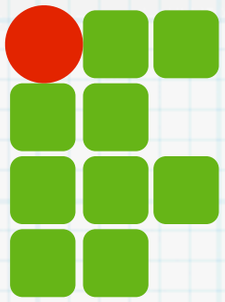


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Programação de Computadores

Novos tipos (classes)

O que veremos hoje?

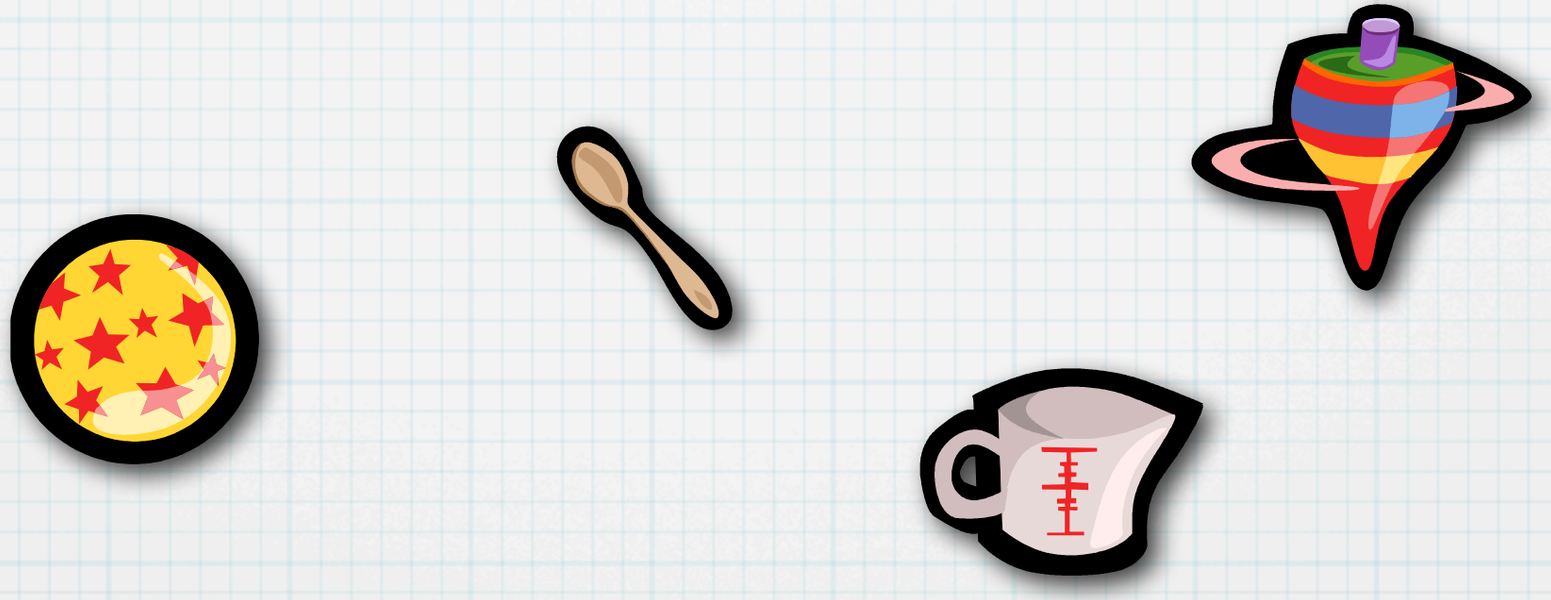
* Introdução
* Modelo de objetos
* Classes e objetos
  * Conceitos
  * Definição de classes
  * Atributos
* Exemplos

Introdução

* Programas realizam operações sobre dados/valores
* Valores pertencem a um tipo
  * Inteiro, Real, String, etc
  * Array (coleção) também é tipo
* Os tipos determinam que operações podem ser efetuadas nos dados
  * Soma para inteiros
  * Tamanho de uma string
  * Ordenar um array

Modelo de objetos

* É como os humanos conhece o mundo
* Desenvolvimento da cognição humana
* "Tudo" é objeto

