a}"
```

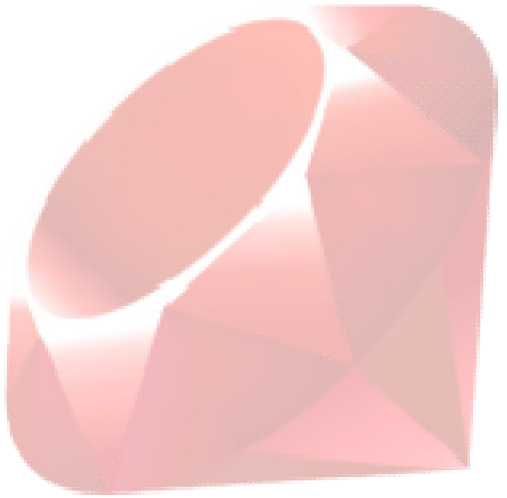
```
a = 11
[Finished in 0.2s]
```

```ruby
a = 10
b = 15
if a>b && (a+=1)<b
end
puts "a = #{a}"
```

```
a = 10
[Finished in 0.2s]
```

Modificadores de Acesso

- Aplicados somente a métodos.
- Public, Protected e Private.
- Padrão Public.



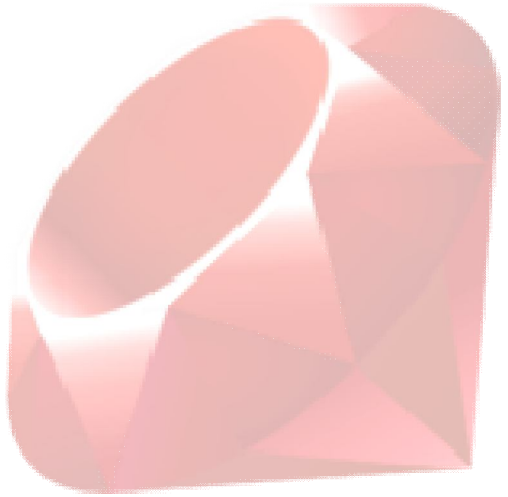
Acesso a Atributos

```ruby
class Teste
    def initialize(x)
        @valor = x
    end
    def get_valor()
        @valor
    end
    def set_valor(x)
        @valor = x
    end
end

```

```ruby
class Teste

    attr_accessor:valor

    def initialize(x)
        @valor = x
    end
end

#attr_writer - Somente os setters
#attr_reader - Somente os getters

```

Serialização de objetos

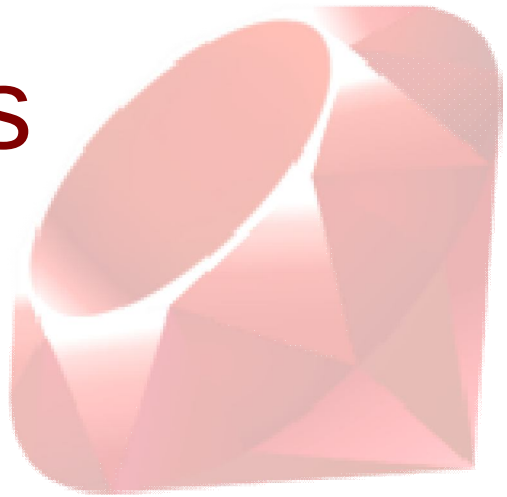
- Marshaling (empacotamento).
- Marshal.dump para salvar versão serializada de objeto no disco.
- Marshal.load para lê-lo.



Tipos de Dados

- Não existe tipos primitivos, todos os tipos são classes
- Object : classe mãe
  - Numeric : classe abstrata
    - Integer : números inteiros
      - Fixnum : inteiros de precisão fixa
      - Bignum : inteiros de precisão infinita
    - Float : números de ponto flutuante
  - String : cadeia de caracteres
  - Symbol : mesmo endereço de memória
  - Array : representar matrizes e vetores
  - Hash : vetor associativo

Declaração de variáveis



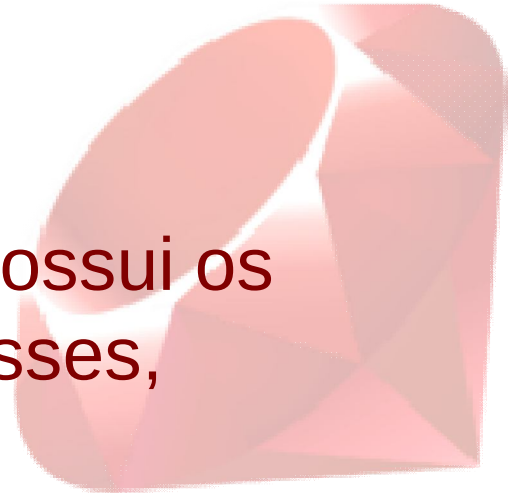
- Objeto: atribuição comum:
  - var1 = 2
  - var2 = Classe.new
  - var3 = Classe2.new(parametro)
- Variável Local: Declarada normalmente
  - local = "local"
- Variável de instância: Declarada com um "@"
  - @instancia = 42
- Variável de classe: É declarada com "@@“
  - @@classe = /f+/
- Variável global: É declarada com "$“
  - $Pi = 3.1415926
- Variáveis constantes: iniciam com uma letra maiúscula

Herança

- Ruby não suporta herança múltipla. Ao invés disso, Ruby usa Mixins para emular herança múltipla:

