

Parámetro

Un argumento o parámetro es el medio a partir del cual podemos expandir el ámbito de variables locales de funciones, hacia otras funciones y además quienes nos permiten establecer comunicaciones entre funciones

Tipo de dato devuelto

El tipo de dato devuelto por la sentencia `return` debe coincidir con el tipo de dato declarado en la cabecera del método; El *tipo de dato* indica el tipo de valor que devuelve la llamada al método y los *parámetros* (entre paréntesis) introducen información para la ejecución del método.

Equals: Un objeto es una cosa distinta a un tipo primitivo, aunque “porten” la misma información.

```
String sCadena1 = new String("Avila");
String sCadena2 = new String("Salamanca");
String sCadena3 = new String("Avila");

if (sCadena1.equals(sCadena2))
    System.out.println(sCadena1 + " y " + sCadena2 + " son IGUALES");
else
    System.out.println(sCadena1 + " y " + sCadena2 + " son DIFERENTES");


if (sCadena1.equals(sCadena3))
    System.out.println(sCadena1 + " y " + sCadena3 + " son IGUALES");
else
    System.out.println(sCadena1 + " y " + sCadena3 + " son DIFERENTES");
```

```
3 public class Persona {
4     private String nombre;
5
6     public String getNombre() {
7         return nombre;
8     }
9
10
11     public void setNombre(String nombre) {
12         this.nombre = nombre;
13     }
14
15     public Persona(String nombre) {
16         super();
17         this.nombre = nombre;
18     }
19
20     @Override
21     public int hashCode() {
22         return nombre.hashCode();
23     }
24
25     @Override
26     public boolean equals(Object obj) {
27         Persona p= (Persona)obj;
28
29         return p.getNombre().equals(this.getNombre());
30     }
31 }
32
33 }
```


`public boolean equals(Object anObject)`

Determina lógicamente si el argumento es igual a la cadena.

Si cadena="Java";

`boolean resultado1 = cadena.equals("Java");`
`boolean resultado2 = cadena.equals("java");` 

resultado1 es igual a **true**

resultado2 es igual a **false** 

equalsIgnoreCase: Compara la cadena de texto contra un objeto ignorando mayúsculas y minúsculas. Devolverá true si las cadenas comparadas son iguales. En caso contrario devolverá false.

```
public class Word {  
    public static void main(String[] args) {  
        String str1 = "Millie Bobby Brown";  
        ArrayList<String> list = new ArrayList<>();  
        list.add("millie bobby brown");  
        list.add("finn");  
        list.add("gaten");  
        list.add("caleb");  
        list.add("noah");  
        for (String str : list) {  
            if (str.equalsIgnoreCase(str1)) {  
                System.out.println("Millie Bobby Brown is present");  
            }  
        }  
    }  
}
```

```
String sCadena1 = new String("Avila");  
String sCadena2 = new String("Salamanca");  
String sCadena3 = new String("AVILA");  
  
if (sCadena1.equalsIgnoreCase(sCadena2))  
    System.out.println(sCadena1 + " y " + sCadena2 + " son IGUALES");  
else  
    System.out.println(sCadena1 + " y " + sCadena2 + " son DIFERENTES");  
  
if (sCadena1.equalsIgnoreCase(sCadena3))  
    System.out.println(sCadena1 + " y " + sCadena3 + " son IGUALES");  
else  
    System.out.println(sCadena1 + " y " + sCadena3 + " son DIFERENTES");
```

```
public class Javaapp {  
  
    public static void main(String[] args){  
  
        String str1 = new String("hajsof");  
        String str2 = new String("HAJSOF");  
  
        if(str1.equals(str2))  
        {  
            System.out.println("equals() : Both values are same");  
        }  
        if(str1.equalsIgnoreCase(str2))  
        {  
            System.out.println("equalsIgnoreCase() : Both values are same");  
        }  
    }  
}
```

