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: INTEGER

MÉTODO → codePointAt()

Descripción

- Este método se encargará de devolvernos en código **UNICODE** del carácter especificado.
  Ver sitio: https://symbl.cc/es/unicode/table/ ← Busca el carácter interesado en obtener el Unicode, luego le podes dar click (*en mi caso solo dejaré el cursor sobre el carácter y me mostrará una vista previa*) y verás el detalle, te tiene que interesar el **Código HTML** y el **Código Alt**, ya que en estos dos verás el resultado que se te ha reflejado por consola.
  Otro sitio para ver el Unicode: http://www.webusable.com/AltKeyCodes.htm

FÓRMULA DEL MÉTODO: ***miCadena*****.codePointAt(posIndice)**

USO del MÉTODO

EJEMPLO VÁLIDO → "Hola".codePointAt(2) = 108 ← Devuelve → 'l' ← ALT + 108

Argumentos: **1-requerido (tipo entero)** → Espera recibir como argumento una entero, este argumento indica la posición del carácter que queremos tomar para reflejar su código **UNICODE**.

MÉTODO → codePointBefore()

Descripción

- Este método se encargará de devolvernos en código **UNICODE** antes de llegar al carácter especificado. Ver sitio: https://symbl.cc/es/unicode/table/ ← Busca el carácter interesado en obtener el Unicode, luego le podes dar click (*en mi caso solo dejaré el cursor sobre el carácter y me mostrará una vista previa*) y verás el detalle, te tiene que interesar el **Código HTML** y el **Código Alt**, ya que en estos dos verás el resultado que se te ha reflejado por consola.
  Otro sitio para ver el Unicode: http://www.webusable.com/AltKeyCodes.htm

FÓRMULA DEL MÉTODO: ***miCadena*.codePointBefore(posIndice)**
***miCadena*.codePointBefore(posIndice) → "Hola".codePointBefore(2) = 111 ← 'o' ← ALT + 111**

USO del MÉTODO
***EJEMPLO VÁLIDO* → "Hola". codePointBefore(2) = 111 ← Devuelve → 'o' ← ALT + 111**

Argumentos: **1-requerido (tipo entero)** → Espera recibir como argumento un entero, este argumento indica la posición del carácter que queremos tomar para reflejar su código **UNICODE**, pero hará **-1**, por lo que el carácter obtenido será antes de la posición del carácter especificado.

NOTA: En este método el primer elemento/carácter equivaldría a **1**, no a **0** como se lo vino haciendo hasta el momento, esto se debe a que luego del **0** comenzarán valores negativos, de ahí el (**posIndice - 1**) si nosotros ponemos (**0 - 1**) ya sería **-1** y estará fuera de rango, por lo que nos lanzará una excepción.

MÉTODO → hashCode()

Descripción

- Este método nos servirá para traernos el código Hash de una cadena completa. El código Hash para un **String** se calcula de la siguiente forma → **s[0]*31^(n-1) + s[1]*31^(n-2) + ... + s[n-1]** donde **s[i]** es el iésimo (**elemento en el puesto n**) carácter de la cadena, **n** es la longitud de la cadena y **^** indica exponenciación.

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

USO del MÉTODO
***EJEMPLO VÁLIDO* → "Java".hashCode() = 2301506 ← Devuelve**
***EJEMPLO VÁLIDO* → "Hola".hashCode() = 2255068 ← Devuelve**

MÉTODO → length()

Descripción

- Este método nos servirá para devolvernos la longitud/cantidad de caracteres que tiene una cadena. En caso de que la cadena se encuentre vacía nos devolverá **0**, se comienza a contar desde el **1**.