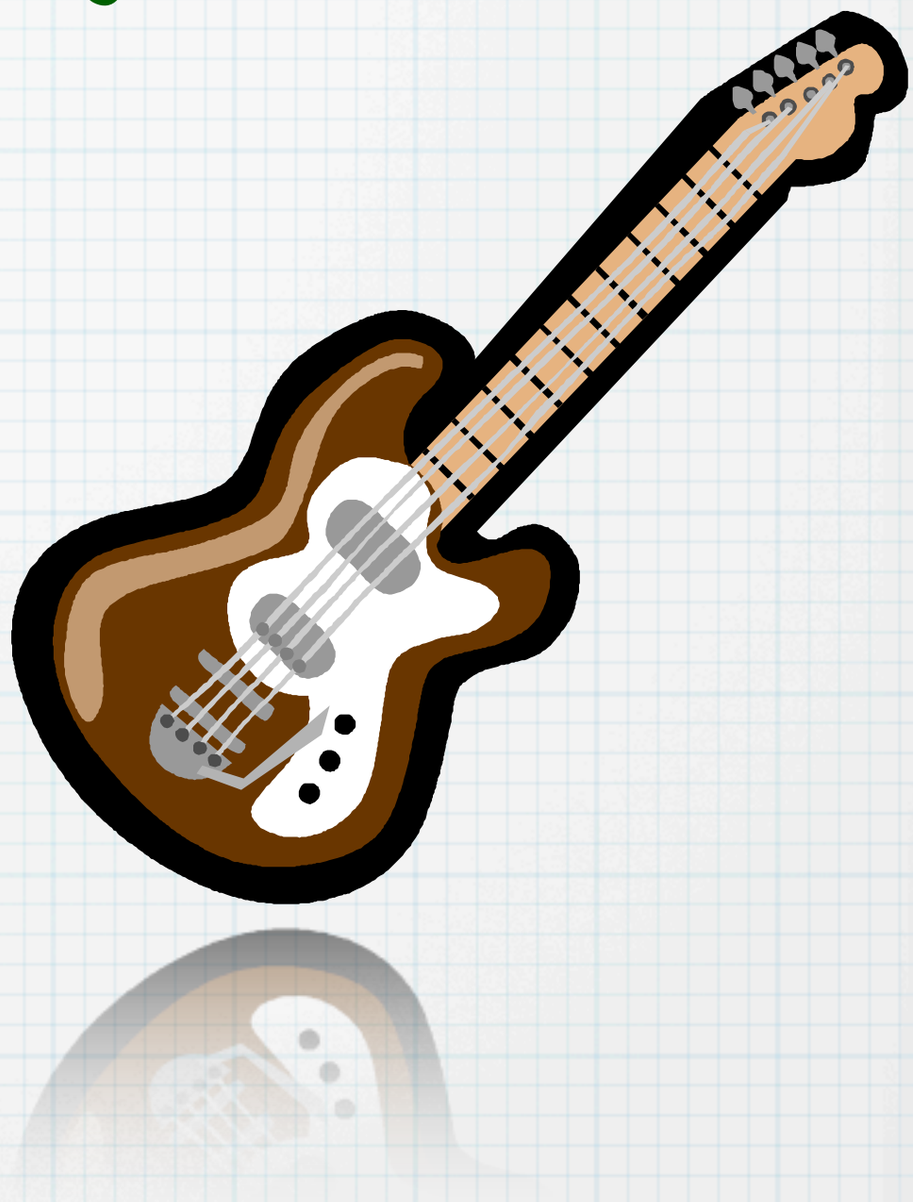


Modelo de objetos

* Um objeto é uma entidade que exibe algum comportamento bem definido.
* Na perspectiva da cognição humana, um objeto é:
  * algo tangível ou visível
  * algo que pode ser apreendido intelectualmente
  * algo para o qual ação ou pensamento é direcionado

