

Hoja resumen del tema 1: La clase String.

Nota: esta hoja es tan solo una hoja-resumen con parte de los contenidos de la asignatura. No viene explicada toda la teoría, tan solo es un esquema para que puedas apoyarte a la hora de hacer los ejercicios.

Las Strings son una ristra de caracteres. Esto significa que una String está formada por la concatenación de varios caracteres.

Como vimos en *introducción a la informática*: “un carácter (char) es cualquier símbolo. Java utiliza el estándar de codificación de caracteres denominado UNICODE, el cual tiene 2 bytes (2^{16}) caracteres; es decir, 65.536 caracteres”

Por lo que si tenemos diferentes caracteres y los unimos, obtenemos una String.

Podemos ver una String como palabras o frases, pero no es del todo correcto dado que la String es todo el conjunto que hay entre dos "", por lo que si tenemos, por ejemplo, “Hola chicos, chicas o @%10”, la String es todo. No “Hola”, “chicos” “y” “chicas”, los espacios, los números, las comas: los caracteres no alfanuméricos también cuentan.

Para inicializar una String podemos hacerlo de la misma forma que hemos hecho con cualquier otra variable.

```
String s = "hola";
```

Debes saber es que la String son **INALTERABLES**:

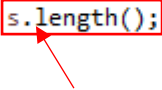
- No se pueden modificar.
- No se pueden quitar ni agregar caracteres a una String creada.
- No se puede subdividir, ni partir...etc.

Por tanto, para trabajar con Strings hace falta que utilicemos una serie de métodos que ya están creados y guardados en la biblioteca de java.

Para poder utilizarlo debemos **invocarlos** y para ello tenemos que utilizar el punto ‘.’

Ejemplo:

```
String s = "Hola chicos y chicas"; //Declaro una String
int tamaño = s.length();          //Invoco al método length()
```



Es importante recalcar que el método length() es un método que pertenece a las Strings. Es necesario una String sobre la que se invocará el método.

Existen una gran diversidad de método de la clase String y en este tema vamos a ver los más útiles.

Para ello, junto con la explicación de cada método acompañaré un con un trozo de código donde podrás ver cómo se declara, como se indican los parámetros y qué devuelve.

Además antes de cada método van estas dos instrucciones que permitirán ver el resultado por pantalla. (la String y el print irán cambiando en los diferentes apartados)

```
String ejemplo = "";
System.out.print("String: \"" + ejemplo + "\"\n Al hacer x da: ");
```

Métodos:

1) *length()*. El método *length()* devuelve la longitud de una String, es decir, el número de caracteres por la cual está compuesta.

(código)

```
// 01234
String ejemplo = "Hola";
int longitud = ejemplo.length(); // 4
System.out.println(longitud);
```

(consola)

```
String: "Hola"
Al hacer ejemplo.length() da: 4
```

¡OJO! No lo confundas con el length de los arrays: *length()* y *length* NO SON LO MISMO. Uno (el que tiene paréntesis) es un método mientras que el otro es un atributo.

2) *charAt()*. El método *charAt(int)* devuelve el carácter (char) que se encuentre en la posición que pasamos por parámetros. Si pasamos una posición que se salga de rango de la String dará **error**.

(código)

```
// 01234
String ejemplo = "Hola";
char caracter = ejemplo.charAt(3); // 'a'
System.out.print(caracter);
```

(consola)

```
String: "Hola"
Al hacer ejemplo.charAt(3) da: 'a'
```

3) *indexOf()*. El método *indexOf()* puede usarse de dos maneras: tenemos *indexOf(char)* e *indexOf(int, char)*. Lo que devuelve el método *indexOf()* es el índice (int) donde se encuentra la primera vez que encontramos el carácter (char) que pasamos por parámetro. Si añadimos un int antes, buscará **a partir** de dicho int.

