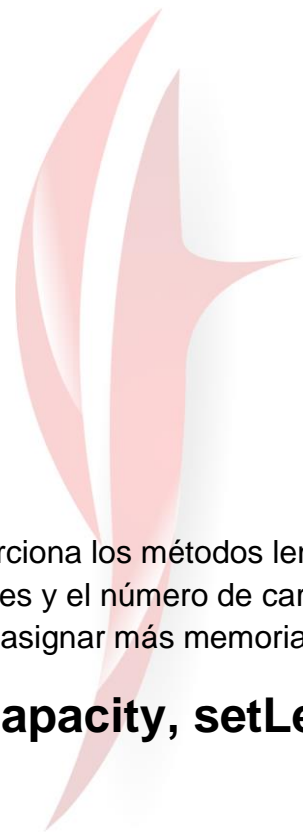




# MÉTODOS DE STRINGBUILDER



La clase `StringBuilder` proporciona los métodos `length` y `capacity` para devolver el número actual de caracteres y el número de caracteres que se pueden almacenar sin necesidad de asignar más memoria.

## Métodos `length`, `capacity`, `setLength` y `ensureCapacity`

El método `setLength` permite al programador incrementar o decrementar la longitud de un objeto `StringBuilder`.

En la aplicación siguiente viene el uso de estos métodos:



```
public class StringBuilderCapLen {

    public static void main(String args[]) {
        StringBuilder bufer = new StringBuilder("Hola, cómo
        estás?");

        System.out.printf("bufer = %s\nlongitud = %d\ncapac
        itad = %d\n\n",
            bufer.toString(), bufer.length(), bufer.capacity())
        ;

        bufer.ensureCapacity(75);

        System.out.printf("Nueva capacidad = %d\n\n", bufer
        .capacity());

        bufer.setLength(10);

        System.out.printf("Nueva longitud = %d\nbufer = %s\
        n",
            bufer.length(), bufer.toString());
    }
}
```

En este programa se utiliza también el método `ensureCapacity` para expandir la capacidad del objeto a un mínimo de 75 caracteres. En realidad, si la capacidad original es menor que el argumento, este método asegura una capacidad que sea el valor mayor entre el número especificado como argumento y el doble de la capacidad original más 2. Si la capacidad actual es más que la capacidad especificada, la capacidad no cambia.

Finalmente se utiliza el método `setLength` para establecer la longitud del objeto en 10. Si la longitud especificada es menor que el número de caracteres actual, el búfer se trunca a la longitud especificada. Si la longitud especificada es mayor que el número de caracteres actual, se anexan caracteres nulos (representación numérica 0) al objeto hasta que el número total de caracteres sea igual a la longitud especificada.



- El proceso de incrementar de forma dinámica la capacidad del objeto `StringBuilder` puede requerir una cantidad importante de tiempo. La ejecución de un gran número de estas operaciones puede degradar el rendimiento de una aplicación. si un objeto `StringBuilder` debe aumentar su tamaño de forma considerable, quizás varias veces, estableciendo su capacidad a un nivel alto desde un principio incrementaremos el rendimiento.

## Métodos `char`, `setCharAt`, `getChars` y `reverse`

La clase `StringBuilder` proporciona estos métodos para manipular los caracteres de el objeto. Podemos ver todos estos métodos en el programa siguiente. Los métodos `charAt` y `getChars` funcionan igual que en la clase `String`. El método `setCharAt` recibe un entero y un carácter como parámetros y asigna el carácter en la posición especificada. el método `reverse` invierte el contenido del objeto `StringBuider`.

```
public class StringBuilderChars {  
  
    public static void main(String args[]) {  
        StringBuilder bufer = new StringBuilder("hola a todos");  
        System.out.printf("bufer = %s\n", bufer.toString());  
        ;  
        System.out.printf("Carácter a 0: %s\nCarácter a 3: %s\n\n", bufer.charAt(0), bufer.charAt(3));  
        char arrayChars[] = new char[bufer.length()];  
        bufer.getChars(0, bufer.length(), arrayChars, 0);  
        System.out.print("Los caracteres son: ");  
        for (char character : arrayChars) {  
            System.out.print(character);  
        }  
        bufer.setCharAt(0, 'H');
```

```
        bufer.setCharAt(7, 'T');  
        System.out.printf("\n\nbufer = %s", bufer.toString()  
));  
        bufer.reverse();  
        System.out.printf("\n\nbufer = %s\n", bufer.toStrin  
g());  
    }  
}
```

## Métodos append

La clase `StringBuilder` proporciona métodos `append` sobrecargados para permitir que añadan valores de diversos tipos al final de un objeto. Se proporcionan versiones para cada uno de los tipos primitivos y para tablas de caracteres, objetos `String`, `Object`, `StringBuidler` y `CharSequence`. Cada uno de los métodos recibe un argumento, lo convierte en cadena y la anexa al objeto `StringBuilder`.

En realidad el compilador utiliza los objetos `StringBuilder` y los métodos `append` para implementar los operadores `+` y `+=` para concatenar objetos `String`. por ejemplo suponemos que realizamos las siguientes declaraciones:

```
String cadena1= "hola";  
String cadena2= "BC";  
int valor = 22;
```

La instrucción:

```
String s = cadena1 + cadena2 + valor;
```

Se realiza de la siguiente forma:

```
new StringBuilder().append("hola").append("BC").append(22).  
toString();
```



## Métodos de inserción y eliminación

La clase `StringBuilder` proporciona métodos `insert` sobrecargados para permitir que inserten valores de diversos tipos en cualquier posición del objeto. Cada uno de los métodos toma el segundo argumento, lo convierte en cadena y lo inserta justo antes del índice especificado por el primer argumento. El primer argumento debe ser mayor o igual a 0 y menor que la longitud del objeto.

La clase `StringBuilder` también proporciona métodos `delete` y `deleteCharAt` por eliminar caracteres en cualquier posición.

El método `delete` recibe dos argumentos: el índice inicial y el índice que se encuentra una posición más allá del último de los caracteres que se deben eliminar.

El método `deleteCharAt` recibe el índice del carácter a eliminar.