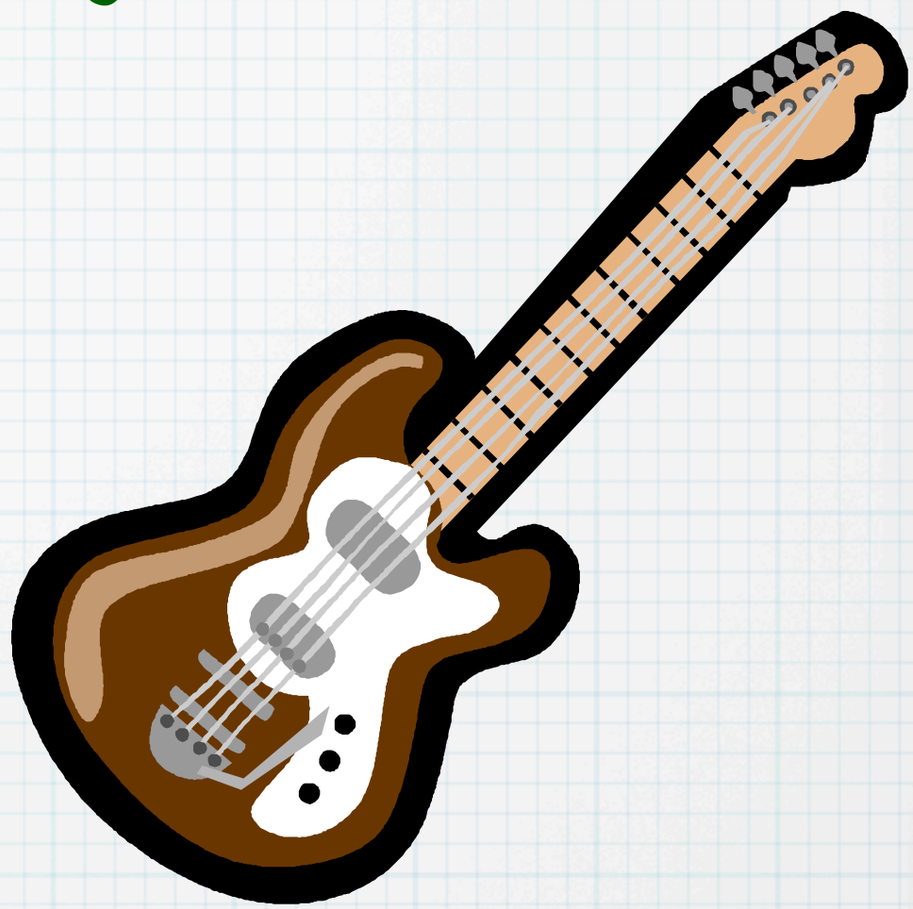
Modelo de objetos

"Um objeto representa um item identificável, uma unidade, ou entidade, individual, seja real ou abstrata, com uma regra bem definida"

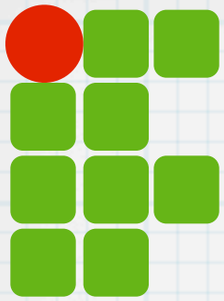


Modelo de objetos

"Um objeto representa um item identificável, uma unidade, ou entidade, individual, seja real ou abstrata, com uma regra bem definida"

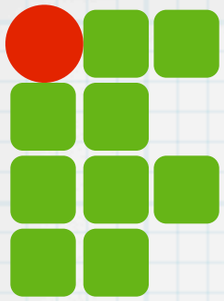


Objeto = Dados + Operações



Mais sobre objetos

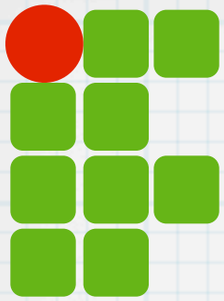
* Objetos têm características (atributos)
* Objetos são/podem ser feitos de outros objetos - atributos de objetos são objetos
* Objetos têm comportamento
* Comportamento pode mudar valores dos atributos
* Exemplos
  * Objetos abstratos
    * Música, Conta bancária, Poema, figuras geométricas, etc
  * Objetos são/podem ser feitos de outros objetos
    * Casa, carro, computador, etc



Objetos

* Exemplo de criação e manipulação de objetos em Ruby

```
meu_carro = Carro.new
meu_carro.definir_cor(azul)
meu_carro.ligar
qtd_combustivel = meu_carro.combustivel
velociade = meu_carro.velocidade
if (velocidade < 50) then
  meu_carro.acelera(10)
end
```



Classes e objetos

* Objetos são semelhantes
  * possuem mesmas características e comportamento
* Classe de objetos
  * Quando falamos em "bola" não estamos falando de nenhum objeto específico

Classes e objetos

* Objetos são semelhantes
  * possuem mesmas características e comportamento
* Classe de objetos
  * Quando falamos em "bola" não estamos falando de nenhum objeto específico

Classe (de objetos):
Definição dos dados e das operações dos objetos.
"Essência do objeto"

Definição de classes

```
class NomeClasse
  def metodo1
    ...
  end
  def metodo2
    ...
  end
  ...
end
```

Definição de classes

```
class NomeClasse
  def metodo1
    ...
  end
  def metodo2
    ...
  end
  ...
end
```

Nome da classe, normalmente em maiúsculo

Definição de classes

```
class NomeClasse
  def metodo1
    ...
  end
  def metodo2
    ...
  end
  ...
end
```

Nome da classe, normalmente em maiúsculo

Lista de definição de métodos para objetos desta classe

Exemplo

* Uma classe para um dado
  * Apenas um método: rolar

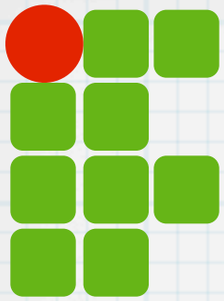
```
class Dado

  def rolar
   return 1 + rand(6)
  end

end
```

```
dado1 = Dado.new
dado2 = Dado.new
puts dado1.rolar
puts dado2.rolar
```