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

USO del MÉTODO
***EJEMPLO VÁLIDO* → "Holala".length() = 6 ← Devuelve**
***EJEMPLO VÁLIDO* → "".length() = 0 ← Devuelve**
***EJEMPLO VÁLIDO* → " ".length() = 6 ← Devuelve (*contiene 6 caracteres en blanco*)**

MÉTODO → codePointCount()

Descripción

- Este método se encargará de devolvernos la cantidad de valores **UNICODE** encontrados en una cadena. Este método consiste en contar el número de puntos **UNICODE** dentro del rango de la cadena especificado; su rango comienza a partir del "**índice inicial**, es decir el **0**" y se extiende hasta el "**índice final**", por lo que la operación interna que hace es la "**índice inicial - índice final**" llevándolo a un caso más practico podría ser → "**0 - 3= 3 | 1 - 3= 2 | 2 - 3= 1 | 3 - 3= 0**" ← Explicación. Tomará como primer argumento el "**índice inicial**", luego restará dicho índice con el "**índice final**", dando así como retorno la resta de ambos argumentos ingresados.
  El lenguaje Java utiliza valores **UNICODE** para representar lo que sería cadenas (*conjunto de caracteres*), donde por cada carácter que conforma nuestra cadena se lo conoce como **PUNTO DE CÓDIGO**; Java utiliza el codificador de caracteres **UTF-16**, donde consiste en el agrupamiento de **2 bytes** por carácter, es decir, **16 bits**, desafortunadamente hay demasiados caracteres, tal como se lo puede ver en esta imagen → https://en.wikipedia.org/wiki/UTF-16#/media/File:Unifont_Full_Map.png
  Para evitar el uso del codificador **UTF-16**, se utiliza **bytes** (*2 pares de 2*) para lo que sería **PUNTO DE CÓDIGO** con números muy grandes; puede resultar confuso su uso interno, ya que tratará un carácter de 4 bytes como si fuese de 2 caracteres. Más Info sobre UTF-16: https://en.wikipedia.org/wiki/UTF-16

FÓRMULA DEL MÉTODO: ***miCadena*.codePointCount(posInicial, posFinal)**

USO del MÉTODO

EJEMPLO VÁLIDO → "Holaoo".codePointCount(0,3) = 3 ← Devuelve
EJEMPLO VÁLIDO → "Holaoo".codePointCount(1,3) = 2 ← Devuelve
EJEMPLO VÁLIDO → "Holaoo".codePointCount(2,3) = 1 ← Devuelve
EJEMPLO VÁLIDO → "Holaoo".codePointCount(3,3) = 0 ← Devuelve

Argumentos: **1-requerido (tipo entero)** → Espera recibir como argumento un entero, este argumento indica la posición "**INICIAL**" donde comenzar la búsqueda. **2-requerido (tipo entero)** → Espera recibir como argumento un entero, este argumento indica la posición "**FINAL**", donde luego restará este con la posición "**INICIAL**" para hacer la resta.

NOTA: El valor del índice del primer argumento no debe superar al índice del segundo argumento, de ser así nos lanzará una excepción ya que saldrá del alcance de la longitud de nuestra cadena, si ambos argumentos cuentan con el mismo valor, nos retornará **0**, ya que la operación interna se encarga de hacer la operación de resta.

MÉTODO → compareTo()

Descripción

- Este método se encargará de comparar dos cadenas de forma ordenada y natural, es decir alfabéticamente. Dicha comparación es basada en el valor **UNICODE** de cada carácter de las cadenas. Este método devolverá **0** si la cadena es igual a la otra cadena, de lo contrario devolverá un número alejado del 0.
  - **Devolverá un valor menor que 0** siempre y cuando la cadena sea menor a la otra cadena, es decir, tenga menos caracteres.
    - **Cadena1 < Cadena2**
  - **Devolverá un valor mayor que 0** siempre y cuando la cadena sea mayor que la otra cadena, es decir, tenga más caracteres.
    - **Cadena1 > Cadena2**

FÓRMULA DEL MÉTODO: ***miCadena*.compareTo("otra cadena")**

USO del MÉTODO

EJEMPLO VÁLIDO → "Hola".compareTo("Hola") = 0 ← Devuelve
EJEMPLO VÁLIDO → "hola".compareTo("Hola") = 32 ← Devuelve
EJEMPLO VÁLIDO → "hola".compareTo("Cómo estás?") = 37 ← Devuelve

Argumentos: **1-requerido (tipo cadena)** → Espera recibir como argumento una cadena, esta cadena será la encargada de ser evaluada con la cadena base que fue la que accedió al método **compareTo()**.

