Tabla de contenido

Métodos de la clase String

La clase String tiene muchos métodos, los cuales, al igual que los demás métodos en Java, son llamados a través de la notación punto (esto es, colocando un punto (.) después del nombre de la variable tipo String y escribiendo el nombre del método a continuación). Algunos de dichos métodos son:

equals() {argumentos: String; retorna: boolean}

Este método tiene una cadena como argumento y compara el valor de la variable que lo invoca con el valor del argumento y devuelve *true* si son EXACTAMENTE iguales o *false* en caso contrario (este método diferencia entre mayúsculas y minúsculas).

Ejemplo:

```
String s1=”Hola”;

String s2=”hola”;

boolean iguales=s1.equals(s2);          //devuelve false porque para Java las mayúsculas y minúsculas son diferentes.

boolean compara=s1.equals(“Hola”);      //devuelve true porque los valores de las cadenas son iguales.
```

equalsIgnoreCase() {argumentos: String; retorna: boolean}

Este método tiene una cadena como argumento y compara el valor de la variable que lo invoca con el valor del argumento y devuelve *true* si son iguales o *false* en caso contrario (este método ignora las diferencias entre mayúsculas y minúsculas).

Ejemplo:

```
String s1=”Hola”;

String s2=”hola”;

boolean iguales=s1.equals(s2);          //devuelve true porque ignora las diferencias entre mayúsculas y minúsculas.
```

split() {argumentos: String; retorna: String[]}

Este método tiene una cadena como argumento y divide el valor de la variable que lo invoca en subcadenas (que no contienen el argumento) cada vez que encuentra una coincidencia con el argumento o al final de la variable. Dichas subcadenas son guardadas en un arreglo unidimensional de cadenas.

Ejemplo:

```
String srt=”split sirve para dividir cadenas”;

String partes[]=srt.split(“ “);          //devuelve un arreglo con los valores [“split”, “sirve”, “para”, “dividir”, “cadenas”].
Tener en cuenta que los espacios desaparecen del arreglo.
```

toCharArray() {argumentos: no; retorna: char[]}

Este método no tiene argumentos y devuelve un arreglo unidimensional de caracteres cuyos valores son TODOS Y CADA UNO de los caracteres que hacen parte de la variable que lo invoca.

Ejemplo:

```
String srt=”Hola”;

char letras[]=srt.toCharArray();          //devuelve un arreglo con los valores [‘H’, ‘o’, ‘l’, ‘a’]
```

concat() {argumentos: String; retorna: String}

Este método tiene una cadena como argumento y devuelve el valor de la variable que lo invoca seguido INMEDIATAMENTE por el valor del argumento.

Ejemplo:

```
String str=”Buenas”;

String resultado=str.concat(“noches”);        //devuelve Buenasnoches porque une las dos cadenas sin importar sus valores

String saludo=str.concat(“ noches”);          //devuelve “Buenas noches” porque el argumento está precedido por un espacio
```

contains() {argumentos: String; retorna: boolean}

Este método tiene una cadena como argumento y devuelve *true* si el valor del argumento se encuentra en el valor de la variable o *false* en caso contrario.

Ejemplo:

```
String str=”Hola”;

boolean encontrado=str.contains(“ola”);       //devuelve true porque ola está en hola

boolean perdido=str.contains(“alo”);          //devuelve false porque alo no está en hola
```

isEmpty() {argumentos: no; retorna: boolean}

Este método no tiene argumentos y devuelve *true* si la variable que lo invoca no tiene ningún caracter en su interior o *false* en caso contrario.

Ejemplo:

```
String str="";

String cad="a";

boolean vacio=str.isEmpty();          //devuelve true porque la cadena no tiene caracteres

boolean lleno=cad.isEmpty();          //devuelve false porque la cadena tiene POR LO MENOS un caracter
```

trim() {argumentos: no; retorna: String}

Este método no tiene argumentos y devuelve el valor de la variable que lo invoca SIN espacios al inicio y final de la cadena (en caso de existir (hay que tener en cuenta que este método no tiene efecto en los espacios intermedios).

Ejemplo:

```
String str="  Al inicio hay dos espacios y al final uno ";

String resultado=str.trim();          //devuelve la cadena "Al inicio hay dos espacios y al final uno" (tenga en cuenta
```

que los espacios intermedios se conservaron).

charAt() {argumentos: int; retorna: char}

Este método tiene un entero como argumento y devuelve, de la variable que lo invoca, el caracter que está ocupando la posición dada por el argumento (hay que recordar que SIEMPRE, en Java, los caracteres de una cadena empiezan a contarse desde la posición cero).

Ejemplo:

```
String str="texto";

char letra=str.charAt(1);                    //devuelve el carácter 'e' ya que éste está en la posición 1 (uno) de la variable.
```

replace() {argumentos: String; retorna: String}

Este método tiene dos cadenas como argumento y devuelve la variable que lo invocacon el espacio del primer argumento reemplazado por el segundo.

