# Clase Character y[[1]](#footnote-0) variable char

**boolean isDigit (char) ->**

Nos indica si un carácter introducido es un dígito numérico.

**boolean isLetter (char) ->**

Nos indica si el carácter introducido es una letra.

**boolean isLetterOrDigit (char) ->**

Nos indica si el carácter introducido es un dígito alfanumérico.

**boolean isLowerCase (char) ->**

Nos indica si el carácter introducido es una letra y si ésta es minúscula.

**boolean isUpperCase (char) ->**

Nos indica si el carácter introducido es una letra y si ésta es mayúscula.

**boolean isSpaceChar (char) ->**

Nos indica si el carácter introducido es un espacio.

**boolean isWhiteSpace (char) ->**

Nos indica si el carácter introducido es un espacio en blanco, es decir, que no es ningún tipo de caracteres especiales del lenguaje, como sería un salto de línea “\n”.

## Conversión entre caracteres

**toLowerCase ->**

Si has introducido un carácter alfabético en mayúscula lo devuelve en minúscula.

**toUpperCase ->**

Si has introducido un carácter afabético en minúscula te lo devolverá en mayúscula.

## Igualdad

**boolean equals (String) ->**

Indica si dos cadenas de caracteres son iguales.

**boolean equalsIgnoreCase (String) ->**

Indica si dos cadenas son iguales independientemente de si tienen mayúsculas o minúsculas.

**boolean regionMatches (int inicio, String otraCadena, int inicioOtra, int longitud) ->**

Compara dos fragmentos de cadenas: el primero corresponde a la cadena invocante y comienzca en el carácter con índice “inicioOtra”. Ambos fragmentos tendrán la longitud indicada.

**boolean regionMatches (boolean ignora, int inicio, String otraCadena, int inicioOtra, int longitud) ->**

Hace lo mismo que el de arriba pero ignora mayúsculas y minúsculas si se indica “true” en el boolean.

## Obtención de un carácter

**char charAt (int posición) ->**

Devuelve el carácter que se encuentra en la posición introducida. Ejemplo:

String cadenita = “Pepa”;

System.out.println(cadenita.charAt(2)); //Se imprimirá p porque se

//encuentra en la posición 2.

# Clase String y variable String

**Static String valueOf (tipo valor) ->**

Construye y devuelve el valor de una cadena. Ejemplo:

String cad;

cad = String.valueOf(1234); //cad = “1234”

cad = String.valueOf(‘C’); //cad = “C”

cad = String.valueOf(false); //cad = “false”

## Comparación alfabética

**int compareTo (String) ->**

Compara alfabéticamente la cadena invocante con la que se pasa como parámetro, es decir, la que es mayor alfabéticamente (las letras van antes) es mayor.

* Si nos indica un 0 es que las cadenas son exactamente iguales.
* Si nos indica un número POSITIVO es que la cadena pasada como parámetro es mayor que la invocante, es decir, va después en orden alfabético.
* Si nos indica un número NEGATIVO es que la cadena pasada como parámetro es mayor alfabéticamente, es decir, va antes.

**int compareToIgnoreCase (String) ->**

Realiza la misma función que la anterior pero no distingue entre mayúsculas y minúsculas.

## Concatenación

El operador “+” concatena dos cadenas diferentes:

String cad1 = “Miguel”;

String cad2 = “pela patatas”;

String cad3 = cad1 + cad2;

System.out.println(cad3); //cad3= “Miguepela patatas”.

## Obtención de una subcadena

**String subString (int inicio) ->**

Devuelve una subcadena formada desde la posición (inicio) indicada. Ejemplo:

String cad1 = “Hola me llamo Olalla”;

String cad2 = cad1.subString(7); //cad2 = “ llamo Olalla”.

**String subString (int inicio, int fin) ->**

Devuelve una subcadena formada desde el parámetro inicio, hasta el parámetro fin, igual que la anterior pero podemos decir hasta donde llegará esa subcadena.

**String strip( ) ->**

Devuelve una cadena (copia) sin espacios delante ni atrás.

**String stripLeading ( ) ->**

Al igual que la 1ª pero ésta solo elimina los espacios del principio de la

cadena.

**String stripTrailing ( ) ->**

Elimina los caracteres blancos del final.

## Cadena vacía

**boolean isEmpty ( ) ->**

Indica si la cadena invocante está vacía. (true si lo está, false si contiene algún carácter).

## Contiene

boolean contains (CharSequence subcadena) ->

Devuelve true si en la cadena invocante (xxxxx.CharSequence(subcadena), se en cuentra la subcadena en cualquier posición.

## Prefijos

boolean startsWith (String prefijo) ->

Indica si la cadena invocante comienza con el prefijo indicado.

boolean startsWith (String prefijo, int inicio) ->

Al igual que la 1ª pero comienza la comparación desde la posición “inicio”.

## Sufijos

boolean endsWith (String sufijo) ->

Devuelve si la cadena acaba con “sufijo”.

## Conversión

String toLowerCase( ) ->

Devuelve la cadena invocada en minúsculas.

String toUpperCase ( ) ->

Devuelve la cadena invocada en mayúsculas.

String replace (char original, char otro) ->

Devuelve una copia de la cadena donde se han substituido los caracteres originales por la variable “otro”.

String replace (CharSequence original, CharSequence otra) ->

Cambia todas las ocurrencias de de la cadena “original” por la cadena “otra”.

## Separación en partes

String [ ] split (String separador) ->

Devuelve las subcadenas creadas a partir de dividir la cadena invocante con “separador”.

## Cadenas y tablas de caracteres

char [ ] toCharArray ( ) ->

Devuelve una tabla de tipo char con el contenido de la cadena invocante. Se introduce un carácter por lugar del array.

static String valueOf (char [ ] tabla) ->

Devuelve un String con el contenido de la tabla de caracteres.

static String valueOf (char [ ] t, int inicio, int cuantos) ->

Funciona como la anterior pero devuelve la cadena formada por un subconjunto de caracteres consecutivos a la tabla t. Ejemplo:

String cad;

char c [] = {‘a’,’b’,’c’,’d’,’e’,’f’,’g’};

cad = String.valueOf(c,2,4); //Esto vale “cdef”.

# Variable tipo int

int indexOf(int c) ->

Busca la primera ocurrencia del carácter introducido.

int indexOf (String cadena) ->

Busca la 1ª ocurrencia de una cadena.

el delimitador de la cadena es cada caracter

int indexOf (int c, int inicio) ->

Busca la 1ª ocurrencia del carácter c, pero en lugar de empezar desde la pos 0, comienza desde la posición indicada en inicio.

int index Of (String cadena, int inicio) ->

Busca la 1ª ocurrencia de cadena a partir de la posición inicio.

Si añadimos “last” invocamos a un método que hace los mismo que estos 4 anteriores pero realiza la búsqueda desde el final de la cadena.

1. [↑](#footnote-ref-0)