(código)

```
// 0123456789
String ejemplo = "Hola amig@";
int indice = ejemplo.indexOf('a'); // 3
System.out.print(indice);
```

(consola)

```
String: "Hola amig@"
Al hacer ejemplo.indexOf('a') da: 3
```

```
// 0123456789
String ejemplo = "Hola amig@";
int indice = ejemplo.indexOf('a', 4); // 5
System.out.print(indice);
```

```
String: "Hola amig@"
Al hacer ejemplo.indexOf('a', 4) da: 5
```

4) *lastIndexOf()*. Funciona exactamente igual que *indexOf()* salvo que devuelve el índice donde se encuentra la última vez que encontramos el carácter pasado por parámetro. También se puede añadir un int que buscará hasta dicho int.

(código)

```
// 0123456789
String ejemplo = "Hola amiga";
int indice = ejemplo.lastIndexOf('a'); // 9
System.out.print(indice);
```

(consola)

```
String: "Hola amiga"
Al hacer ejemplo.lastIndexOf('a') da: 9
```

```
// 0123456789
String ejemplo = "Hola amiga";
int indice = ejemplo.lastIndexOf('a', 4); // 3
System.out.print(indice);
```

```
String: "Hola amiga"
al hacer ejemplo.lastIndexOf('a', 4) da: 3
```

5) `toUpperCase()` y `toLowerCase()`. Convierte toda la String a mayúscula o a minúscula respectivamente.

(código)

```
String ejemplo = "Hola ¿TODO BIEN?";
String mayuscula = ejemplo.toUpperCase();
System.out.print(mayuscula);
```

```
String ejemplo = "Hola ¿TODO BIEN?";
String mayuscula = ejemplo.toLowerCase();
System.out.print(mayuscula);
```

(consola)

```
String: "Hola ¿TODO BIEN?"
Al hacer ejemplo.toUpperCase() da: "HOLA ¿TODO BIEN?"
```

```
String: "Hola ¿TODO BIEN?"
Al hacer ejemplo.toLowerCase() da: "hola ¿todo bien?"
```

6) `trim()`. Elimina todos los caracteres espaciadores (espacios, tabulaciones, salto de líneas...) de los extremos de una String. Recalco: solo quita de los **extremos**.

(código)

```
String ejemplo = "  Hola   gente  ";
String resultado = ejemplo.trim();
System.out.print(resultado);
```

(consola)

```
String: "  Hola   gente  "
al hacer ejemplo.trim() da: "Hola   gente"
```

7) `concat()`. Concatena (une) una string con otra. También se puede usar el '+'. El orden importa: no es lo mismo "Hola" + "chic@s" -> "Holachic@s" que "chic@s" + "Hola" -> "chic@sHola".
¡También hay que tener en cuenta los espacios!

(código)

```
String ejemplo = "Hola ";
String frase = "chic@s";
String resultado = ejemplo.concat(frase);
System.out.print(resultado);

System.out.print("\n\"Hola\" + \" \" + \"chic@s\\\"");
```

(consola)

```
String: "Hola" y String: "chic@s"
Al hacer ejemplo.concat(frase) da: "Holachic@s"

"Hola chic@s"
```

8) `startsWith()` y `endsWith()`. Devuelve "true" o "false" si la String empieza o acaba (respectivamente) por la String que se pasa por parámetro.

(código)

```
String ejemplo = "Hola gente";
String frase = "Ho";
boolean resultado = ejemplo.startsWith(frase);
System.out.print(resultado);
```

(consola)

```
String 1: "Hola gente" y String 2: "Ho"
Al hacer ejemplo.startsWith(frase) da: true
```

```
String ejemplo = "Hola gente";
String frase = "Ho";
boolean resultado = ejemplo.endsWith(frase);
System.out.print(resultado);
```

```
String 1: "Hola gente" y String 2: "Ho"
Al hacer ejemplo.endsWith(frase) da: false
```

9) *equals()*. Devuelve “true” o “false” si la String es exactamente igual que la que se pasa por parámetro. Una String se considera igual a otra si tiene los mismo caracteres en el mismo orden. (Implicítamente también son del mismo tamaño)

(código)

```
String ejemplo = "Hola gente";
String frase = "Ho";
boolean resultado = ejemplo.equals(frase);
System.out.print(resultado);
```

```
String ejemplo = "Hola gente";
String frase = "hola gente";
boolean resultado = ejemplo.equals(frase);
System.out.print(resultado);
```

(consola)

```
String 1: "Hola gente" y String 2: "Hola gente"
Al hacer ejemplo.equals(frase) da: true
```

```
String 1: "Hola gente" y String 2: "hola gente"
Al hacer ejemplo.equals(frase) da: false
```

10) *contains()*. Devuelve “true” o “false” si la String contiene a la String que se la pasa por parámetro.

