INF600A-20 Quiz formatif no. 7
27 novembre 2018

---

1. Éléments de base de métaprogrammation

Soit la classe `Foo` définie ci-dessous. Indiquez ce qui sera imprimé par les appels à `p` ci-bas — dans l'ordre indiqué.

```ruby
class Foo
  def yy( x )
    yield( yield(x) )
  end
end

def Foo.yy( x )
  yield(x) + yield(x)
end
```

```ruby
class Foo
  def self.def_ms( *symbs )
    symbs.each do |symb|
      define_method(symb) do
        instance_variable_get "@#{symb}"
      end

      define_method("#{symb}=") do |v|
        instance_variable_set "@#{symb}", v
      end
    end
  end
end
```

```ruby
#a.
p Foo.send( "y" * 2, 2 ) { |x| x * x }


# b.
Foo.class_eval( "def self.yy( x ) yield(x) * yield(x); end" )
p Foo.yy(2) { |x| x * x }


#c.
p (f1 = Foo.new).yy(3) { |x| x * x }


# d.
a = [:foo, "bar"]
Foo.def_ms(*a)
f1.foo = 0; f1.bar = 99
p a.map { |m| f1.send(m) }


# e.
p a.map { |m| f1.instance_eval("@#{m}= 33"); f1.send(m) }
```

2. Définition de classes par métaprogrammation

Soit la classe `TupleClass` ci-dessous. Indiquez ce qui sera imprimé par les appels à `p` et `puts` ci-bas — dans l'ordre indiqué.

```ruby
class TupleClass
  def self.new( taille )
    klass = Class.new

    klass.define_singleton_method(:[]) do |*vals|
      tc = klass.new
      tc.instance_variable_set "@vals", vals

      tc
    end

    (0...taille).each do |k|
      klass.send(:define_method, "_#{k+1}" ) do
        @vals[k]
      end
    end

    klass.send(:define_method, :to_s) { "(#{@vals.join(', ')})" }

    klass
  end
end
```

```ruby
# a.
Paire = TupleClass.new(2)
p Paire.class
p Paire


# b.
p1 = Paire[10, 20]
p p1.class
puts p1

# c.
p p1._1

# d.
Point = TupleClass.new(3)
p Point[1.0, 0.0, 9.9]
```