| |
|--|
| compareTo: se usa para realizar una ordenación natural en una cadena. Clasificación natural significa el orden de clasificación que se aplica al objeto, por ejemplo, orden léxico para Cadena, orden numérico para ordenar enteros, etc. |
| <pre>public class Sample_String { public static void main(String[] args) { String str_Sample = "a"; System.out.println("Compare To 'a' b is : " + str_Sample.compareTo("b")); str_Sample = "b"; System.out.println("Compare To 'b' a is : " + str_Sample.compareTo("a")); str_Sample = "b"; System.out.println("Compare To 'b' b is : " + str_Sample.compareTo("b")); } }</pre> |
| <pre>public class Sample_String { public static void main(String[] args) { String str_Sample = "a"; System.out.println("Compare To 'a' b is : " + str_Sample.compareTo("b")); str_Sample = "b"; System.out.println("Compare To 'b' a is : " + str_Sample.compareTo("a")); str_Sample = "b"; System.out.println("Compare To 'b' b is : " + str_Sample.compareTo("b")); } }</pre> |
| <pre>public class Compare { public static void main(String[] args) { String s1 = "Guru1"; String s2 = "Guru2"; System.out.println("String 1: " + s1); System.out.println("String 2: " + s2); // Compare the two strings. int S = s1.compareTo(s2); // Show the results of the comparison. if (S < 0) { System.out.println("\"" + s1 + "\"" + " is lexicographically higher than " + "\"" + s2 + "\""); } else if (S == 0) { System.out.println("\"" + s1 + "\"" + " is lexicographically equal to " + "\"" + s2 + "\""); } else if (S > 0) { System.out.println("\"" + s1 + "\"" + " is lexicographically less than " + "\"" + s2 + "\""); } } }</pre> |

charAt: Devuelve el carácter en el índice definido. En este método, el valor del índice debe estar entre 0 y la longitud de la cadena menos 1

```
1 var cualquierCadena="Brave new world";
2
3 console.log("El carácter en el índice 0 es '" + cualquierCadena.charAt(0) + "'")
4 console.log("El carácter en el índice 1 es '" + cualquierCadena.charAt(1) + "'")
5 console.log("El carácter en el índice 2 es '" + cualquierCadena.charAt(2) + "'")
6 console.log("El carácter en el índice 3 es '" + cualquierCadena.charAt(3) + "'")
7 console.log("El carácter en el índice 4 es '" + cualquierCadena.charAt(4) + "'")
8 console.log("El carácter en el índice 999 es '" + cualquierCadena.charAt(999) + "'")
```

```
public class CharAtGuru99 {
    public static void main(String args[]) {
        String s1 = "This is String CharAt Method";
        //returns the char value at the 0 index
        System.out.println("Character at 0 position is: " + s1.charAt(0));
        //returns the char value at the 5th index
        System.out.println("Character at 5th position is: " + s1.charAt(5));
        //returns the char value at the 22nd index
        System.out.println("Character at 22nd position is: " + s1.charAt(22));
        //returns the char value at the 23th index
        char result = s1.charAt(-1);
        System.out.println("Character at 23th position is: " + result);
    }
}
```

```
1 package com.beginnersbook;
2 public class JavaExample {
3     public static void main(String[] args) {
4         String str = "BeginnersBook";
5
6         //initialized the counter to 0
7         int counter = 0;
8
9         for (int i=0; i<=str.length()-1; i++) {
10             if(str.charAt(i) == 'B') {
11                 //increasing the counter value at each occurrence of 'B'
12                 counter++;
13             }
14         }
15         System.out.println("Char 'B' occurred "+counter+" times in the string");
16     }
17 }
```

| |
|---|
| Concat: se usa para unir dos o más arrays. Este método no cambia los arrays existentes, sino que devuelve un nuevo array. |
| <pre>const array1 = ['a', 'b', 'c']; const array2 = ['d', 'e', 'f']; const array3 = array1.concat(array2); console.log(array3);</pre> |
| <pre>const letters = ['a', 'b', 'c']; const numbers = [1, 2, 3]; letters.concat(numbers);</pre> |
| <pre>const num1 = [1, 2, 3]; const num2 = [4, 5, 6]; const num3 = [7, 8, 9]; const numbers = num1.concat(num2, num3); console.log(numbers);</pre> |

| |
|--|
| Length: un objeto que es una instancia de tipo Array establece o devuelve la cantidad de elementos en esa matriz. |
| <pre>var namelistA = new Array(4294967296); //2 a la 32a potencia = 4294967296 var namelistC = new Array(-100) //signo negativo console.log(namelistA.length); //RangeError: longitud de la matriz inválida console.log(namelistC.length); //RangeError: longitud de la matriz inválida var namelistB = []; namelistB.length = Math.pow(2,32)-1; //establecer una longitud de la matriz menor que 2 a la 32ª potencia console.log(namelistB.length);</pre> |
| <pre>var arr = [1, 2, 3]; printEntries(arr); arr.length = 5; // establecer la longitud de la matriz en 5 mientras que actualmente es 3. printEntries(arr); function printEntries(arr) { var length = arr.length; for (var i = 0; i < length; i++) { console.log(arr[i]); } console.log('=== printed ==='); }</pre> |
| <pre>var numbers = [1, 2, 3, 4, 5]; var length = numbers.length; for (var i = 0; i < length; i++) { numbers[i] *= 2; }</pre> |

toCharArray: devuelve una cantidad Arrayde caracteres después de convertir una cadena en una secuencia de caracteres.

```

/**
 * @author Crunchify.com
 */

public class CrunchifyStringToCharArray {

    public static void main(String[] args) {
        String testString = "This Is Test";
        char[] stringToCharArray = testString.toCharArray();

        for (char output : stringToCharArray) {
            System.out.println(output);
        }
    }
}

