INF600A-20 Quiz formatif no. 6
6 novembre 2018

Note : Dans **la description de l'API** d'une classe Ruby `Foo` (et non pas dans du code Ruby!) :

- `Foo.bar` : méthode **de classe** `bar` ;
- `Foo#bar` : méthode **d'instance** `bar`.

---

1. Une classe `Foo` avec méthodes/attributs de classe/instance

Soit la classe suivante, définie dans le fichier `./foo.rb` :

```
class Foo
  @x = "abc"

  def x
    puts "Dans Foo#x: self = #{self} (#{self.class})"
    @x
  end

  def Foo.x
    puts "Dans Foo.x: self = #{self} (#{self.class})"
    @x
  end

  puts "end de Foo: self = #{self} (#{self.class})"
end
```

Qu'est-ce qui sera produit par l'exécution, dans `irb`, des expressions suivantes :

```
>> require_relative 'foo'     # require './foo'
end de Foo: self = Foo (Class)
=> true

>> require_relative 'foo'
=> false

>> o = Foo.new
=> #<Foo:0x00000002acada8>

>> o.x
Dans Foo#x: self = #<Foo:0x00000002acada8> (Foo)
=> nil

>> Foo.x
Dans Foo.x: self = Foo (Class)
=> "abc"
```

2. Une classe pour des Documents

```
class Document
  attr_reader :isbn, :titre, :auteurs

  def initialize( isbn, titre, auteurs )
    @isbn = isbn; @titre = titre; @auteurs = auteurs
  end
  private_class_method :new

  def self.creer( isbn, titre, *auteurs )
    @documents ||= {}
    @documents[isbn] = new( isbn, titre, auteurs )
  end

  def self.document( cle, sorte_de_recherche = :isbn )
    case sorte_de_recherche
    when :isbn then @documents[cle]
    when :titre then @documents.each { |_,d| return d if /#{cle}/i =~ d.titre }
    end
  end

  def to_s
    "#<#{self.class}: '#{titre}', #{@auteurs.join(' et ')} (#{isbn})>"
  end
end

>> puts d0 = Document.creer( :"10-99", "Ruby Optimization", "Dymo" )
#<Document: 'Ruby Optimization', Dymo (10-99)>
=> nil

>> puts d1 = Document.creer( :"11-77", "Programming Ruby", "Hunt", "Thomas" )
#<Document: 'Programming Ruby', Hunt et Thomas (11-77)>
=> nil

>> puts Document.document( :"10-99" )
#<Document: 'Ruby Optimization', Dymo (10-99)>
=> nil

>> Document.document( "11-77" )
=> nil

>> Document.document( "OPTI", :titre )
=> #<Document:0x00000001168a90 @isbn=:"10-99", @titre="Ruby Optimization", @auteurs=["Dymo"]>
```

3. Une classe pour des `Documents` (suite)

Supposons qu'on modifie (qu'étende, i.e., on réouvre) la classe `Document` comme suit :

```
class Document
  extend Enumerable # Comme include mais pour une classe

  def self.each
    @documents.each_pair do |_, doc|
      yield( doc )
    end
  end
end
```

Qu'est-ce qui sera imprimé par les trois derniers appels (`map` et `flat_map`) si on exécute, dans un état initial **vide** (sans aucun document encore créé), les expressions suivantes :

```
>> Document.creer(:"10-99", "Ruby Optimization", "Dymo")
=> #<Document:[...] @isbn=:"10-99", @titre="Ruby Optimization", @auteurs=

>> Document.creer(:"11-77", "Programming Ruby", "Hunt", "Thomas")
=> #<Document:[...] @isbn=:"11-77", @titre="Programming Ruby", @auteurs=[

>> Document.map { |d| d.isbn }
=> [:"10-99", :"11-77"]


>> Document.map(&:auteurs)
=> [["Dymo"], ["Hunt", "Thomas"]]


>> Document.flat_map(&:auteurs)
=> ["Dymo", "Hunt", "Thomas"]
```