Introducción a objetos

Introducción a objetos 1
- ¿Qué aprenderás? 2
- Objetos y métodos 2
  - Revisando la documentación 2
    - ¿Cómo se lee la documentación? 2
      - Leyendo la documentación 3
    - ¿Cómo ocupamos estos métodos? 4
  - Usando métodos de clase 4
  - Usando métodos de instancia 4
  - Utilizando el método .size 4
  - Métodos con opciones 5
- El retorno de un método 5
  - ¿Y los paréntesis? 5
  - Algunos parámetros son obligatorios 6
  - Existen diversas formas de escribir lo mismo 6
    - Ejercitando lo aprendido 6
      - Algoritmo 6
      - Código 7
- Resumen del capítulo 7

¡Comencemos!

¿Qué aprenderás?

- Leer la documentación de Ruby.
- Conocer la sintaxis de métodos y argumentos.

¡Vamos con todo!

Objetos y métodos

Hasta el momento hemos trabajado con tipos de datos Integer y String, estos tipos de datos reciben el nombre de clases. Mientras que los datos de un tipo en específico reciben el nombre de objetos.

Las operaciones que realizamos sobre ellos reciben el nombre de métodos. Los métodos nos permiten operar con los objetos bajo ciertas restricciones.

```
2 + 2 # => 4
'hola' + ' a todos' # 'hola a todos'
2 + 'hola' # TypeError: String can't be coerced into Integer
```

Por ejemplo, el método + nos permite sumar dos enteros, o concatenar dos strings, pero no nos permite sumar un string con un entero, porque no tiene sentido la idea de 'sumar un texto'.

Para saber cómo ocupar un método en específico o conocer otros métodos debemos ocupar la documentación.

Revisando la documentación

Consultar la documentación es un hábito que debemos adoptar. Todos los programadores la ocupan, incluyendo los expertos.

¿Cómo se lee la documentación?

Utilizaremos la documentación de ruby-doc. En esta página podemos ver la documentación con todos los objetos incluidos en Ruby.

Comencemos revisando uno de los objetos que más hemos utilizado: Los Integers.

Leyendo la documentación

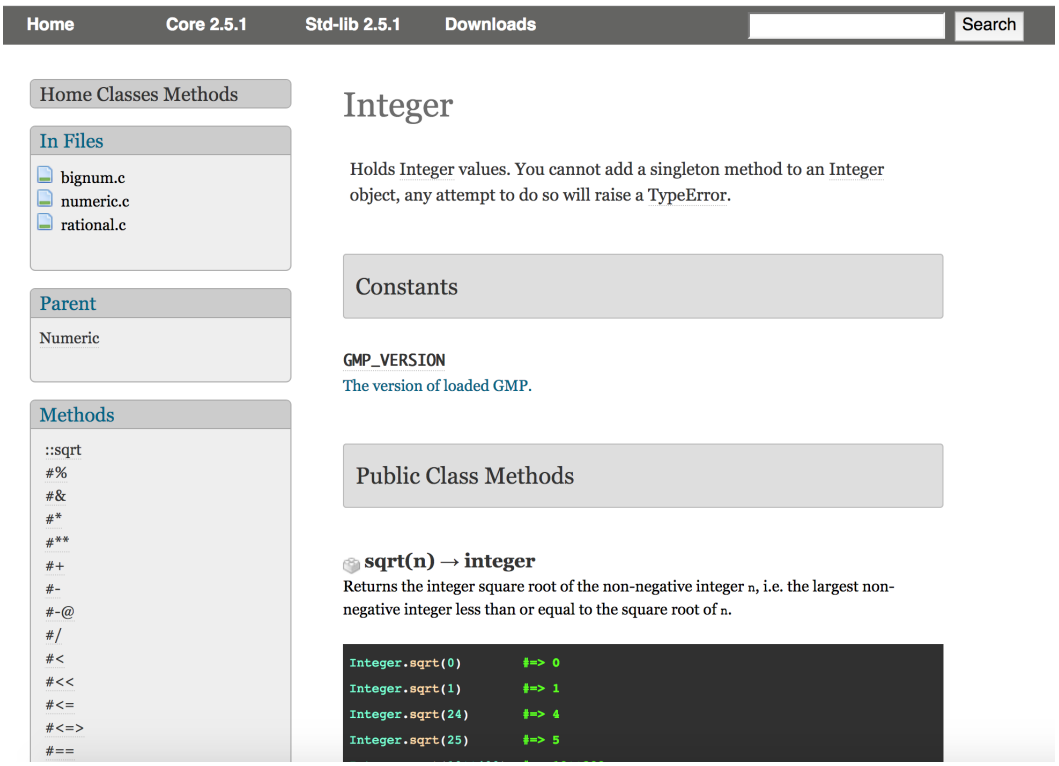

Imagen 1. Lectura de documentos.
Fuente: Desafío Latam

En la documentación veremos 2 columnas:

- La columna derecha nos muestra toda la documentación de los métodos.
- La columna izquierda nos muestra 3 secciones:
    - **In files:** Muestra los archivos donde está definido nuestro objeto.
    - **Parent:** Muestra de dónde heredó algunos de los comportamientos, en algunas ocasiones la consultaremos.
    - **Methods:** Corresponde a los métodos. Esta será la sección que consultaremos con mayor frecuencia.

