PROGRAMACIÓN

ANEXO II: Clase String

1. ¿Qué es la clase String?

Una cadena (String) es una secuencia de caracteres incluidos, letras del alfabeto, caracteres especiales y espacios en blanco.

Java no tiene un tipo primitivo de variable como int, double, etc., para la manipulación de cadenas, para ello utiliza la clase String.

A todos los efectos una variable de tipo String es un objeto de la clase Java.Lang.String.

2. ¿Cómo usarla? Métodos

Ya sabemos como usar los Strings, pero vamos a ver diferentes curiosidades sobre esta clase.

Lo primero que debemos saber es que los Strings están indexados, es decir, cada carácter del String está asociado a una posición; por ejemplo:

0	1	2	3
H	O	L	A

Por lo que podemos acceder a estos caracteres de forma individual.

Además, la clase String tiene muchísimos métodos ya definidos que podemos usar directamente.

Entre ellos destacamos:

LENGTH

(Devuelve un entero)

Devuelve la longitud de la cadena. Por ejemplo del String “hola”, nos devolverá un 4.

```
String str = “hola”;
int longitud_string = str.length();
```

CHAR AT

(Recibe un entero)

(Devuelve un caracter)

Hemos visto que los Strings están indexados, por tanto, si le pasamos una posición, nos devolverá el carácter en esta.

```
String str = “hola”;

char caracter_string = str.charAt(3);

// carácter_string vale “a”
```

TO LOWER CASE

(Devuelve un String)

Devuelve la cadena que use el método, pero entera en minúsculas. Muy útil cuando queremos comparar Strings sin tener en cuenta mayúsculas y minúsculas.

```
String str = “HOLa”;

String str_minus = str.toLowerCase();

// str_minus vale “hola”
```

TO UPPER CASE

(Devuelve un String)

Devuelve la cadena que use el método, pero entera en mayúsculas. Muy útil cuando queremos comparar Strings sin tener en cuenta mayúsculas y minúsculas.

```
String str = “Hola”;

String str_mayus = str.toUpperCase();

// str_mayus vale “HOLA”
```

INDEX OF

(Recibe un String)

(Devuelve un entero)

Recibe un carácter o una palabra y busca en el String original si se encuentra este carácter o esta palabra dentro de él. En caso de encontrarlo devuelve la PRIMERA posición donde lo encuentre.

Por ejemplo, si a “buenos días” le pasamos la letra “s”, nos devolverá que se encuentra en la posición 5.

Si le pasamos “buenos” nos dirá que se encuentra en la posición 0.

Si le pasamos “wsw” nos devolverá un -1, indicando que no se encuentra.

```
String str = “hola”;
int posicion_string = str.indexOf(“a”);
// posicion_string vale 3
```

LAST INDEX OF

(Recibe un String)

(Devuelve un entero)

Recibe un carácter o una palabra y busca en el String original si se encuentra este carácter o esta palabra dentro de él. En caso de encontrarlo devuelve la ÚLTIMA posición donde lo encuentre.

Por ejemplo, si a “buenos días” le pasamos la letra “s”, nos devolverá que se encuentra en la posición 10.

SUBSTRING

(Recibe 2 enteros)

(Devuelve un String)

Recibe dos posiciones y devuelve lo que se encuentre en la cadena original, pero solo dentro de esas posiciones.

Por ejemplo si tenemos nuestra cadena “buenos días”, y le pasamos los índices 3 y 6 nos devolverá “nos ”

```
String str = “hola que tal”;

String sub_string = str.substring(0, 3);

// sub_string vale “hola”
```

REPLACE

(Recibe 2 caracteres)

(Devuelve un String)

Recibe dos caracteres y devuelve la cadena original sustituyendo todos los caracteres iguales al primero por el segundo.

Ideal para quitar espacios. (Se puede pasar el carácter vacio)

Por ejemplo si tenemos nuestra cadena “buenos días”, y le pasamos los índices “s” y “S” nos devolverá “buenoS díaS"

```
String str = “hola que tal”;

String replace_string = str.replace(“a”, “x”);

// replace_string vale “holx que txl”
```

COMPARE TO

(Recibe 1 string)

(Devuelve un entero)

Compara el String original con el recibido.

Devuelve un 0 si son iguales, un número negativo si el original va alfabéticamente primero, y un número positivo si es al revés.´

Ideal para cuando necesitamos ordenar matrices.

COMPARE TO IGNORE CASE

(Recibe 1 string)

(Devuelve un entero)

Compara el String original con el recibido.

Devuelve un 0 si son iguales, un número negativo si el original va alfabéticamente primero, y un número positivo si es al revés.

EN ESTE CASO IGNORA LA DIFERENCIA ENTRE MAYÚSCULAS Y MINÚSCULAS.

Ideal para cuando necesitamos ordenar matrices.

Hay un montón de métodos más que nos pueden ser útiles.

¡Investígalos en la documentación de Java!

3. Un poco más

Por último, una manera más limpia de concatenar cadenas con variables, en vez de estar usando "……." + variable + "……." + variable2 + "….." …

Es usar cadenas formateadas, ¿Cómo se usa esto?

Con el método FORMAT.

Usando este método, que recibe primero la cadena original, y después las variables que vamos a introducir dentro de ellas.

Parece un poco confuso, pero mejor veámoslo con un ejemplo.

String str = String.format("El alumno %s %s ha sacado una calificación de %i", nombre, apellido, nota);

Automáticamente introducirá en la variable str el valor de nombre donde encuentre el primer %s, de apellido donde encuentre el segundo %s, y la calificación donde encuentre %i

¿Por qué %s? Porque vamos a introducir un String.

Si vamos a introducir un entero (int) escribiremos %i, %d para doubles, …