JAVA: STRING MÉTODOS

MÉTODO CONSTRUCTOR *(creación de un objeto de tipo cadena, llamando a la clase String)*

- **String** miCadena = **new String("Mi cadena")**;

EXCEPCIONES: Si nuestra variable de tipo **String** almacena un valor **null** y queremos hacer uso de algunos métodos propuestos en este documento, nos lanzará una excepción de tipo → **NullPointerException** proveniente de la clase **NullPointerException** que se encuentra en el paquete **java** subpaquete **lang** tal que el acceso completo al documento de la excepción es → **java.lang.NullPointerException**.

EXCEPCIONES: Si en nuestro argumento ponemos un índice superior al límite de nuestra cadena, nos lanzará una excepción de tipo → **StringIndexOutOfBoundsException** proveniente de la clase **StringIndexOutOfBoundsException** que se encuentra en el paquete **java** subpaquete **lang** tal que el acceso completo al documento de la excepción es → **java.lang.StringIndexOutOfBoundsException**.

TIPO DE RETORNO: CHAR

MÉTODO → charAt()

Descripción

- Este método nos servirá para traernos el carácter solicitado perteneciente a nuestra cadena, lo traemos indicándole su posición/índice. Debido a que nuestra cadena es tratada por un conjunto de caracteres y también es accesible como si se tratase de un arreglo, comenzamos por el índice **0**.

FÓRMULA DEL MÉTODO: ***miCadena*.charAt(índice)**

USO del MÉTODO

EJEMPLO VÁLIDO → "hola". charAt(2) = 'l' ← Devuelve

Argumentos: **1-requerido (tipo cadena)** → Espera recibir como argumento la posición/índice del elemento/carácter.

CONSEJO: Si estamos interesados en traernos el último o el cercano próximo elemento/carácter de nuestra cadena, en lugar de ir por un número más alto, podemos hacer la siguiente fórmula → **miCadena.charAt(miCadena.length() - 1)** donde usaremos nuevamente nuestra "cadena" para medir su longitud, luego le restamos por **-1**, de esta forma traeremos el último elemento, luego podemos hacer **-2**, **-3**, etc. y ya sabremos que serán elementos cercanos al final de nuestra cadena.

NOTA: Si estamos interesados en traernos el último o el cercano próximo elemento/carácter de nuestra cadena, en lugar de ir por un número más alto, podemos hacer la siguiente fórmula → **miCadena.charAt(miCadena.length() - 1)** donde usaremos nuevamente nuestra "cadena" para medir su longitud, luego le restamos por **-1**, de esta forma traeremos el último elemento, luego podemos hacer **-2, -3**, etc. y ya sabremos que serán elementos cercanos al final de nuestra cadena.

TIPO DE RETORNO: CHAR[]

MÉTODO → toCharArray()

Descripción

- Este método nos servirá para splitear/separar carácter por carácter una cadena, por lo que por medio de una cadena, al estar utilizando este método, crearemos un arreglo de caracteres, por lo que podremos iterar por medio de un bucle ya que ahora será un arreglo.

FÓRMULA DEL MÉTODO: ***miCadena*****.toCharArray()**

USO del MÉTODO

EJEMPLO VÁLIDO → "hola".toCharArray() = {'h','o','l','a'} ← Devuelve

Argumentos: 1-requerido (tipo cadena) → Espera recibir como argumento la posición/índice del elemento/carácter.

CONSEJO: Si queremos ver su resultado podemos intentar iterar por medio de un bucle o accediendo a algún carácter como lo haríamos en un arreglo.

NOTA: Si no usamos bucles o no accedemos al carácter por medio de su posición/índice y mostramos la información como tal (como se puede ver en el uso del método), nos mostrará el conjunto de caracteres como si fuese una cadena, pero no lo es, sigue siendo un arreglo de caracteres.

Más información:

- Repositorio del JDK (**importante**): https://github.com/openjdk/jdk
- https://openjdk.org/projects/jdk/
- https://docs.oracle.com/javase%2F7%2Fdocs%2Fapi%2F%2F/java/lang/String.html
- https://www.w3schools.com/java/java_ref_string.asp