Al seleccionar cualquiera de los métodos de la tercera sección, seremos redirigidos a la explicación del método en la columna derecha.

¿Cómo ocupamos estos métodos?

Existen dos tipos de métodos que se ocupan de forma muy parecida: Los métodos de clase y los métodos de instancia.

En la documentación se agrupan por el tipo de método. Por ejemplo, en el caso de la documentación de integer vemos que el método sqrt aparece bajo Public class Methods, mientras los métodos de instancia aparecen bajo Public Instance Methods

Usando métodos de clase

Para ocupar un método de clase utilizamos la sintaxis nombre del tipo de dato (nombre de la clase) + .metodo

Ejemplo:

```
Integer.sqrt(4) # 2
Time.now
```

Usando métodos de instancia

Para utilizar un método utilizaremos la sintaxis: objeto.método por ejemplo:

```
'paralelepipedo'.size
```

En la gran mayoría de los casos estaremos utilizando métodos de instancia.

Utilizando el método .size

Podemos ver en la documentación que el método .size del objeto String devuelve un entero con la longitud del String.

```
'Paralelepipedo'.size # => 14
```

Utilizar un método es sinónimo de llamarlo o invocarlo, el término más frecuentemente utilizado es llamar.

Métodos con opciones

También existen métodos que requieren recibir información para operar. Por ejemplo el método .count, del objeto String, recibe un substring (conjunto de caracteres más corto) para poder contar cuántas veces está contenido ese substring en el String principal.

Imaginemos que necesitamos saber cuántas letras 'p' tiene la palabra 'paralelepipedo':

```
'paralelepipedo'.count('p')
# => 3
```

El retorno de un método

Lo que devuelve un método se conoce como el retorno del método, y la información que recibe se conoce como parámetro.

De tal manera podemos decir que el método .count, del objeto String:

- Recibe como parámetro, al menos, un substring. Este es obligatorio.
- Retorna un entero correspondiente a la cantidad de veces que está contenido el substring en el objeto String.

```
'paralelepipedo'.count 'p'
# => 3
```

¿Y los paréntesis?

Cuando utilizamos un método, el uso de paréntesis es opcional siempre y cuando no haya ambigüedad en cómo se lea.

```
'paralelepipedo'.count('p')
```

Es lo mismo que:

```
'paralelepipedo'.count 'p'
```

Algunos parámetros son obligatorios

Dijimos que .count exigía un valor, probemos que sucede si no lo ingresamos.

```
'prueba'.count # ArgumentError: wrong number of arguments (given 0, expected 1+)
```

Existen diversas formas de escribir lo mismo

Algunos métodos, como el más (+), pueden ser utilizados sin el punto. Por ejemplo 2.+(2) es lo mismo que 2 +(2) y, dado que el paréntesis es opcional, esto es lo mismo que 2 + 2.

Ejercitando lo aprendido

Lo aprendido lo podemos utilizar para crear un pequeño programa donde el usuario introduce un valor y mostramos la cantidad de letras:
¿Qué método tenemos que ocupar .size o .count?

Algoritmo

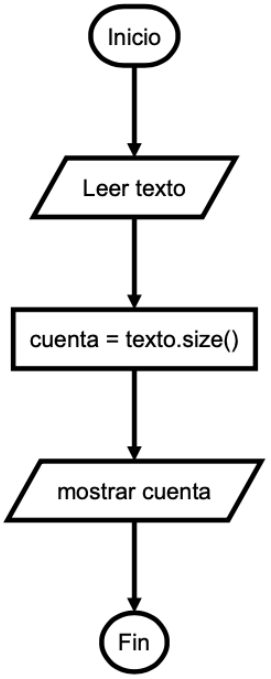

Imagen 2. Ejemplo de algoritmo.
Fuente: Desafío Latam

Código

```
text = gets.chomp # Leer texto
count = text.size # Recordemos que los paréntesis son opcionales
puts "El contenido tiene #{count} letras"
```

Todo ejercicio de programación tiene diversas formas de ser resuelto, por lo que el código de arriba es solo un ejemplo de solución.

Resumen del capítulo

En este capítulo aprendimos sobre objetos, métodos y cómo consultar la documentación.

- **Objetos**
    - En Ruby existen distintos tipos de datos. Un dato de cualquier tipo recibe el nombre genérico del objeto. El nombre formal en Ruby para el "tipo de dato" es clase.
    - Los objetos tienen métodos que nos permiten realizar diversas acciones sobre ellos.
- **Métodos**
    - Corresponden a las formas que tenemos de trabajar con un objeto.
    - Utilizar un método, llamar e invocar son sinónimos.
    - Sólo podemos llamar métodos que ya han sido definidos. De momento sólo hemos trabajado con métodos existentes que consultamos en la documentación, pero más adelante aprenderemos a definir nuestros propios métodos.
    - Los métodos pueden recibir valores (parámetros) y esto los vuelve flexibles.
- **Al consultar la documentación**
    - Nombre del método.
    - Tipo de método (de instancia o de clase).
    - Parámetros que recibe: opcionales y obligatorios.
    - Información que retorna.