NOTA: Si queremos comparar dos cadenas tener en cuenta los valores **UNICODE**, debemos utilizar el método **equals()**.

MÉTODO → compareToIgnoreCase()

Descripción

- Este método es igual al método **compareTo()**, con la salvedad de que este método se encargará de comparar dos cadenas de forma ordenada y natural, es decir alfabéticamente, pero, a diferencia del método **compareTo()** no tendrá en cuenta los caracteres que se encuentren en minúsculas y mayúsculas. Dicha comparación es basada en el valor **UNICODE** de cada carácter de las cadenas. Este método devolverá **0** si la cadena es igual a la otra cadena, de lo contrario devolverá un número alejado del 0.
    - **Devolverá un valor menor que 0** siempre y cuando la cadena sea menor a la otra cadena, es decir, tenga menos caracteres.
        - **Cadena1 < Cadena2**
    - **Devolverá un valor mayor que 0** siempre y cuando la cadena sea mayor que la otra cadena, es decir, tenga más caracteres.
        - **Cadena1 > Cadena2**

  Si queremos que las cadenas a evaluar no tengan en cuenta los caracteres mayúsculos y minúsculos, debemos utilizar este método **compareToIgnoreCase()**.

FÓRMULA DEL MÉTODO: ***miCadena*****.** compareToIgnoreCase(**"otra cadena"**)

USO del MÉTODO

EJEMPLO VÁLIDO → "Hola".compareToIgnoreCase("hola") = 0 ← Devuelve
EJEMPLO VÁLIDO → "HOLa". compareToIgnoreCase("Hola") = 0 ← Devuelve

Argumentos: **1-requerido (tipo cadena)** → Espera recibir como argumento una cadena, esta cadena será la encargada de ser evaluada con la cadena base que fue la que accedió al método **compareTo()**.

MÉTODO → indexOf()

Descripción

- Este método nos servirá para traernos la posición de la primera coincidencia/aparición de los caracteres/criterio de búsqueda especificada en una cadena.
  Si el criterio de búsqueda fue exitoso nos debería mostrar la posición/índice, de lo contrario nos devolverá **-1**.

FÓRMULA DEL MÉTODO: ***miCadena*****.**indexOf(**"cadena" | 'carácter'**)

USO del MÉTODO

EJEMPLO VÁLIDO → "Hola".indexOf("la") = 2 ← Devuelve (*comienza a partir del índice 2*)
EJEMPLO VÁLIDO → "Hola".indexOf("las") = -1 ← Devuelve (*no se ha encontrado coincidencia*)
EJEMPLO VÁLIDO → "Hola".indexOf("ola") = 1 ← Devuelve (*comienza a partir del índice 1*)
EJEMPLO VÁLIDO → "Hola".indexOf("ola",2) = -1 ← Devuelve (*no se ha encontrado coincidencia*)
EJEMPLO VÁLIDO → "Hola".indexOf("ola",1) = 1 ← Devuelve (*comienza a partir del índice 1*)
EJEMPLO VÁLIDO → "Hola".indexOf('l') = 2 ← Devuelve (*comienza a partir del índice 2*)
EJEMPLO VÁLIDO → "Hola".indexOf('l',2) = 2 ← Devuelve (*comienza a partir del índice 2*)
EJEMPLO VÁLIDO → "Hola".indexOf('l',1) = -1 ← Devuelve (*no se ha encontrado coincidencia*)

Argumentos: **1-requerido (tipo cadena | tipo carácter)** → Espera recibir como argumento una cadena, esta cadena será la encargada de buscar coincidencia en la cadena base. **2-requerido (tipo entero)** → Espera recibir como argumento una entero que será el encargado de verificar si el criterio de búsqueda comienza a partir de esa posición.

