SANTA ANA
Y SAN RAFAEL
Madrid

COLEGIOS
MARIANISTAS

CLASE STRING

UD 5 Clases en Java

PROGRAMACIÓN – 1ero DAM

CLASE STRING

- ✓ La clase **String** se utiliza para almacenar cadenas de caracteres.
- ✓ Para usarla debemos crear un **objeto** de la clase **String**, y podemos hacer uso del mismo y de todos sus métodos (ya existentes dentro de la clase).
- ✓ Como cualquier **objeto**, hay que **declararlo**:

  String pepe;

- ✓ Pero en este caso **no es necesario utilizar el constructor**, (ni usar *new*) sino que podemos asignarle contenido directamente:

  String pepe = "Hola, que tal?";

- ✓ **Importante**: Una variable de tipo **char** sólo almacena un caácter, y se indica con comillas simples. Un **String** es una cadena de caracteres y se indica con doble comilla.

CLASE STRING

- ✓ La clase **String** tiene varios métodos muy útiles que nos ayudarán a trabajar con ella, como saber su longitud, trocear la cadena, etc.
- ✓ Recuerda que para invocar un **método** debemos escribir el nombre del objeto de tipo **String**, un punto y el nombre del **método**, más sus **parámetros**.
- ✓ Es importante que si necesitas almacenar el valor devuelto, uses una variable para ello. Por ejemplo:

  String cadena=”americano”;
- ✓ Y en el siguiente método de la clase **String** podemos usar la variable de tipo **String**:

```
boolean empiezaPor = cadena.startWith (“am”);
//este método devuelve true o false si la cadena empieza por am
```

CLASE STRING

- ✓ La clase **String** tiene varios métodos muy útiles que nos ayudarán a trabajar con ella, como saber su longitud, trocear la cadena, etc.
- ✓ Recuerda que para invocar un **método** debemos escribir el nombre del objeto de tipo **String**, un punto y el nombre del **método**, más sus **parámetros**.
- ✓ Es importante que si necesitas almacenar el valor devuelto, uses una variable para ello. Por ejemplo:

  String cadena=”americano”;
- ✓ Y en el siguiente método de la clase **String** podemos usar la variable de tipo **String**:

```
boolean empiezaPor = cadena.startWith (“am”);
//este método devuelve true o false si la cadena empieza por am
```

MÉTODOS DE LA CLASE STRING (1)

MÉTODO	DESCRIPCIÓN	PARÁMETRO	TIPO DE DATO DEVUELTO
charAt	Devuelve el carácter indicado por parámetro	Un parámetro int	char
compareTo	Sirve para comparar cadenas, devuelve un número según el resultado. Recuerda que no sigue el alfabeto español, lo compara según la tabla ASCII.	Un parámetro String, la cadena a comparar.	int - Si devuelve un número mayor que 0: la primera cadena es mayor que la segunda. - Si devuelve un 0: las cadenas son iguales. - Si devuelve un número menor que 0: la primera cadena es menor que la segunda
compareToIgnoreCase	Es igual que el anterior, pero ignorando mayúsculas o minúsculas.	Un parámetro String, la cadena a comparar	int - Si devuelve un número mayor que 0: la primera cadena es mayor que la segunda. - Si devuelve un 0: las cadenas son iguales. - Si devuelve un número menor que 0: la primera cadena es menor que la segunda
concat	Concatena dos cadenas, es como el operador +.	Un parámetro String, la cadena a concatenar	Un nuevo String con las cadenas concatenadas.

MÉTODOS DE LA CLASE STRING (2)

