Nombre y Apellido: ..... N° Legajo: .....

Primer Parcial de Programación Orientada a Objetos (72.33)

17/04/2019

Ejercicio 1	*Ejercicio 2*	*Ejercicio 3*	*Nota*	*Firma Docente*
/3	/4	/3		

- **<u>Condición mínima de aprobación: SUMAR 5 PUNTOS.</u>**
- **Las soluciones que no se ajusten al paradigma OO, no serán aceptadas.**
- **Las soluciones que no se ajusten estrictamente al enunciado, no serán aceptadas.**
- **Puede entregarse en lápiz.**
- **No es necesario escribir las sentencias `require`.**
- **Además de las clases solicitadas se pueden agregar las que consideren necesarias.**
- **Escribir en cada hoja Nombre, Apellido, Legajo, Número de Hoja y Total Hojas entregadas.**
- **Resolver los ejercicios en hojas separadas.**

Ejercicio 1

Un **iterador paralelo** es un iterador que, a partir de dos colecciones, permite recorrerlas paralelamente a ambas hasta que se alcance el final de una de las colecciones.

La clase **`ParallelIterator`** recibe en su constructor las dos colecciones a iterar y cuenta con un método `each` para poder acceder al iterador paralelo.

Implementar todo lo necesario para que el siguiente programa de prueba imprima la salida indicada:

```ruby
v1 = ['hola', 'mundo', 'adios']
v2 = ['hello', 'world']

my_iterator = ParallelIterator.new(v1, v2)
p my_iterator.each.next
p my_iterator.each.next
begin
 p my_iterator.each.next
rescue StopIteration => e
 puts e.message
end

puts '##########'

p my_iterator.each.take(4)

puts '##########'

begin
 ParallelIterator.new(nil, v2)
rescue ArgumentError => e
 puts e.message
end

puts '##########'

begin
 ParallelIterator.new(v1, nil)
rescue ArgumentError => e
 puts e.message
end
```

```
["hola", "hello"]
["mundo", "world"]
iteration reached an end
##########
[["hola", "hello"], ["mundo", "world"]]
##########
First collection missing
##########
Second collection missing
```

Ejercicio 2

Se cuenta con un conjunto de clases que modelan el proceso de envío de mensajes de texto. La clase **`PhoneCentral`** es la central que permite la designación de números de teléfono. La clase **`CellPhone`** modela el teléfono y la clase **`Message`** modela el mensaje.

```ruby
class PhoneCentral
 def initialize
   @phones = Hash.new
 end

 def new_phone(number)
   @phones[number] = CellPhone.new(number,
self)
 end

 def route_message(message)
   cellphone = @phones[message.to]
   raise UnknownRecipientError if
cellphone.nil?
   cellphone.receive_message(message)
 end
end
```

```ruby
class Message
 attr_reader :from, :to, :text

 def initialize(from, to, text)
   @from, @to, @text = from, to, text
 end
end

class UnknownRecipientError < StandardError
end
```

```ruby
class CellPhone
 def initialize(number, phone_central)
   @number, @phone_central = number, phone_central
 end

 def receive_message(message)
   puts "FROM: #{message.from} TO: #{message.to} MSG: #{message.text}"
 end

 def send_message(to, text)
   @phone_central.route_message(Message.new(@number, to, text))
 end
end
```

Se pide implementar la clase **`EnhancedPhoneCentral`** que modela a una central que sea capaz de registrar estadísticas de envío de mensajes. Las estadísticas corresponden a cantidad de caracteres enviados, cantidad de mensajes enviados y cantidad de mensajes fallidos (por error en el número destino).