Exemplo

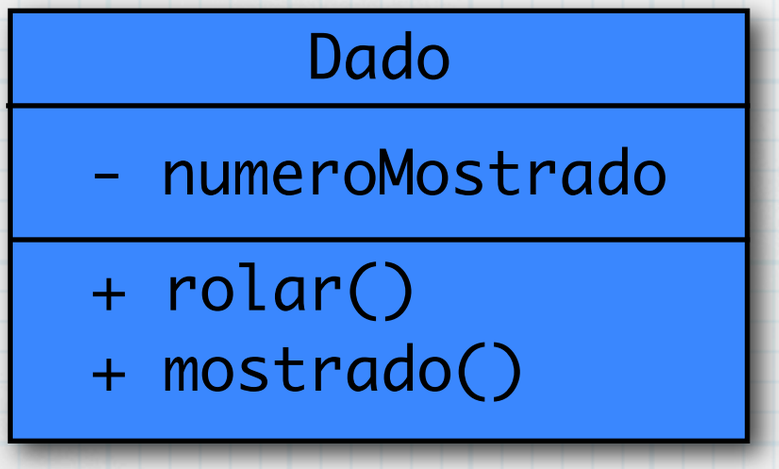
* Atributos
  * Variáveis que pertencem ao objeto
  * TODOS os métodos acessam
  * CADA objeto tem sua variável local
* Considere o dado
  * Armazenar o valor da última jogada
  * um método para acessar

```
class Dado

  def rolar
    @numeroMostrado = 1 + rand(6)
    return @numeroMostrado
  end

  def mostrado
    return @numeroMostrado
  end

end
```