```ruby
class Pessoa < Mamifero # Herança de Mamifero
  include Humano # Emulando herança múltipla
end
```

Modules



- Além das classes normais, Ruby possui os "Modules", que são classes de classes, permitindo espaço de nomes:

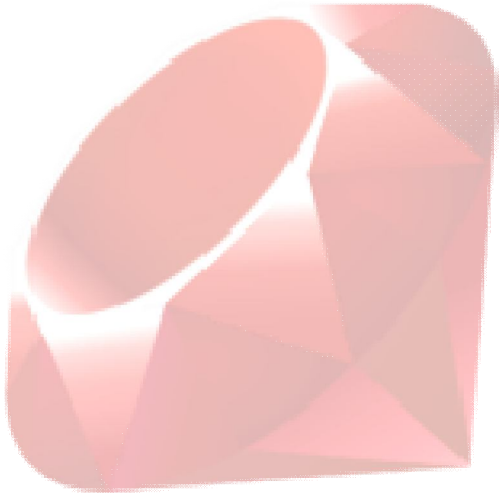
```ruby
module Humano
  class Classe1
    def info
      "#{self.class} (\##{self.object_id}): #{self.to_s}"
    end
  end
end
```

Tratamento de exceções

- Palavras-chave:
  - "begin", "rescue" e "ensure". "Begin"

```
begin
# Faça algo
rescue
# Trata alguma exceção
else
# Faça isto se nenhuma exceção for lançada
ensure
# Faça isto se alguma ou nenhuma exceção for lançada
end
```

Ruby a partir de Java

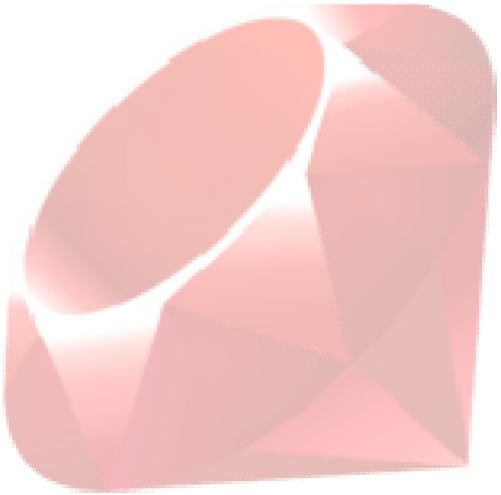


- Semelhanças
  - garbage collector
  - Objectos fortemente tipados
  - Métodos públicos, privados e protegidos
  - Existem ferramentas de documentação embebidas
    - Rdoc e javadoc

- Diferenças
  - Não precisará de compilar código-fonte;
  - Há inúmeras ferramentas de interface gráfica;
  - Todas as variáveis de instância são privadas;
  - Tudo é um objeto;
  - Não existe validação estática de tipos de dados;
  - Os nomes de variáveis são apenas etiquetas;
  - Não existe declaração de tipo de dados;
  - O construtor é sempre chamado “initialize” em vez do nome da classe.

Mini - Tutorial

```
irb(main):001:0>

irb(main):002:0> puts “Ola Mundo”
Ola Mundo
=> nil

irb(main):003:0>  3 + 5
=>  8

irb(main):004:0> def h
irb(main):005:1> puts “Ola Mundo!”
irb(main):006:1> end
irb(main):007:0> h
“Ola Mundo!”
=> nil
```

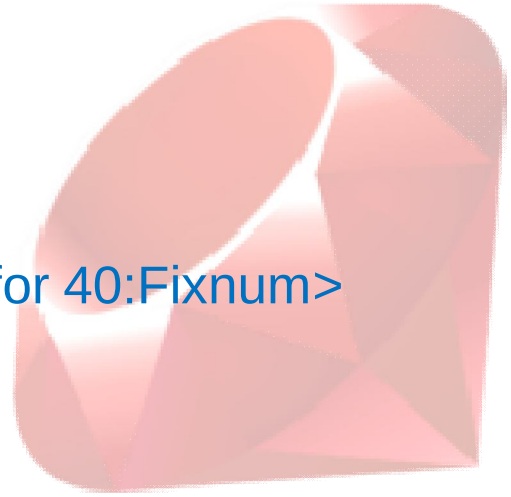