(código)

```
String ejemplo = "Hola gente";
String frase = "ola";
boolean resultado = ejemplo.contains(frase);
System.out.print(resultado);
```

```
String ejemplo = "Hola gente";
String frase = "alo";
boolean resultado = ejemplo.contains(frase);
System.out.print(resultado);
```

(consola)

```
String: "Hola gente" y String 2: "alo"
Al hacer ejemplo.contains(frase) da: true
```

```
String: "Hola gente" y String 2: "alo"
Al hacer ejemplo.contains(frase) da: false
```

11) *substring()*. Devuelve como resultado un trozo de la String a partir de dos posiciones que se pasan por parámetros. Si le pasamos solo uno, devolverá a partir de ese punto hasta el final sin incluirlo.

(código)

```
//          0123456789
String ejemplo = "Hola Mundo";
String resultado = ejemplo.substring(4);
System.out.print(resultado);
```

```
//          0123456789
String ejemplo = "Hola Mundo";
String resultado = ejemplo.substring(1,4);
System.out.print(resultado);
```

(consola)

```
String: "Hola Mundo"
al hacer ejemplo.substring(4) da: " Mundo"
```

```
String: "Hola Mundo"
al hacer ejemplo.substring(1,4) da: "ola"
```

13) *split()*. Devuelve como resultado una array de String a partir de una expresión que pasamos por parámetros.

```
String ejemplo = "Hola Mundo"; //String
//Separamos por espacios
String[] resultado = ejemplo.split(" ");
//usamos el for para imprimir los valores
//del array
for(int i = 0; i < resultado.length; i++){
    System.out.println("\n" + resultado[i] + "\n");
}
```

```
String: "Hola Mundo"
al hacer ejemplo.split(" ") da:
"Hola"
"Mundo"
```

12) *compareTo()*. Devuelve la diferencia numérica entre las dos Strings.

La diferencia numérica de dos String se calcula recorriendo las dos Strings y, si se encuentra un carácter diferente, se devuelve la diferencia entre ellos. En caso de no encontrar ningún carácter distinto, se devolverá la diferencia de los tamaños. Si esta operación da 0, significan que las dos String tienen los mismos caracteres en la misma posición y que son del mismo tamaño, lo que implica que son exactamente iguales.

(código)

```
//Dos Strings con los mismos caracteres
//pero distinto tamaño
String ejemplo = "Hola";    //pequeña
String frase   = "Hola gente"; //grande
int resultado = ejemplo.compareTo(frase);
System.out.println(resultado);
```

```
//Dos Strings con los mismos caracteres
//pero distinto tamaño
String ejemplo = "Hola gente"; //grande
String frase   = "Hola";       //pequeña
int resultado = ejemplo.compareTo(frase);
System.out.println(resultado);
```

```
//Dos Strings distintas
String ejemplo = "Hola chic0"; //0
String frase   = "Hola chicA"; //A
int resultado = ejemplo.compareTo(frase);
System.out.println(resultado); //0 - A
```

```
//Dos Strings distintas
String ejemplo = "Hola chicA"; //A
String frase   = "Hola chic0"; //0
int resultado = ejemplo.compareTo(frase);
System.out.println(resultado); //A - 0
```

(consola)

String 1: "Hola" y String 2: "Hola gente"
al hacer ejemplo.compareTo(frase) da: -6

String 1: "Hola gente" y String 2: "Hola"
al hacer ejemplo.compareTo(frase) da: 6

String 1: "Hola chic0" y String 2: "Hola chicA"
al hacer ejemplo.compareTo(frase) da: 14

String 1: "Hola chicA" y String 2: "Hola chic0"
al hacer ejemplo.compareTo(frase) da: -14