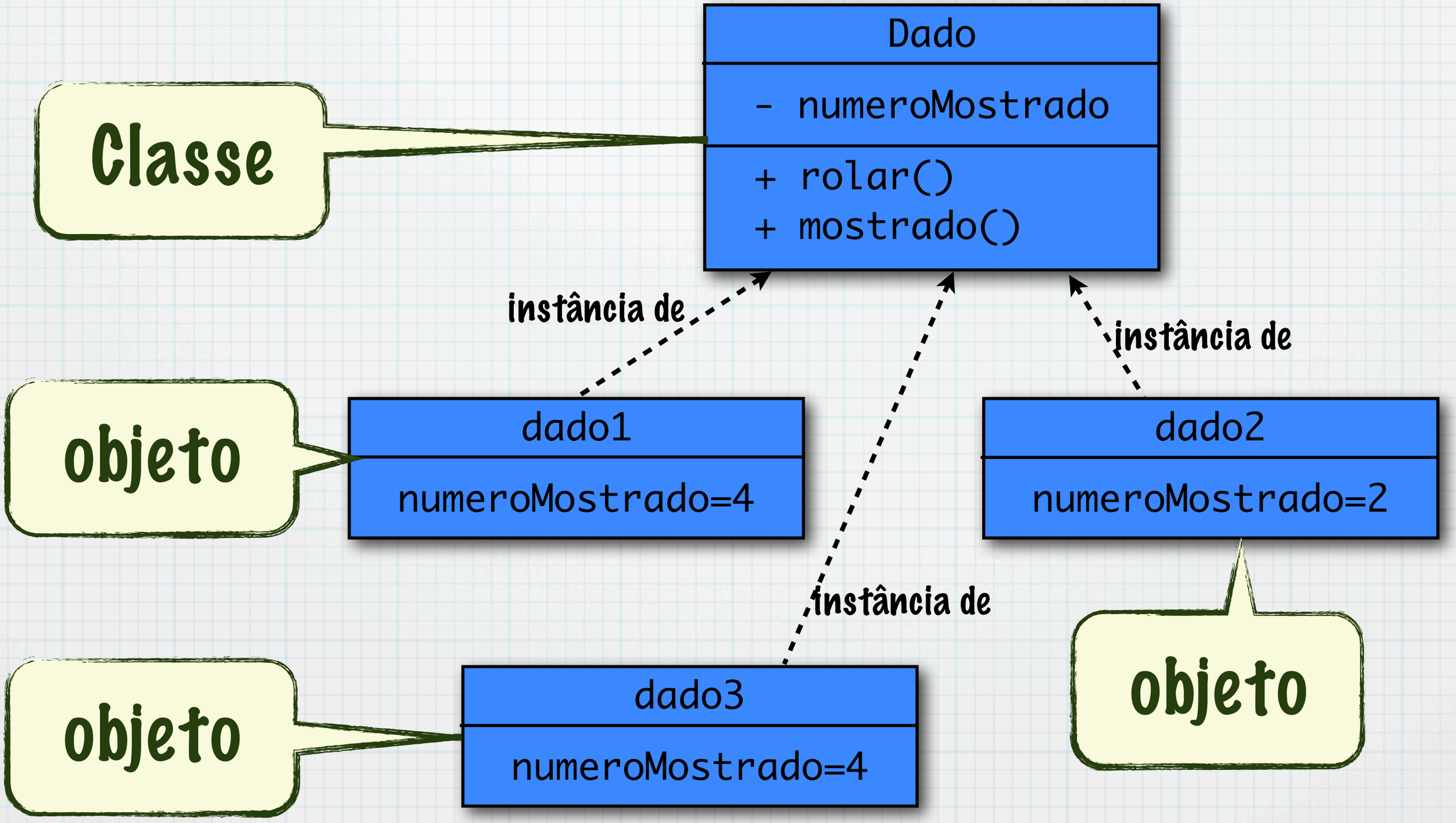
Atributos

* Atributos são variáveis que pertencem a cada objeto
  * Começam pelo caractere @
  * Escopo de objeto




```
class Dado
  def rolar
    @numeroMostrado = 1 + rand(6)
    return @numeroMostrado
  end
  def mostrado
    return @numeroMostrado
  end
end

dado1 = Dado.new
dado2 = Dado.new
puts dado1.rolar
puts dado2.rolar
puts dado1.mostrado
puts dado1.mostrado
puts dado2.mostrado
dado1.rolar
puts dado1.mostrado
```

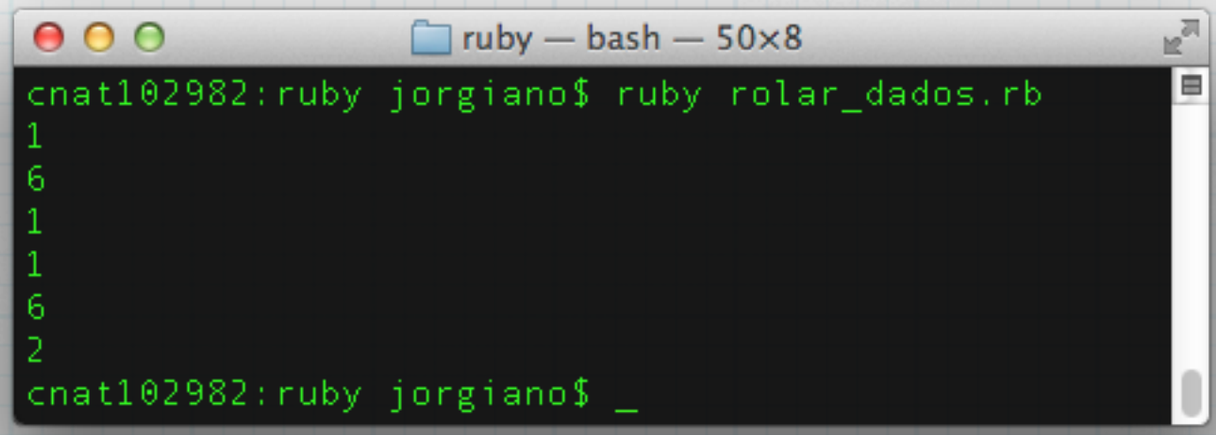
Atributos


Classe
Dado
- numeroMostrado
+ rolar()
+ mostrado()
instância de
instância de
instância de
objeto
dado1
numeroMostrado=4
dado2
numeroMostrado=2
instância de
objeto
dado3
numeroMostrado=4
objeto

Execução

rolar_dados.rb

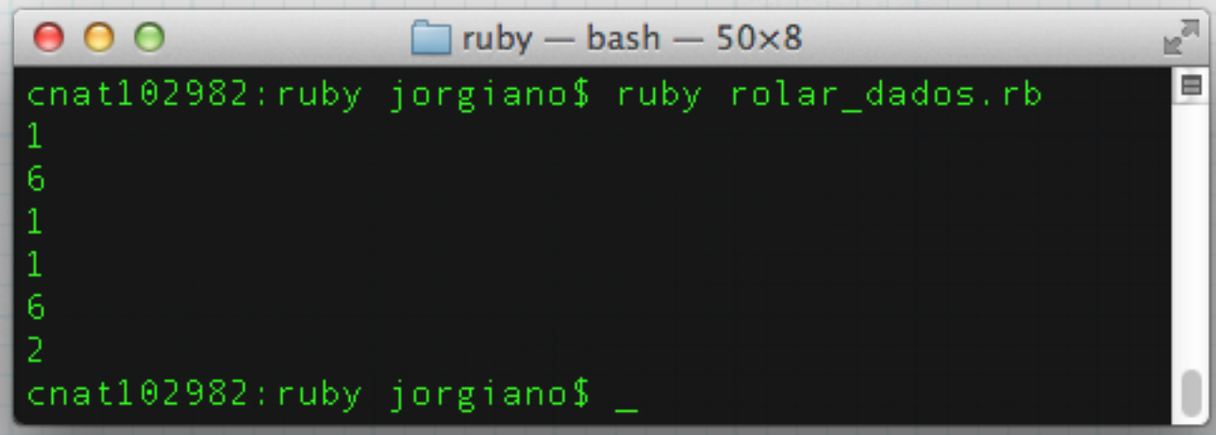
```
class Dado
  def rolar
    @numeroMostrado = 1 + rand(6)
    return @numeroMostrado
  end
  def mostrado
    return @numeroMostrado
  end
end

dado1 = Dado.new
dado2 = Dado.new
puts dado1.rolar
puts dado2.rolar
puts dado1.mostrado
puts dado1.mostrado
puts dado2.mostrado
dado1.rolar
puts dado1.mostrado
```



