Actividad 020 - Módulos, Mixins y Rack.

- Para poder realizar este actividad debes haber realizado los cursos previos junto con los videos online correspondientes a la experiencia 10.
- Crea una carpeta y guarda cada archivo .rb con el número de la pregunta, siguiendo las instrucciones 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: 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 2:

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 módulo llamado *Formula*.
- Dentro del módulo *Formula* crear un método llamado *perimetro* que reciba dos argumentos (lados) y devuelva el perímetro.
- Dentro del módulo *Formula* crear un método llamado *area* que reciba dos argumentos (lados) y devuelva el área.
- Implementar -mediante *Mixin*- el módulo en las clases *Rectangulo* y *Cuadrado*.
- Instanciar un *Rectangulo* y un *Cuadrado*.
- Imprimir el área y perímetro de los objetos instanciados utilizando el método del módulo implementado.

Ejercicio 3: Rack

Se tiene el archivo *config.ru* :

```
# config.ru
require 'rack'
class MiPrimeraWebApp
  def call(env)
    if env['REQUEST_PATH'] == '/'
      [202, { 'Content-Type' => 'text/html' }, ['<h1> INDEX </h1>']]
    end
  end
end
run MiPrimeraWebApp.new
```

Crear un archivo llamado *404.html* cuyo *body* contenga una etiqueta de título con el texto *"No se ha encontrado la página :("*.

Modificar el archivo ***config.ru*** para adaptarlo a los siguientes requerimientos:

- Si se ingresa a la url ***/index***:
    - Agregar un código de respuesta ***200***.
    - Agregar en los *Response Headers* un *Content-type* de tipo *text/html*.
    - Agregar en el *Response Body* una etiqueta de título que contenga un texto *"Estás en el Index!"*.
- Si se ingresa a la url ***/otro***:
    - Agregar un código de respuesta ***200***.
    - Agregar en los *Response Headers* un *Content-type* de tipo *text/html*.
    - Agregar en el *Response Body* una etiqueta de título que contenga un texto *"Estás en otro landing!"*.
- Si se ingresa a cualquier otra url:
    - Agregar un código de respuesta ***404***.
    - Agregar en los *Response Headers* un *Content-type* de tipo *text/html*.
    - Agregar en el *Response Body* el archivo *404.html* creado al inicio.