```
irb(main):011:0> def h(nome)
irb(main):012:1> puts “Ola #{nome}!”
irb(main):013:1> end
=> nil
irb(main):014:0> h(“Matz”)
Ola Matz!
=> nil
irb(main):015:0> def h(nome = “Mundo”)
irb(main):016:1> puts “Ola #{nome.capitalize}!”
irb(main):017:1> end
=> nil
irb(main):018:0> h(“Chris”)
Ola Chris!
irb(main):019:0> h
Ola Mundo!
```

Ampla biblioteca

```
irb(main):021:0> LP = “LinguagemRuby”
=> “LinguagemRuby”
irb(main):022:0> 2.times.{print LP}
LinguagemRubyLinguagemRuby
irb(main):023:0> LP.length
=> 13
irb(main):024:0> LP.reverse
=> “ybuRmegaugniL”
irb(main):025:0> LP = 12345
=> 12345
```

```
irb(main):027:0> LP.reverse
=> #<NoMethodError: undefined method `reverse' for 40:Fixnum>
irb(main):028:0> LP.to_s.reverse
=> “54321”
irb(main):029:0> LP.to_i + 2468
=> 56789
```



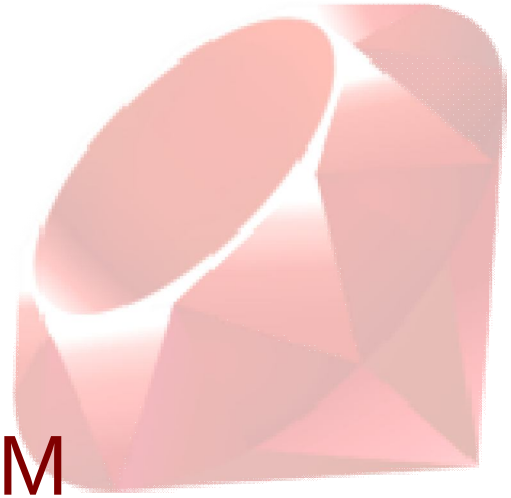
- to_s converter para strings
- to_i converter para integers
- to_a converter para arrays

Sobre Operadores

```
class Fixnum
  def  + ( outro )
       self - outro
  end
end
```

=, .., ..., !, not, &&, and, ||, or, !=, !~, ::

Arquivo Ruby



- Extensão “.rb”.
- Executado diretamente pela VM
  - Não há processo de compilação

```
> ruby programa.rb
```

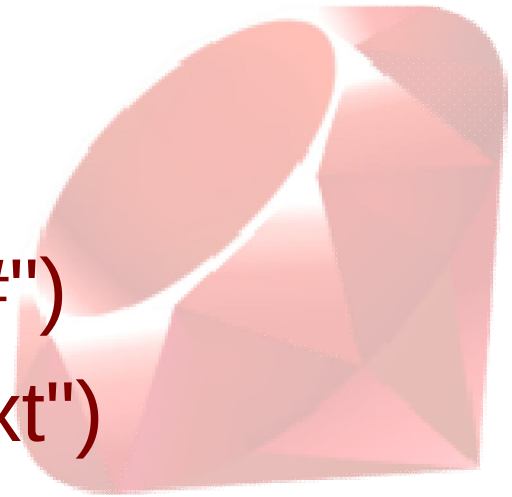
- public, private e protected são métodos

```
class MyClass
    def metodo_um; true; end
    private
    def metodo_dois; false; end
    def metodo_tres; false; end
end
```

Singleton methods

```
class Car
    def inspect
        "Carro barato"
    end
end
porsche = Car.new
porsche.inspect # => Carro barato
def porsche.inspect
    "Carro caro"
end
porsche.inspect # => Carro caro
other_car = Car.new
other_car.inspect # => Carro barato
```

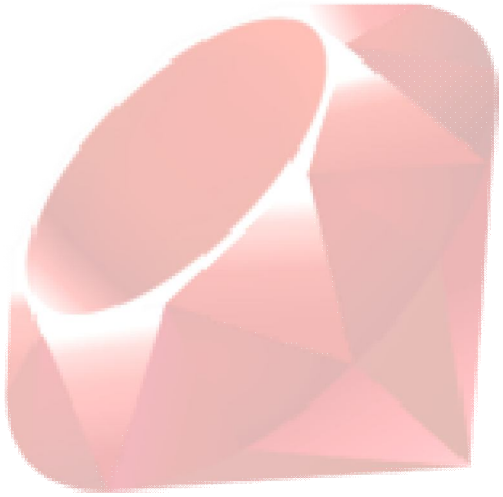
Manipulando arquivos



- arq = File.new("arquivo.txt", “#")
- arq = File.readlines("arquivo.txt")
- arq = File.open("arquivo.txt")

- File.exists?(...)
- File.directory?(...)
- File.size(...)
- File.zero?(...)
- file.closed?(...)
- File.ctime(...)
- File.mtime(...)
- File.atime(...)

Para se aprofundar



- Sobre Ruby
  - www.ruby-lang.org
- Tutorial interativo
  - tryruby.org
- Tudo sobre RGSS
  - www.rgss.com.br