TALLER OBJETOS III

Link Github

- Crea una carpeta y guarda cada archivo .rb con el número de la pregunta de manera local con **Sublime** o **Atom**.
- Luego guarda los cambios y súbelos a tu repositorio de Github.
- Luego de pusheados los últimos cambios, sube el link de Github en el desafío de la sección correspondiente en la plataforma.

Ejercicio 1: Sintaxis

Corregir los errores para poder ejecutar ambos métodos.

```
class MiClase
    def de_instancia
        puts 'Método de instancia!'
    end
    def.self de_clase
        puts 'Método de clase!'
    end
end

MiClase.de_instancia
MiClase.new.de_clase
```

Ejercicio 2: Sintaxis

Corregir los errores de sintaxis para que las últimas cuatro líneas se ejecuten de manera correcta.

> La última instrucción debe imprimir *"Hola! Soy la clase MiClase"*

```
class MiClase
    attr_writer :name
    def initialize(name)
        @name = name
    end

    def self.saludar
        "Hola! Soy la clase #{@name}"
    end
end

c = MiClase.new('Clase Uno')
puts c.name
c.name = 'Nombre Nuevo'
puts MiClase.saludar
```

Ejercicio 3: Herencia

Se tiene la clase *Vehicle* que recibe como argumento un modelo y un año. El método *engine_start* enciende el vehículo.

```
class Vehicle
  def initialize(model, year)
    @model = model
    @year = year
    @start = false
  end

  def engine_start
    @start = true
  end
end
```

Se pide:

- Crear una clase *Car* que herede de *Vehicle*
- El constructor de *Car*, además de heredar las propiedades de *Vehicle*, debe incluir un contador de instancias de *Car*.
- Crear un método de clase en *Car* que devuelva la cantidad de instancias.
- El método *engine_start* heredado debe además imprimir *'El motor se ha encendido!'*.
- Instanciar 10 *Cars*.
- Consultar la cantidad de instancias generadas de *Car* mediante el método de clase creado.

Ejercicio 4: Método de clase

El archivo *notas.txt* contiene las notas de 4 alumnos.

```
David, 90, 60, 10, 30
Mai, 40, 34, 77, 11
Gonzalo, 34, 86, 55, 91
JP, 100, 100, 100, 99
```

La clase *Alumno* posee un constructor que recibe el nombre del alumno junto a sus cuatro notas.

```
class Alumno
  def initialize(nombre, nota1, nota2, nota3, nota4)
    @nombre = nombre
    @nota1 = nota1
    @nota2 = nota2
    @nota3 = nota3
    @nota4 = nota4
  end
end

alumnos = []
data = []
File.open('notas.txt', 'r') { |file| data = file.readlines }
data.each do |alumno|
  alumnos << Alumno.new(*alumno.split(', '))
end

print alumnos
```

Se pide:

- Crear un método de clase llamado *read_file* que reciba como argumento el nombre del archivo (por defecto debe ser 'notas.txt') y devuelva la colección de objetos.
- El método debe alojar las instrucciones que se encuentran después de la clase.
- Finalmente imprimir la colección de objetos generada.

> Hint: Debes reemplazar el argumento de *File.open* con el nombre del argumento del método *read_file*.

Ejercicio 5:

Se tienen las clases *Rectangulo* y *Cuadrado* cuyos constructores reciben las medidas de los lados correspondientes.

```
class Rectangulo
  def initialize(largo, ancho)
    @largo = largo
    @ancho = ancho
  end
end

class Cuadrado
  def initialize(lado)
    @lado = lado
  end
end
```

Se pide:

- Agregar un método de instancia llámado *lados* en ambas clases. El método debe imprimir un *string* con las medidas de los lados.
- Crear un método llamado *perimetro* que reciba dos argumentos (lados) y devuelva el perímetro.
- Crear un método llamado *area* que reciba dos argumentos (lados) y devuelva el área.
- Instanciar un *Rectangulo* y un *Cuadrado*.
- Imprimir el área y perímetro de los objetos instanciados utilizando los métodos implementados.