Implementar todo lo necesario para que, con el siguiente programa de prueba:

```ruby
central = EnhancedPhoneCentral.new
phones = []
(0...4).each_with_index {|i| phones[i] = central.new_phone("111#{i}")}

begin
 phones[0].send_message('1111', 'Hola')
 phones[1].send_message('1110', 'Que tal?')
 phones[2].send_message('1110', 'Buen dia')
 phones[2].send_message('1117', 'Buen dia')
rescue UnknownRecipientError
 puts 'Alguna llamada falló'
end

puts central.phone_stats
```

se obtenga la siguiente salida:

```
FROM: 1110 TO: 1111 MSG: Hola
FROM: 1111 TO: 1110 MSG: Que tal?
FROM: 1112 TO: 1110 MSG: Buen dia
Alguna llamada falló
Teléfono: 1110 Caracteres enviados: 4 Mensajes enviados: 1 Mensajes fallidos: 0
Teléfono: 1111 Caracteres enviados: 8 Mensajes enviados: 1 Mensajes fallidos: 0
Teléfono: 1112 Caracteres enviados: 8 Mensajes enviados: 1 Mensajes fallidos: 1
Teléfono: 1113 Caracteres enviados: 0 Mensajes enviados: 0 Mensajes fallidos: 0
```

Ejercicio 3

HTML es un lenguaje utilizado en el desarrollo de páginas web. Permite describir la estructura de un documento, así como darle formato. Para hacer esto, utiliza "etiquetas" que permiten indicar el formato a aplicar. A continuación se describen algunas de estas etiquetas:

- **b**: Permite definir un texto en **negrita**. Ejemplo: `<b>hola</b>`
- **i**: Permite definir un texto en **cursiva**. Ejemplo: `<i>hola</i>`
- **a**: Permite definir un **enlace**, agregando un atributo que indica la página a la cual se desea ir al hacer clic en el enlace. Ejemplo: `<a href="www.itba.edu.ar">Ir a la página del ITBA</a>`

Estas etiquetas pueden anidarse para combinar distintos formatos. Por ejemplo, el siguiente texto HTML muestra la cadena de texto "hola" en negrita y en cursiva: `<b><i>hola</i></b>`. El siguiente código hace lo mismo: `<i><b>hola</b></i>.`

Se cuenta con un módulo `HTMLText` que representa un texto HTML y provee un método para obtener el código fuente. Se cuenta además con una implementación para textos sin formato (en los cuales el código fuente coincide con el texto a mostrar, ya que no se aplica ninguna etiqueta).

```ruby
module HTMLText
 def source
   raise 'Not implemented'
 end

 def to_s
   source
 end
end
```

```ruby
class PlainText
 include HTMLText

 attr_writer :content

 def initialize(content)
   @content = content
 end

 def source
   @content
 end
end
```

Se quiere ofrecer más funcionalidad para permitir representar textos en negrita, en cursiva y enlaces.

Implementar todo lo necesario para que, con el siguiente programa de prueba, se obtenga la salida indicada en los comentarios.

```ruby
text = PlainText.new('Hola')
bold_text = text.bold
italic_text = text.italic
puts bold_text # <b>Hola</b>
puts italic_text # <i>Hola</i>
bold_italic_text = italic_text.bold
puts bold_italic_text # <b><i>Hola</i></b>
text.content = 'ITBA'
puts bold_text # <b>ITBA</b>
puts italic_text # <i>ITBA</i>
puts bold_italic_text # <b><i>ITBA</i></b>
link_text = text.link('www.itba.edu.ar')
link_bold_italic_text = bold_italic_text.link('www.itba.edu.ar')
bold_link_text = text.link('www.itba.edu.ar').bold
puts link_text # <a href:"www.itba.edu.ar">ITBA</a>
puts link_bold_italic_text # <a href:"www.itba.edu.ar"><b><i>ITBA</i></b></a>
puts bold_link_text # <b><a href:"www.itba.edu.ar">ITBA</a></b>
text.content = 'Ejemplo'
puts link_bold_italic_text # <a href:"www.itba.edu.ar"><b><i>Ejemplo</i></b></a>
puts bold_link_text # <b><a href:"www.itba.edu.ar">Ejemplo</a></b>
```