Um Toque de Classe: Objetos com Atributos

PROGRAMAÇÃO DE COMPUTADORES

O que veremos hoje

Atributos

- Leitura (attr ou attr_reader)
- Escrita (attr_writer)
- Leitura e Escrita (attr_accessor)

Métodos Getter e Setter

Métodos Publicos e Privados

Self (o “eu” de cada um)

Atributos

Por padrão, os atributos de um objeto não são visíveis por objetos de outras classes.

```
class Conta
  def initialize
    @senha = rand(100000)
  end
end

conta = Conta.new()
puts conta.senha
```

Atributos

Mas em muitos casos os atributos representam características de um objeto.

- cor
- portas
- placa

Características normalmente devem ser vistas e, eventualmente, modificadas externamente.

Definição de atributos

Atributos só de leitura

- `attr :cor, :porta`

- `attr` é uma função que recebe com parâmetros os nomes dos atributos que podem ser lidos externamente
  - `attr(:cor, :porta)`

```
class Carro
  attr :cor, :portas, :placa

  ...
end

meu_carro = Carro.new()
puts meu_carro.cor

meu_carro.cor = azul
```

Definição de atributos

Atributos de leitura e escrita
- `attr_accessor :cor`

Atributos só de escrita
- `attr_writer :placa`

```
class Carro
  attr_accessor :cor
  attr_writer :placa
end

meu_carro = Carro.new()

meu_carro.cor = azul
puts meu_carro.cor

meu_carro.placa='IFRN-001'
puts meu_carro.placa
```

Métodos Getter e Setter

As funções `attr`, `attr_accessor` e `attr_writer` criam, implicitamente,
métodos getter e setter

```ruby
class Carro
  attr_accessor :cor
  attr_writer :placa
  attr :portas
end
```

=

```ruby
class Carro
  def cor
    return @cor
  end
  def cor=(valor)
    @cor = valor
  end
  def portas
    return @portas
  end
  def placa=(valor)
    @placa = placa
  end
end
```

Criando métodos Getter e Setter

Os métodos getter e setter podem ser redefinidos pelo programador

```
class Aluno
  attr_accessor :nome
  attr :nota1, :nota2
  def initialize(nome)
    @nome = nome
  end
  def nota1=(valor)
    if (valor>=0.0 and valor<=10.0) then
      @nota1 = valor
    end
  end
  def nota1
    if @nota1>=5.0 then @nota1
    else "Nota baixa" end
  end
end
```

```
> aluno = Aluno.new("João")
> aluno.nota1 = 9.0
> puts aluno.nota1
9.0
> aluno.nota1 = 12.0
> puts aluno.nota1
9.0
> aluno.nota1 = 3.0
> puts aluno.nota1
Nota baixa
```

Métodos Públicos e Privados

Por padrão os métodos são visíveis externamente

Mas alguns métodos só fazem sentido para o próprio objeto.

A instrução **`private`** torna os métodos definidos na sequência como métodos privados, não visíveis por objetos de outras classes.

Uma boa prática de programação é ter poucos métodos públicos.

```
class Carro
  def acelerar
    passar_marcha
    vel = vel + 5
  end

  private
  def passar_marcha
  ..
  end
end
```

Self: o “eu” de cada um

Cada objeto chama a si próprio de **self**


```
class Fracao
  attr :num, :den

  def initialize(num,den)
    @num, @den = num, den
  end

  def multiplicar(outra)
    num = self.num * outro.num
    den = self.den * outro.den
    return Fracao.new(num, den)
  end
end
```

Duvidas ?