Execução

rolar_dados.rb

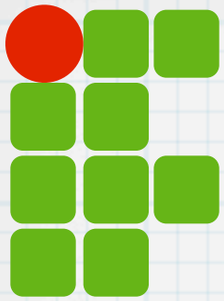
```
class Dado
  def rolar
    @numeroMostrado = 1 + rand(6)
    return @numeroMostrado
  end
  def mostrado
    return @numeroMostrado
  end
end

dado1 = Dado.new
dado2 = Dado.new
puts dado1.rolar
puts dado2.rolar
puts dado1.mostrado
puts dado1.mostrado
puts dado2.mostrado
dado1.rolar
puts dado1.mostrado
```



Execução

* O que acontece com o código abaixo?

```
dado1 = Dado.new
dado2 = Dado.new
puts dado1.rolar
puts dado1.mostrado
puts dado1.mostrado
puts dado2.mostrado
dado1.rolar
puts dado1.mostrado
```



Execução

* O que acontece com o código abaixo?

```
dado1 = Dado.new
dado2 = Dado.new
puts dado1.rolar
puts dado1.mostrado
puts dado1.mostrado
puts dado2.mostrado
dado1.rolar
puts dado1.mostrado
```

O método é chamado sem sem ter sido atribuído valor ao atributo

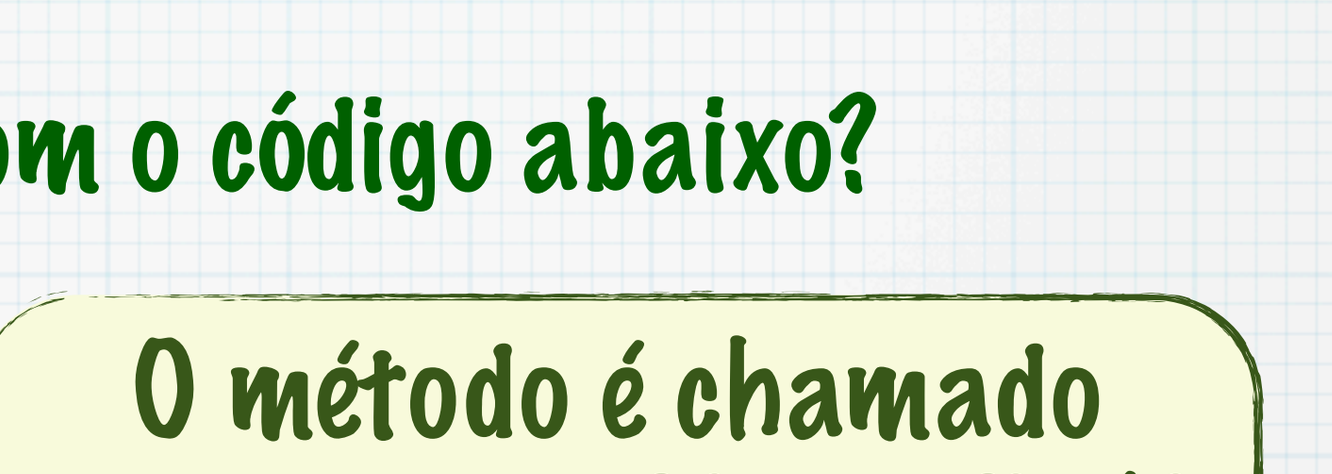
Execução

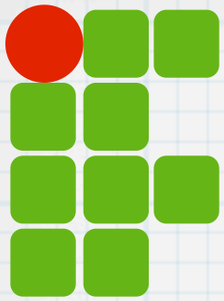
* O que acontece com o código abaixo?

```
dado1 = Dado.new
dado2 = Dado.new
puts dado1.rolar
puts dado1.mostrado
puts dado1.mostrado
puts dado2.mostrado
dado1.rolar
puts dado1.mostrado
```

O método é chamado sem sem ter sido atribuído valor ao atributo

Nada é mostrado

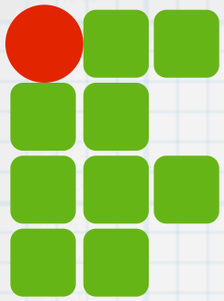




Exemplo

* Construtor
  * Método chamado no momento da criação do objeto (new)
  * Em Ruby o nome do construtor é SEMPRE `initialize`

```ruby
class Dado
  def initialize
    #  Eu vou apenas rolar o dado, apesar de
    #  podermos fazer qualquer coisa que
    #  queiramos fazer, como colocar a face '6'
    #  para cima
    rolar
  end
  def rolar
    @numeroMostrado = 1 + rand(6)
  end
  def mostrado
    @numeroMostrado
  end
end
```