Ejemplo:

String mensajePirata= " ¡Cofres Piratas Enterrados! ";

		¡	C	o	f	r	e	s		P	i	r	a	t	a	s		E	n	t	e	r	r	a	d	o	s	!	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

String s2=mensajePirata.replace("Cofres", "Baules"); //Se reemplaza en la cadena mensajePirata, la palabra "Cofres" por "Baules"

		¡	B	a	u	l	e	s		P	i	r	a	t	a	s		E	n	t	e	r	r	a	d	o	s	!	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

length() {argumentos: no; retorna: int}

Este método no tiene argumentos y devuelve la cantidad de caracteres que componen la variable que lo invoca (hay que recordar que SIEMPRE, en Java, los caracteres de una cadena empiezan a contarse desde la posición cero).

Ejemplo:

String mensajePirata= “ ¡Cofres Piratas Enterrados! ”;

		¡	C	o	f	r	e	s		P	i	r	a	t	a	s		E	n	t	e	r	r	a	d	o	s	!	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

int i=mensajePirata.length(); //30 (porque va de 0 a 29)

startsWith() {argumentos: String; retorna: boolean}

Este método tiene una cadena como argumento y devuelve *true* en caso de que la variable que lo invoca EMPIECE con el argumento o *false* en caso contrario.

Ejemplo:

String s1=“Este es un ejemplo”;

boolean b=s1.startsWith(“Est”); //devuelve *true* porque la frase “Este es un ejemplo” comienza con “Est”.

endsWith() {argumentos: String; retorna: boolean }

Este método tiene una cadena como argumento y devuelve *true* en caso de que la variable que lo invoca TERMINE con el argumento o *false* en caso contrario.

Ejemplo:

String s1="Este es un ejemplo";

boolean b=s1.endsWith("plo"); //devuelve *true* porque la frase "Este es un ejemplo" termina con "plo".

substring(){argumentos: int; retorna: String}

Este método tiene uno o dos enteros como argumento y devuelve, de la variable que lo invoca, la subcadena que está ocupando la posición dada desde el primer argumento hasta el segundo SIN incluirlo (hay que recordar que SIEMPRE, en Java, los caracteres de una cadena empiezan a contarse desde la posición cero).

Ejemplo:

String mensajePirata= " ¡Cofres Piratas Enterrados! ";

		¡	C	o	f	r	e	s		P	i	r	a	t	a	s		E	n	t	e	r	r	a	d	o	s	!	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

String s1=mensajePirata.substring(10); //devuelve "Piratas Enterrados! "

String s2=mensajePirata.substring(10,11); //devuelve "P"

String s3=mensajePirata.substring(10,10); //devuelve en blanco

toLowerCase() {argumentos: no; retorna: String}

Este método no tiene argumentos y devuelve el contenido de la variable que lo invoca en MINÚSCULAS.

Ejemplo:

String s1="FRASE GRANDE";

String s2=s1.toLowerCase(); //Devuelve la cadena "frase grande" (note que está en minúsculas).

toUpperCase() {argumentos: no; retorna: String}

Este método no tiene argumentos y devuelve el contenido de la variable que lo invoca en MAYÚSCULAS.

Ejemplo:

String s1="frase pequeña";

String s2=s1.toUpperCase(); //Devuelve la cadena "FRASE PEQUEÑA" (note que está en mayúsculas).

indexOf() {argumentos: char; retorna: int}

Este método tiene un caracter como argumento y devuelve, de la variable que lo invoca, la posición que está ocupando el argumento DE IZQUIERDA A DERECHA (hay que recordar que SIEMPRE, en Java, los caracteres de una cadena empiezan a contarse desde la posición cero).

Ejemplo:

String mensajePirata= " ¡Cofres Piratas Enterrados! ";

		¡	C	o	f	r	e	s		P	i	r	a	t	a	s		E	n	t	e	r	r	a	d	o	s	!	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

int i1=mensajePirata.indexOf('r'); //devuelve 6 (1° posición de r)

int i2= mensajePirata.indexOf('z'); //devuelve -1 porque no existe el carácter z en la frase dada.

lastIndexOf() {argumentos: char; retorna: int}

Este método tiene un caracter como argumento y devuelve, de la variable que lo invoca, la posición que está ocupando el argumento DE DERECHA A IZQUIERDA (hay que recordar que SIEMPRE, en Java, los caracteres de una cadena empiezan a contarse desde la posición cero).

Ejemplo:

String mensajePirata= " ¡Cofres Piratas Enterrados! ";

		¡	C	o	f	r	e	s		P	i	r	a	t	a	s		E	n	t	e	r	r	a	d	o	s	!	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

int i1=mensajePirata.lastIndexOf('r'); //devuelve 23 (última posición de r)

int i2= mensajePirata.lastIndexOf('z'); //devuelve -1 porque no existe el carácter z en la frase dada.