MÉTODO	DESCRIPCIÓN	PARÁMETRO	TIPO DE DATO DEVUELTO
copyValueOf	Crea un nuevo String a partir de un array de char. Este método debe invocarse de manera estática, es decir, String.copyValueOf(array_char)	Un array de char	String
endsWith	Indica si la cadena acaba con el String pasado por parámetro.	String	boolean
equals	Indica si una cadena es igual que otra.	String	boolean
equalsIgnoreCase	Es igual que el anterior, pero ignorando mayúsculas o minúsculas.	String	boolean
getBytes	Devuelve un array de bytes con el código ASCII de los caracteres que forman el String.	Ningún parámetro	Un array de bytes
indexOf	Devuelve la posición en la cadena pasada por parámetro desde el principio. -1 si no existe.	String o char	int
indexOf	Igual que el anterior, pero ademas le indicamos la posición desde donde empezamos a buscar.	String o char, el segundo parámetro es un int	int
lastIndexOf	Devuelve la posición en la cadena pasada por parámetro desde el final. -1 si no existe.	String o char	int
lastIndexOf	Igual que el anterior, pero ademas le indicamos la posición desde donde empezamos a buscar.	String o char, el segundo parámetro es un int	int
lenght	Devuelve la longitud de la cadena.	Ningún parámetro	int
matches	Indica si la cadena cumple con la expresión pasada como parámetro. Pincha aquí para tener mas detalles.	String	boolean
replace	Devuelve un String cambiando los caracteres que nosotros le indiquemos.	Dos parámetros char, el primero es el carácter que existe en el String y el segundo por el que queremos cambiar.	String
replaceFirst	Devuelve un String intercambiando solo la primera coincidencia.	Dos parametros String, el primero son los caracteres que existe en el String y el segundo por el que queremos cambiar.	String
replaceAll	Devuelve un String intercambiando todas las coincidencias que se encuentren.	Dos parametros String, el primero son los caracteres que existe en el String y el segundo por el que queremos cambiar.	String
startsWith	Indica si la cadena empieza por una cadena pasada por parámetro.	String	boolean

MÉTODOS DE LA CLASE STRING (3)

MÉTODO	DESCRIPCIÓN	PARÁMETRO	TIPO DE DATO DEVUELTO
substring	Trocea un String desde una posición a otra.	Dos parámetros int, indica desde donde empieza hasta donde acaba, este ultimo no se incluye.	String
toCharArray	Devuelve en un array de char, todos los caracteres de una String.	Ningún parámetro	Un array de char
toLowerCase	Convierte el String a minúsculas.	Ningún parámetro	String
toUpperCase	Convierte el String a mayúsculas.	Ningún parámetro	String
trim	Elimina los espacios del String.	Ningún parámetro	String
valueOf	Transforma una variable primitiva en un String. Para invocarse debe usarse con String. Por ejemplo, **String.valueOf(variable)**	Un parámetro, que puede ser un: • boolean • char • double • int • float • long • Array de char • Referencia a un objeto	String

UD5: CLASE STRING

EJEMPLO

```
1 public class PruebaApp {
2
3     public static void main(String[] args) {
4
5         String cadena="El que se fue a Sevilla perdio su silla y el que se fue al Torreon, su sillon";
6
7                 System.out.println(cadena.charAt(0)); // Nos devolvera E
8
9                 System.out.println(cadena.charAt(11)); //Nos devolvera u
10
11                System.out.println(cadena.endsWith("n")); //Nos devuelve true
12
13                System.out.println(cadena.startsWith("e")); //Nos devuelve false, Java distingue entre mayusculas y minusculas
14
15                System.out.println(cadena.equals("El que se fue a Sevilla perdio su silla y el que se fue al Torreon, su sillon")); //Nos devuelve true
16
17                byte[] array_bytes=cadena.getBytes(); //Creamos un array de bytes e insertamos la devolucion del metodo
18
19                System.out.println("Codigo ASCII de cada caracter");
20
21                for (int i=0;i<array_bytes.length;i++){
22                    System.out.print(array_bytes[i]+" "); //Muestra los codigos
23                }
24                System.out.println("");
25                System.out.println(cadena.indexOf("fue")); //localiza la posicion donde se encuentra "fue"
26
27                System.out.println(cadena.length()); // Nos devuelve la longitud: 77
28
29                System.out.println(cadena.replace('a', 'e')); // Cambia todas las a por e
30
31                System.out.println(cadena.toLowerCase()); //Transforma el String a minusculas
32
33                System.out.println(cadena.toUpperCase()); //Transforma el String a mayusculas
34
35    }
36
37}
```

LEER UN STRING DE PANTALLA

Limpiar buffer scanner después de leer un String

Al utilizar una variable del objeto **Scanner** para leer un **String** de pantalla, tenemos que limpiar posteriormente el buffer de Scanner si queremos seguir leyendo de pantalla otro tipo de dato diferente.

El motivo es el siguiente: al coger un **String** por pantalla, toma la cadena de caracteres seguida de un retorno de carro. Ese **retorno de carro se queda residual como basura en el buffer**, por lo que si volvemos a leer de pantalla otro dato, por ejemplo un **int**, no nos cogerá el dato.
La solución, después de leer un **String** por pantalla con **Scanner**, hacer una llamada al método .**nextLine()** de la variable **Scanner**, y ya podemos seguir usándola con normalidad:

Ejemplo:
Si la variable **Scanner** que estamos usando para leer de pantalla se llama teclado, la línea de limpieza del buffer será la siguiente:

```
teclado.nextLine();       //Limpiamos buffer de entrada
```