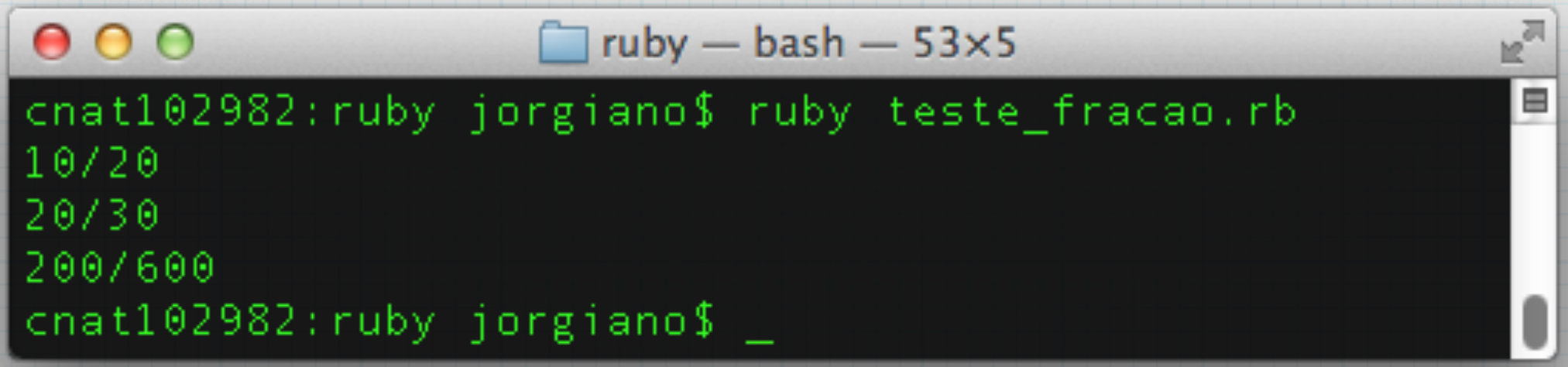
Exemplo

* Fração

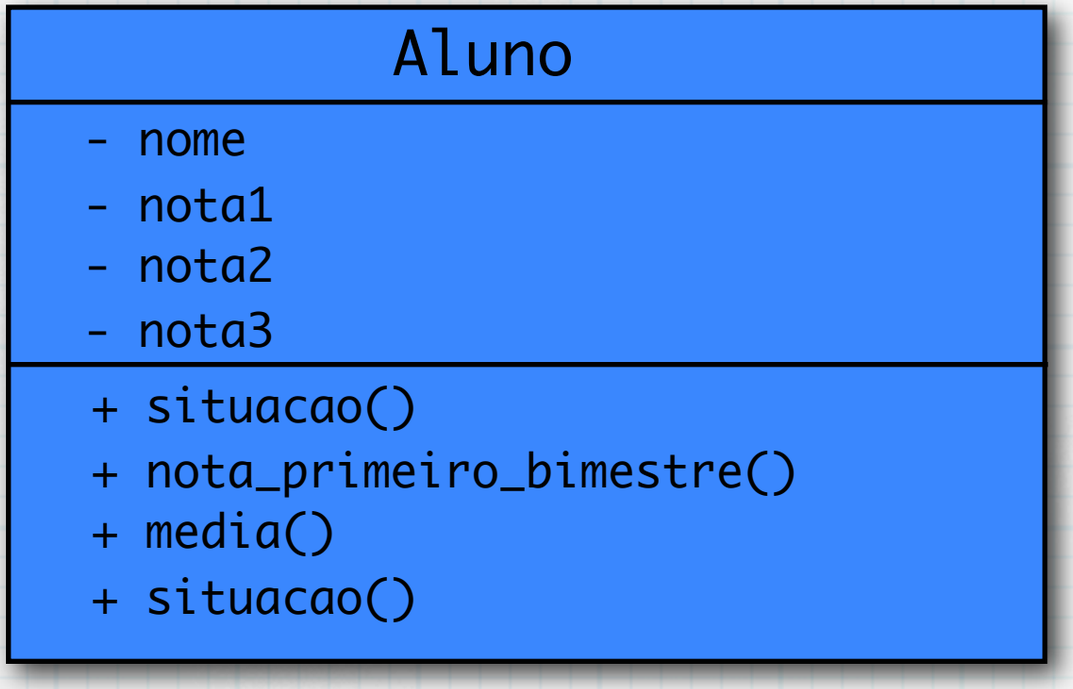
```
class Fracao
  def initialize (numerador, denominador)
    @numerador = numerador
    @denominador = denominador
  end
  def to_f
    return @numerador.to_f / @denominador.to_f
  end
  def to_s
    return @numerador.to_s + "/" + @denominador.to_s
  end
  def numerador
    return @numerador
  end
  def denominador
    return @denominador
  end
  def multiplica(outra)
    novo_numerador = outra.numerador * @numerador
    novo_denominador = outra.denominador * @denominador
    nova = Fracao.new(novo_numerador, novo_denominador)
    return nova
  end
end
```

