Laboratoire #1 : Fichiers `ensemble-imperatif.rb` et `ensemble-enumerable.rb`

Fichier `ensemble-imperatif.rb`

```
  def un_element
    fail "L'ensemble est vide" if cardinalite == 0

    @elements.first
  end

  def max
    fail "L'ensemble est vide" if cardinalite == 0

    m = @elements[0]
    @elements.each do |x|
      m = [m, x].max
    end

    m
  end

  #
```

```ruby
def union( autre )
  res = Ensemble.new( *@elements )
  autre.each do |x|
    res << x
  end

  res
end

def intersection( autre )
  res = Ensemble.new
  @elements.each do |x|
    res << x if autre.contient?(x)
  end

  res
end

def <=( autre )
  @elements.each do |x|
    return false unless autre.contient?(x)
  end

  true
end

def to_a
  res = []
  @elements.each do |x|
    res << x
  end

  res
end

#
```

```ruby
def select_lambda( predicat )
  res = Ensemble.new
  @elements.each do |x|
    res << x if predicat.call(x)
  end

  res
end

def select
  res = Ensemble.new
  @elements.each do |x|
    res << x if yield(x)
  end

  res
end

def partitionner( pivot )
  petits = @elements.select { |x| x < pivot}
  egaux = @elements.select { |x| x == pivot}
  grands = @elements.select { |x| x > pivot}

  [
   Ensemble.new(*petits),
   Ensemble.new(*egaux),
   Ensemble.new(*grands),
  ]
end

def trier
  return [] if cardinalite == 0
  return [un_element] if cardinalite == 1

  petits, egaux, grands = partitionner(un_element)

  petits.trier + egaux.to_a + grands.trier
end
```

Fichier ensemble-enumerable.rb

```ruby
  def un_element
    fail "L'ensemble est vide" if cardinalite == 0

    @elements.first
  end

  def max
    fail "L'ensemble est vide" if cardinalite == 0

    @elements.max
  end

  def union( autre )
    Ensemble.new( *(@elements + autre.to_a) )
  end

  def intersection( autre )
    Ensemble.new( *select { |x| autre.contient?(x) } )
  end

  def <=( autre )
    all? { |x| autre.contient?(x) }
  end

  def to_a
    @elements.clone
  end

  def select_lambda( predicat )
    Ensemble.new( *select { |x| predicat.call(x) } )
  end

  def select
    Ensemble.new( *@elements.select { |x| yield(x) } )
  end

  #
```

```
def partitionner( pivot )
  petits = select { |x| x < pivot }
  egaux = select { |x| x == pivot }
  grands = select { |x| x > pivot }

  # NOTE: les .to_a pourraient etre omis, car il y a appel implicite
  # a to_a lors d'une utilisation de l'operateur prefixe '*'
  # (splat).
  [
   Ensemble.new(*petits.to_a),
   Ensemble.new(*egaux.to_a),
   Ensemble.new(*grands.to_a),
  ]
end

def trier
  return [] if cardinalite == 0
  return [un_element] if cardinalite == 1

  petits, egaux, grands = partitionner(un_element)

  petits.trier + egaux.to_a + grands.trier
end
```

Comparaisons ensemble-imperatif.rb/ensemble-enumerable.rb

Fichier ensemble-imperatif.rb

```ruby
def intersection( autre )
  res = Ensemble.new
  @elements.each do |x|
    res << x if autre.contient?(x)
  end
   res
end
```

Fichier ensemble-enumerable.rb

```ruby
def intersection( autre )
  Ensemble.new( *select { |x| autre.contient?(x) } )
end
```

```ruby
def <=( autre )
  @elements.each do |x|
    return false unless autre.contient?(x)
  end

  true
end
```

```ruby
def <=( autre )
  all? { |x| autre.contient?(x) }
end
```

```ruby
def select
  res = Ensemble.new
  @elements.each do |x|
    res << x if yield(x)
  end

  res
end
```

```ruby
def select
  Ensemble.new( *@elements.select { |x| yield(x) } )
end
```