```

```

class Chararr {

    public static void main(String args[]) {

        // initializing the string
        String str = "Appdividend";

        // converting the string into character array
        char[] ch = str.toCharArray();

        // printing array values
        for (int i = 0; i < ch.length; i++) {

            System.out.print(ch[i] + " ");

        }

        System.out.println("\n");

    }

}

```

```

using System;

class Sample {
    public static void Main() {
        string str = "012wxyz789";
        char[] arr;

        arr = str.ToCharArray(3, 4);
        Console.Write("The letters in '{0}' are: ", str);
        Console.Write(arr);
        Console.WriteLine("");
        Console.WriteLine("Each letter in '{0}' is:", str);
        foreach (char c in arr)
            Console.WriteLine(c);
    }
}

```

| |
|--|
| Replace: devuelve una nueva cadena con algunas o todas las coincidencias de un patrón, siendo cada una de estas coincidencias reemplazadas por remplazo |
| <pre>var str = "Mr Blue has a blue house and a blue car"; var res = str.replace(/blue house car/gi, function (x) { return x.toUpperCase(); });</pre> |
| <pre>var str = "Mr Blue has a blue house and a blue car"; var res = str.replace(/blue/gi, "red");</pre> |
| <pre>var str = "Mr Blue has a blue house and a blue car"; var res = str.replace(/blue/g, "red");</pre> |

| |
|--|
| toLowerCase: devuelve el valor de la cadena convertida a minúsculas. toLowerCase no afecta al valor de la cadena en sí misma. |
| <pre>var str = "Hello World!"; var res = str.toLowerCase();</pre> |
| <pre>public class Guru99 { public static void main(String args[]) { 3 String S1 = new String("MAYÚSCULA CONVERTIDA EN MINÚSCULA"); 4 // Convertir a LowerCase 5 System.out.println(S1.toLowerCase()); 6 } 7 }</pre> |
| <pre>public static void main(String args[]) { String S1 = new String("minúsculas convertidas en mayúsculas"); // Convertir a UpperCase System.out.println(S1.toUpperCase()); } }</pre> |

| |
|---|
| toUpperCase: método devuelve el valor convertido en mayúsculas de la cadena que realiza la llamada. |
| <pre>String sCadena = "Esto Es Una Cadena"; System.out.println(sCadena.toUpperCase()); //ESTO ES UNA CADENA</pre> |
| <pre>let strA = 'avengers will be a great movie'; console.log(strA.toUpperCase()); console.log('appdividend'.toUpperCase()); console.log('krunallathiya'.toUpperCase());</pre> |
| <pre>package prjstrings; public class StringManipulation { public static void main(String[] args) { String changeCase = "text to change"; System.out.println(changeCase); String result; result = changeCase.toUpperCase(); System.out.println(result); } }</pre> |

toString: se llama automáticamente cuando el objeto se representa como un valor de texto o cuando un objeto se referencia de tal manera que se espera una cadena.

```
var objeto = new Object();
objeto.toString(); // Devuelve [object Object]
```

```
cadena = new String("Hello world");
alert(cadena.toString()) // Displays "Hello world"
```

```
var num = 15;
var a = num.toString();
var b = num.toString(2);
var c = num.toString(8);
var d = num.toString(16);
```

Trim: elimina los espacios en blanco en ambos extremos del string. Los espacios en blanco en este contexto, son todos los caracteres sin contenido

```
var orig = ' foo ';
console.log(orig.trim()); // 'foo'
```

```
var orig = 'foo ';
console.log(orig.trim()); // 'foo'
```

```
if (!String.prototype.trim) {
  (function() {
    // Make sure we trim BOM and NBSP
    var rtrim = /^[\s\uFEFF\xA0]+|[\s\uFEFF\xA0]+$/g;
    String.prototype.trim = function() {
      return this.replace(rtrim, '');
    };
  })();
}
```

valueOf: método devuelve el valor primitivo de un objeto String.

```
cadena = new String("Hello world");
alert(cadena.valueOf()) // Displays "Hello world"
```

```
1 package com.beginnersbook;
2 public class JavaExample{
3     public static void main(String args[]){
4         int number = 23;
5         String str = String.valueOf(number);
6         System.out.println(str+99);
7     }
8 }
```

```
1 package com.beginnersbook;
2 public class JavaExample{
3     public static void main(String args[]){
4         char vowel[] = {'A', 'E', 'I', 'O', 'U'};
5         String str = String.valueOf(vowel);
6         System.out.println(str);
7     }
8 }
```