Fração

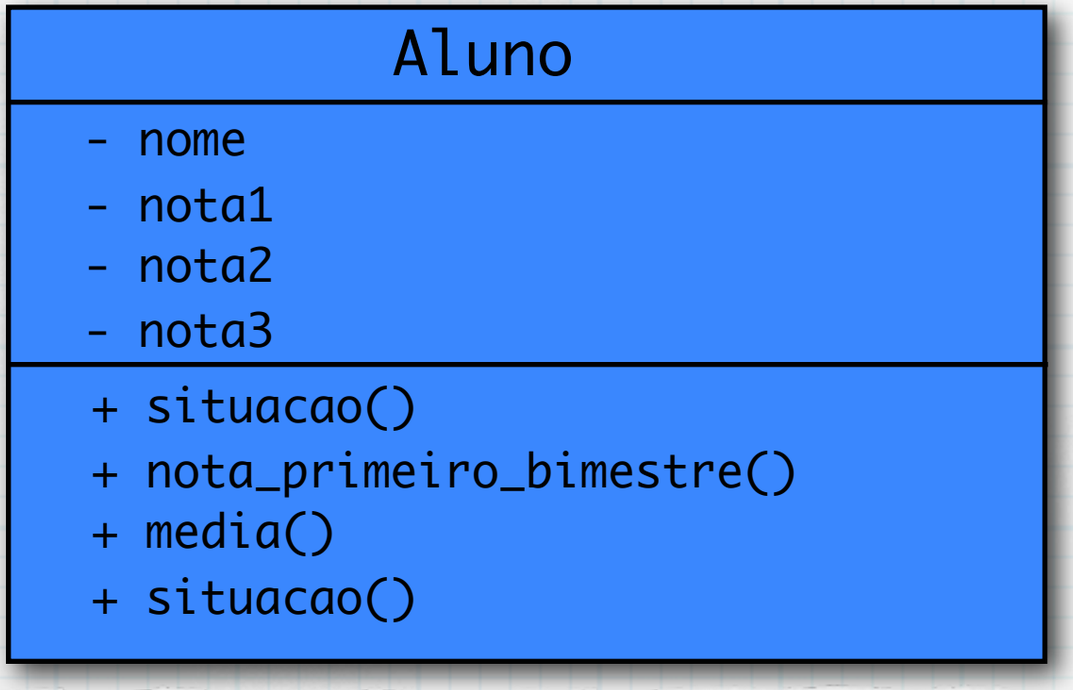
```
f1 = Fracao.new(10,20)
f2 = Fracao.new(20,30)
f3 = f1.multiplica(f2)
puts f1
puts f2
puts f3
```



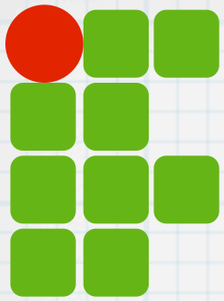
Exemplo


Aluno
- nome
- nota1
- nota2
- nota3
+ situacao()
+ nota_primeiro_bimestre()
+ media()
+ situacao()

Exemplo




```ruby
class Aluno
  def initialize(nome="Sem nome")
    @nome = nome
    @nota1 = 0 # Primeiro bimestre
    @nota2 = 0 # Segundo bimestre
    @nota3 = 0 # Recuperacao
  end
  def nota_primeiro_bimestre
    return @nota1
  end
# ...
```



Exemplo

Aluno
- nome - nota1 - nota2 - nota3
+ situacao() + nota_primeiro_bimestre() + media() + situacao()

```ruby
class Aluno
  def initialize(nome="Sem nome")
    @nome = nome
    @nota1 = 0 # Primeiro bimestre
    @nota2 = 0 # Segundo bimestre
    @nota3 = 0 # Recuperacao
  end
  def nota_primeiro_bimestre
    return @nota1
  end
# ...
```

```ruby
# ...
def definir_nota_primeiro_bimestre (nota)
    if (nota.class==Float and
        nota >=0 and nota <=10.0) then
      @nota1=nota
    end
  end
  def media
    m = (@nota1*2+@nota2*3)/5
    if (m<6.0 and m>2.0 and nota3>0) then
      # Cálculo da média
    end
    return m
  end
  def situacao
    s = "Matriculado"
    if (...) then
      s = "Aprovado"
    elsif (...)
      s = "Em recuperação"
    else
      s = "Reprovado"
  end
  return s
end
```

Dúvidas?