Nombre y Apellido: Nº Legajo:

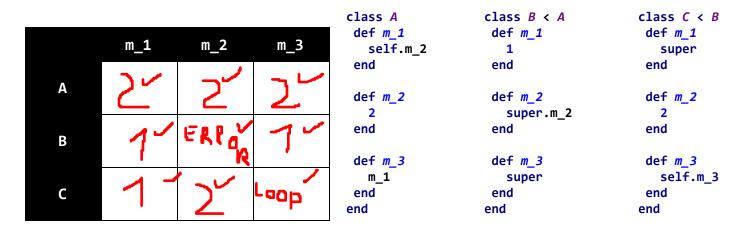
Primer Parcial de Programación Orientada a Objetos (72.33) 12/04/2018

Ejercicio 1	Ejercicio 2	Ejercicio 3	Nota	Firma Docente

- ♦ Condición mínima de aprobación: Tener BIEN o BIEN- dos de los tres ejercicios.
- 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.**

Ejercicio 1

Dada la siguiente jerarquía de clases, con los métodos de instancia indicados para cada una, se cuenta con tres instancias homónimas a la clase a la cual pertenecen.



Completar el cuadro de doble entrada (clase y mensaje) indicando qué se obtiene al enviar cada uno de los mensajes a instancias de cada una de las clases.

Ejercicio 2 🏑

Se cuenta con la siguiente familia de clases que representan **elementos** que pueden imprimirse en la consola:

```
module Element
                                                    class TextElement
def contents
                                                     include Element
   raise 'Not Implemented'
                                                     def initialize(text)
end
                                                      @text = text
def to_s
                                                     end
  contents
end
                                                     def contents
                                                       "#{@text}\n"
end
                                                     end
                                                    end
class UniformElement
```

```
class UniformElement
include Element

def initialize(element, width, height)
    @element = element
    @width = width
    @height = height
end
```

```
def contents
  content = ''
  (1..@height).each do
    (1..@width).each do
    content += @element
    end
    content += "\n"
  end
  content
end
end
```

Se quiere dar la posibilidad al usuario de formar **nuevos elementos combinando otros ya existentes**. Para esto se pueden agregar líneas de código a los fuentes de Element, UniformElement y/o TextElement o bien crear clases nuevas, de forma tal que el siguiente programa de prueba imprima lo que se indica a continuación:

```
elem1 = UniformElement.new('+', 6, 2)
                                                    elem4:
elem2 = TextElement.new('hola')
                                                    +++++
elem3 = TextElement.new('mundo')
                                                    +++++
                                                    hola
elem4 = elem1.above(elem2)
elem5 = elem3.below(elem2)
puts "elem4:\n#{elem4}\n"
                                                    elem5:
puts "elem5:\n#{elem5}\n"
                                                    hola
elem1.element = '.'
                                                    mundo
elem3.text = 'adios'
puts "elem4:\n#{elem4}\n"
                                                    elem4:
puts "elem5:\n#{elem5}\n"
                                                    . . . . . .
                                                    hola
                                                    elem5:
                                                    hola
                                                    adios
```

Ejercicio 3

Para aumentar las ventas, el dueño del comercio desea ofrecer un descuento de acuerdo a la categoría del producto a vender. El cuadro de descuentos es el siguiente:

- Electrodomésticos: 25% de descuento para aquellos productos que tengan un precio de lista de hasta \$5000
- Electrónica: 10% de descuento para todos los productos, sin tope

En el siguiente programa de prueba se agregan al catálogo de la tienda dos productos, indicando el nombre del mismo, la categoría a la cual pertenece y el precio de lista sin descuento del producto.

Tener en cuenta que en futuro pueden incorporarse más categorías con sus descuentos particulares. Diseñar para lograr la máxima reusabilidad posible.

Implementar todo lo necesario y completar los ... para que el siguiente programa de prueba produzca la salida indicada en los comentarios: