¿POR QUÉ STRINGBUILDER?

- String es inmutable.
- Un objeto StringBuilder es un String que se puede modificar.
- Métodos más eficientes.

Mejora el uso de String porque no tiene sobrecarga. Al ser String inmutable, es decir, que cuando por ejemplo unes dos cadenas, esa unión la almacenas en una nueva variable String, eso con los métodos que hemos visto de String, es así, no puede hacerse de otra manera. Por eso StringBuilder es más eficiente.

TAMAÑO Y CAPACIDAD

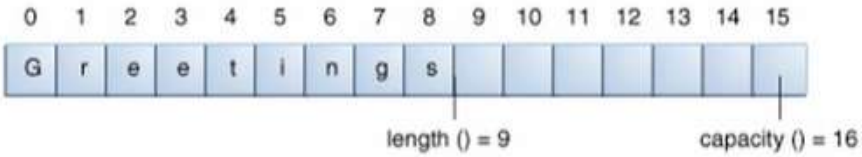
- Tamaño: **length()**. (Igual que String)
- Capacidad: número de caracteres que puede alojar. **capacity()**

Constructor	Descripción
StringBuilder()	Crea uno vacío, con capacidad = 16
StringBuilder(CharSequence cs)	Crea uno con los caracteres de *cs*, y 16 elementos vacíos adicionales.
StringBuilder(int initialCapacity)	Crea uno vacío, con la capacidad *initialCapacity*
StringBuilder(String s)	Crea uno con los caracteres de *s*, y 16 elementos vacíos adicionales.

Con el concepto de capacidad, podemos crear una instancia de StringBuilder, que tenga un tamaño menor de la capacidad que tenga. Observa la siguiente imagen y puedes entenderlo mejor:

TAMAÑO Y CAPACIDAD

```
// creates empty builder, capacity 16
StringBuilder sb = new StringBuilder();
// adds 9 character string at beginning
sb.append("Greetings");
```



MÉTODOS DE TAMAÑO Y CAPACIDAD

Método	Descripción
void setLength(int newLength)	Cambia la longitud. Si *newLength* es menor que la actual, los últimos caracteres son truncados. Si es mayor, se añaden elementos vacíos.
void ensureCapacity(int minCapacity)	Nos asegura que la capacidad sea mayor o igual que *minCapacity*.

Algunos métodos, como **append()**, pueden aumentar la capacidad de nuestro StringBuilder.

OPERACIONES CON STRINGBUILDER

Método	Descripción
StringBuilder append(String s) StringBuilder append(tipoPrimitivo t)	Añade el argumento que hemos pasado al StringBuilder. Si el dato no es String, se convierte antes de pasarlo.
StringBuilder delete(int start, int end) StringBuilder deleteCharAt(int index)	Eliminan una secuencia de caracteres o un carácter.
StringBuilder insert(int offset, String s) StringBuilder insert(int offset, tipoPrimitivo t)	Inserta el segundo argumento en la cadena. El primer entero indica la posición.
StringBuilder replace(int start, int end, String s) void setCharAt(int index, char c)	Reemplaza un carácter o una serie de ellos
StringBuilder reverse()	Devuelve la cadena invertida
String toString()	Transforma el StringBuilder en un String.

Observa el ejemplo, creamos un String, y luego lo convierto a StringBuilder. Podría, por ejemplo, invertirlo y visualizarlo. Eso para hacerlo con String, sería más trabajoso, usando dos procesos repetitivos.

```
 */
public static void main(String[] args) {

    String palindromo = "Dabale arroz a la zorra el abad";

    StringBuilder sb = new StringBuilder(palindromo);

    sb.reverse();  // lo invertimos

    System.out.println(sb); //llamada implícita a sb.toString().

}
```

Por tanto, usamos StringBuildir cuando trabajamos con cadenas que van a ser modificada en el tiempo. Y la clase String para cadenas que no van a ser modificadas nunca.