Cadenas

Clase String

Un String en Java representa una **cadena de caracteres no modificable**.
Todos los literales de la forma "*cualquier texto*", es decir, literales entre comillas dobles, que aparecen en un programa java se implementan como objetos de la clase String.

CREAR UN STRING

Se puede **crear un String** de varias formas, entre ellas:

- Utilizando una **cadena de caracteres** entre comillas:

```
String s1 = "abcdef";
```

- Utilizando **operador de concatenación** + con dos o más objetos String:

```
String s2 = s1 + "ghij"; //s2 contiene "abcdefghij“
String s3 = s1 + s2 + "klm"; //s3 contiene " abcdefabcdefghijklm"
```

Constructores

CONSTRUCTOR	DESCRIPCIÓN

String()	Constructor por defecto. El nuevo String toma el valor ""
	String s = new String(); //crea el string s vacío.
	Equivale a: String s = "";

String(String s)	Crea un nuevo String, copiando el que recibe como parámetro. String s = "hola"; String s1 = new String(s); //crea el String s1 y le copia el contenido de s
String(char[] v)	Crea un String y le asigna como valor los caracteres contenidos en el array recibido como parámetro. char [] a = {'a', 'b', 'c', 'd', 'e'}; String s = new String(a); //crea String s con valor "abcde"
String(char[] v , int pos, int n)	Crea un String y le asigna como valor los n caracteres contenidos en el array recibido como parámetro, a partir de la posición pos. char [] a = {'a', 'b', 'c', 'd', 'e'}; String s = new String(a, 1, 3); //crea String s con valor "bcd";

Métodos

MÉTODO	DESCRIPCIÓN
length()	Devuelve la longitud de la cadena
indexOf('caracter')	Devuelve la posición de la primera aparición de carácter dentro del String. Devuelve -1 si no lo encuentra.
lastIndexOf('caracter')	Devuelve la posición de la última aparición de carácter dentro del String. Devuelve -1 si no lo encuentra.
charAt(n)	Devuelve el carácter que está en la posición n
substring(n1,n2)	Devuelve la subcadena desde la posición n1 hasta n2 - 1
toUpperCase()	Devuelve la cadena convertida a mayúsculas
toLowerCase()	Devuelve la cadena convertida a minúsculas
equals(otroString)	Compara dos cadenas y devuelve true si son iguales
equalsIgnoreCase(otroString)	Igual que equals pero sin considerar mayúsculas y minúsculas
compareTo(OtroString)	Devuelve 0 si las dos cadenas son iguales. <0 si la primera es alfabéticamente menor que la segunda ó >0 si la primera es alfabéticamente mayor que la segunda.
compareToIgnoreCase(OtroString)	Igual que compareTo pero sin considerar mayúsculas y minúsculas.
valueOf(N)	Convierte el valor N a String. N puede ser de cualquier tipo.

Strings NO SON modificables

Debemos recordar que:

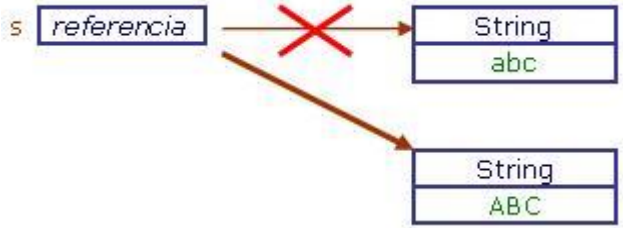
Los objetos String no son modificables.

Por lo tanto, los métodos que actúan sobre un String con la intención de modificarlo lo que hacen es crear un **nuevo** String a partir del original y devolverlo modificado.

Por ejemplo: Una operación como convertir a mayúsculas o minúsculas un String no lo modificará sino que creará y devolverá un nuevo String con el resultado de la operación.

```
String s = "abc";
```


```
s = s.toUpperCase(); //convertir a mayúsculas el contenido del String s
```



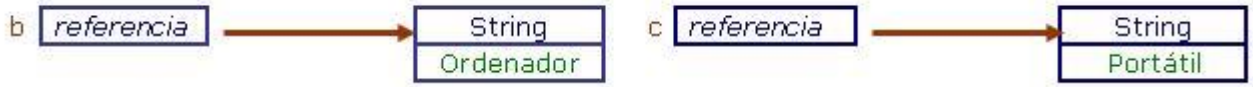
El **recolector de basura** es el encargado de eliminar de forma automática los objetos a los que ya no hace referencia ninguna variable.

Concatenación

EL OPERADOR DE CONCATENACIÓN +

La clase proporciona el operador + (concatenación) para unir dos o más String.
El resultado de aplicar este operador es un nuevo String concatenación de los otros.
Por ejemplo, si tenemos dos String b y c:

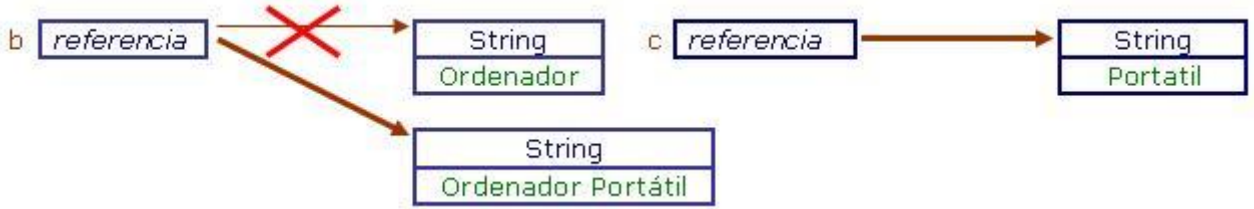
```
String b = "Ordenador";
String c = " Portátil";
```



La operación:

```
b = b + c;
```

crea un nuevo String (b + c) y le asigna su dirección a b:



END

jgardur081@g.educaand.es