Pauta Suplemento I2:

```ruby
class Coffee
  # type can be 'espresso', 'latte', or 'cappuccino'
  def initialize(type)
    @type = type
  end

  def description
    "#{@type}"
  end

  def contains_decorator?(decorator_tag)
    false
  end

  def match(decorator_tag, type)
    false
  end
end

class CoffeeDecorator
  attr_accessor :decorator

  def initialize(decorator)
    @decorator = decorator
  end

  def contains_decorator?(decorator_tag)
    if tag == decorator_tag
      true
    elsif @decorator.is_a? Coffee
      false
    else
      @decorator.contains_decorator?(decorator_tag)
    end
  end

  def match(decorator_tag, type)
    decorator_tag == tag
  end

end

class SweetDecorator < CoffeeDecorator
  attr_accessor :type

  def description
    if @type == 'sugar'
      "#{@decorator.description}\nAdd sugar"
    else
      "#{@decorator.description}\nAdd sweetener"
    end
  end

  def tag
    'sweet'
  end

  def match(decorator_tag, type)
    decorator_tag == tag && type == @type
  end
end

class StrawberryDecorator < CoffeeDecorator
  def description
    "#{@decorator.description}\nAdd shot of strawberry"
  end

  def tag
    'strawberry'
  end
end

class VanillaDecorator < CoffeeDecorator
  def description
    "#{@decorator.description}\nAdd shot of vanilla"
  end

  def tag
    'vanilla'
  end
end

class CreamDecorator < CoffeeDecorator
  def description
    "#{@decorator.description}\nAdd cream"
  end

  def tag
    'cream'
  end
end

def add_decorator(object, decorator_tag, type)
  if decorator_tag == 'sweet'
    decorator = SweetDecorator.new(object)
    decorator.type = type
  elsif decorator_tag == 'strawberry'
    decorator = StrawberryDecorator.new(object)
  elsif decorator_tag == 'vanilla'
    decorator = VanillaDecorator.new(object)
  else
    decorator = CreamDecorator.new(object)
  end

  decorator
end

def remove_decorator(decorated_object, decorator_tag, type)
  parent_decorator = nil
  object = decorated_object
  until object.is_a? Coffee
    if object.match(decorator_tag, type)
      if parent_decorator.nil?
        return decorated_object.decorator
      else
        parent_decorator.decorator = object.decorator
        return decorated_object
      end
    end
    parent_decorator = object
    object = object.decorator
  end
  decorated_object
end

coffee_type = ''
puts "Insert coffee type ('espresso, latte or cappuccino):"
loop do
  coffee_type = gets.strip
  if ['espresso', 'latte', 'cappuccino'].include? coffee_type
    break
  else
    puts "Invalid coffee type"
  end
end

coffee = Coffee.new(coffee_type)
puts "Customize coffee:"
loop do
  command = gets.strip
  if command == 'end'
    break
  end

  command_parts = command.split(' ')
  if command_parts.length != 2 or
     not ['add', 'remove'].include? command_parts[0] or
     not ['sugar', 'sweetener', 'strawberry', 'vanilla', 'cream'].include?
command_parts[1]
    puts 'Invalid command'
    next
  end

  decorator_tag = command_parts[1]
  type = nil
  if ['sugar', 'sweetener'].include? decorator_tag
    type = decorator_tag
    decorator_tag = 'sweet'
  end

  if command_parts[0] == 'add'
    unless coffee.contains_decorator? decorator_tag
      coffee = add_decorator(coffee, decorator_tag, type)
    end
  else
    coffee = remove_decorator(coffee, decorator_tag, type)
  end
end

puts coffee.description
```

Nota de Pauta:

Desglose de puntaje:

- Jerarquía de decorators
    - 1 punto
- Método para añadir nuevos decoradores revisando casos bordes
    - 2 puntos
- Método para eliminar decoradores revisando casos bordes
    - 2 puntos
- Finalizar script que parsea comandos ingresados por el usuario e imprimir descripción de la bebida final
    - 1 punto