MÉTODO/ESTÁTICO → offsetByCodePoints()

Descripción

Este método se encargará de devolvernos el índice que se encuentra dentro de la secuencia de caracteres, las secuencias de caracteres provienen de la clase **CharSequence**.
Este método trabaja con la clase **Character** que hace que nos devuelva el índice especificado de la submatriz/subcadena de caracteres dada que ha sido compensada por **codePoints**, todos los sustitutos no emparejados que se encuentran en el rango del texto proporcionado por el índice y la compensación de **codePoints** se cuentan como un **codePoint** cada uno.
Devolverá el número de la operación **index + longitud**.

FÓRMULA DEL MÉTODO:

- ***miCadena*.offsetByCodePoints(índice inicial, longitud dada)**
- ***CharSequence*.offsetByCodePoints(secuencia, índice inicial, longitud dada)**

USO del MÉTODO

EJEMPLO VÁLIDO → "Hola".offsetByCodePoints(1,2) = 3 ← Devuelve (*índice inicial + longitud*)
EJEMPLO VÁLIDO → "Hola".offsetByCodePoints(0,2) = 2 ← Devuelve (*índice inicial + longitud*)
EJEMPLO VÁLIDO → "Hola".offsetByCodePoints(0,1) = 1 ← Devuelve (*índice inicial + longitud*)
EJEMPLO VÁLIDO → "Hola Java".offsetByCodePoints(1,6) = 6 ← Devuelve (*índice inicial + longitud*)
EJEMPLO VÁLIDO → "Hola Java".offsetByCodePoints(0,8) = 8 ← Devuelve (*índice inicial + longitud*)
EJEMPLO VÁLIDO → "Hola".offsetByCodePoints(0,5) = IndexOutOfBoundsException

OTRO EJEMPLO USANDO LA CLASE → CharSequence & Character

```
CharSequence secuencia = "Hola Java";
SALIDA → Character.offsetByCodePoints(secuencia, 0, secuencia.length()) ← 9 → (0 + 9)
SALIDA → Character.offsetByCodePoints(secuencia, 1, 8) ← 9 → (1 + 8)
SALIDA → Character.offsetByCodePoints(secuencia, 1, 9) ← IndexOutOfBoundsException
```

Argumentos: **1-requerido - En el caso de usar la clase Character (tipo secuencia de caracteres)** → Espera recibir como argumento una secuencia de caracteres. **2 si usamos Character, 1 de lo contrario-requerido (tipo entero)** → Espera recibir como argumento una entero que será el encargado decirle al método a partir desde que carácter "**INICIAL**" quiere comenzar. **3 si usamos Character, 2 de lo contrario-requerido (tipo entero)** → Espera recibir como argumento una entero que será el encargado de medir la longitud de nuestra secuencia.

NOTA: La suma realizada entre el índice y la longitud de nuestra secuencia de caracteres debe:

- ***miCadena*.offsetByCodePoints(índice inicial, longitud dada)** ← Si usa esta opción no debe ser igual o superior a la longitud de la secuencia dada.
- ***CharSequence*.offsetByCodePoints(secuencia, índice inicial, longitud dada)** ← si usa esta opción debe ser igual o inferior a la longitud de la secuencia dada.

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

Te espero del otro lado ´\(O.<)/`

YouTube: https://www.youtube.com/@bailadev93
Twitter: https://twitter.com/bailadev93
Facebook: https://www.facebook.com/bailadev1993

Donativos: https://cafecito.app/bailadev93

https://github.com/bailadev93