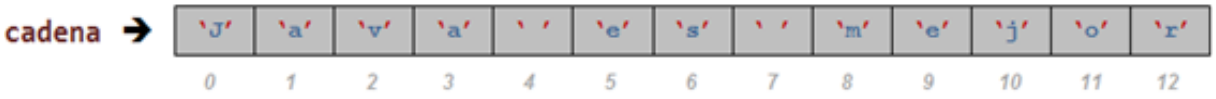
Métodos de la clase *String*

```
String cadena = "Java es mejor";
```



MÉTODO	DESCRIPCIÓN

public int length() {
}
	Devuelve la longitud de la cadena.
	`int longitud = cadena.length();`
	`longitud ← 13`

public char charAt(int) {
}
	Devuelve una copia del carácter que encuentre en la posición indicada por el parámetro.
	`char caracter = cadena.charAt(8);`
	`caracter ← 'm'`

public boolean equals(String) {
}
	Comprueba si dos cadenas son iguales. En este caso comprueba que el objeto dado como argumento sea de tipo *String* y contenga la misma cadena de caracteres que el objeto actual.
	`String s = "Java";`
	`boolean b = cadena.equals(s);`
	`b ← false`

public int compareTo(String) {
}
	Devuelve un entero menor que cero si la cadena es alfabéticamente menor que la dada como argumento, cero si las dos cadenas son léxicamente iguales y un entero mayor que cero si la cadena es mayor alfabéticamente.
	`String s1 = "Java es lo máximo",`
	`s2 = "Java es mejor",`
	`s3 = "Java gusta a todos";`
	`int i = cadena.compareTo(s1),`
	`j = cadena.compareTo(s2),`
	`k = cadena.compareTo(s3);`
	`i ← 1`
	`// cadena mayor que s1 alfabéticamente`
	`j ← 0`
	`// cadena contiene lo mismo que s2`
	`k ←-2`
	`// cadena menor que s3 alfabéticamente`

public boolean equalsIgnoreCase(String) {
}
	Realiza la misma tarea que *equals* pero sin tener en cuenta las mayúsculas o minúsculas.
	`String s = "Java Es MeJor";`
	`boolean b = cadena.equalsIgnoreCase(s);`
	`b ← true`

public boolean startsWith(String) {
}
	Comprueba si el comienzo de la cadena actual coincide con la cadena pasada como parámetro.
	`String s = "JavvaX";`
	`boolean b = cadena.startsWith(s);`
	`b ← false`

public boolean endsWith(String) {
}
	Comprueba si el final de la cadena actual coincide con la cadena pasada como parámetro.
	`String s = "mejor";`
	`boolean b = cadena.endsWith(s);`
	`b ← true`

public int indexOf(char) {
}
	Devuelve la posición que por primera vez aparece el carácter (expresado como entero) pasado como parámetro. En caso no exista devuelve -1.
	`int i = cadena.indexOf('e');`
	`i ← 5`

public int indexOf(char, int) {
}
	Devuelve la posición que por primera vez aparece el carácter (expresado como entero) a partir de la posición especificada como segundo parámetro.
	`int i = cadena.indexOf('e', 6);`
	`i ← 9`

public int indexOf(String) {
}
	Devuelve la posición que por primera vez aparece la cadena pasada como parámetro.
	`int i = cadena.indexOf("va");`
	`i ← 2`

public int indexOf(String, int) {
}
	Devuelve la posición que por primera vez aparece la cadena pasada como parámetro, pudiendo especificar en un segundo parámetro a partir de dónde buscar.
	`int i = cadena.indexOf("ej", 5);`
	`i ← 9`

public int lastIndexOf(char) {
}
public int lastIndexOf(char, int) {
}
public int lastIndexOf(String) {
}
public int lastIndexOf(String, int) {
}
	Devuelve la última vez que aparece el carácter (expresado como entero) o cadena pasada como parámetro, pudiendo especificar en un segundo parámetro, a partir de dónde buscar (búsqueda hacia atrás).
	`String s = "e";`
	`int i = cadena.lastIndexOf(s);`
	`i ← 9`

public String toLowerCase() {
}
	Retorna la cadena en minúsculas.
	`String s = "CiberJava – Lima - Perú";`
	`s = s.toLowerCase();`
	`s ← "ciberjava – lima – perú"`

public String toUpperCase() {
}
	Retorna la cadena en mayúsculas.
	`String s = "CiberJava – Lima - Perú";`
	`s = s.toUpperCase();`
	`s ← "CIBERJAVA – LIMA – PERÚ"`

public String trim() {
}
	Retorna la cadena sin espacios al principio y al final.
	`String s = "  CiberJava  Lima  ";`
	`s = s.trim();`
	`s ← "CiberJava  Lima"`

public String substring(int) {
}
public String substring(int, int) {
}
	Retorna una subcadena de la cadena actual, empezando por el primer índice indicado hasta antes del segundo índice (si se especifica) o hasta el final de la cadena.
	`String s1 = "viva el Perú",`
	`  s2 = s1.substring(5),`
	`  s3 = s1.substring(3, 9);`
	`s2 ← "el Perú"`
	`s3 ← "a el P"`

public String replace(char, char) {
}
	Retorna la cadena luego de reemplazar todos los caracteres iguales al primer parámetro y los sustituye por el carácter que pasamos en segundo lugar, teniendo en cuenta lo mismo una mayúscula que una minúscula.
	`String s = "biba el Perú";`
	`s = s.replace('b', 'v');`
	`s ← "viva el Perú"`

public String[] split(String) {
}
	Busca un tope en una cadena y distribuye una copia de las subcadenas en un arreglo lineal de cadenas.
	`String linea = "123;Ana;20;55.0";`
	`String[] s;`
	`s = linea.split(";");`
	`s[0] ← "123"`
	`s[1] ← "Ana"`
	`s[2] ← "20"`
	`s[3] ← "55.0"`

public char[] toCharArray() { }	Convierte la cadena en un vector de caracteres. `char[] arreglo = cadena.toCharArray();`

Métodos *static* de conversión	La clase *String* posee métodos para transformar valores de otros tipos de datos a cadena. Todos se llaman valueOf y son estáticos.

public static String valueOf(boolean) {
}
	`double r = 3.1416;`
	`String s = String.valueOf(r);`
	`s ← "3.1416"`

public static String valueOf(int) { }	
public static String valueOf(long) { }	
public static String(float) { }	
public static String(double) { }	
public static String(Object) { }	
public static String(char[]) { }	

public static String(char[], int, int) {
}
	Transforma una subcadena de un arreglo de caracteres, especificando una posición y la longitud.
	`char[] c = {'C','i','b','e','r','J','a','v','a'};`
	`String s = String.valueOf(c, 3, 5);`
	`s ← "erJav"`

"Sólo quienes dan un paso adelante viven la esperanza y la oportunidad."

MP