Métodos de la Clase StringBuffer.

TIPO DEVUELTO	NOMBRE Y SIGNATURA DEL MÉTODO	DESCRIPCIÓN DE SU FUNCIÓN.

El objeto StringBuffer que crea, ya que es uno de los constructores disponibles.	`StringBuffer(String str)`	Constructor de un `StringBuffer` a partir de un `String`, o de un literal `String`.
		La clase `StringBuffer` tiene otros constructores, que permiten crear un `StringBuffer` vacío o nulo, o un `StringBuffer` con una determinada capacidad, pero sin caracteres, por lo que serán poco usados.
		
		Ejemplo:
		`String frase= "En un lugar de La Mancha";`
		`StringBuffer fraseModificable=new StringBuffer(frase);`

StringBuffer	`append(String str)`
	`append(StringBuffer sb)`
	`append(int i)`
	`append(double d)`
		Todos los distintos métodos `append()` sobrecargados, realizan la misma funcionalidad: añadir la representación como `String` del valor dado como argumento en la llamada a continuación del valor que tenga el `StringBuffer`. Se obtiene por tanto un objeto `StringBuffer` más largo, como en la concatenación.
		
		Existen en total 13 métodos `append()` sobrecargados, y cada uno de ellos se usará para añadir al `StringBuffer` un tipo diferente, previamente representado como `String`.
		
		Ejemplo:
		Continuando con el ejemplo de la casilla anterior :
		`fraseModificable.append(" de cuyo nombre no quiero acordarme.");`
		`System.out.println(fraseModificable);`
		
		El resultado es que se escribe:
		`En un lugar de La Mancha de cuyo nombre no quiero acordarme.`

char	`charAt(int index)`	Devuelve el carácter (`char`) que se encuentra en la posición que indica el parámetro entero `index`. Debes tener en cuenta que el primer carácter de cada `StringBuffer` se encuentra en la posición 0 (cero). Funciona exactamente igual que en el ejemplo visto para la clase `String`.

int	`capacity()`	Devuelve el total de almacenamiento disponible para hacer modificaciones sobre este `StringBuffer`, sin necesidad de realojar el objeto en una nueva posición de memoria. Si tras las modificaciones se supera esa capacidad, el objeto será realojado en una nueva posición, reservando una cantidad mayor de espacio de almacenamiento.
		
		Ejemplo:
		Continuando con el ejemplo de las casillas anteriores:
		`System.out.println("La frase reserva espacio para "+ fraseModificable.capacity()+ " caracteres.");`
		
		Como resultado se escribirá el mensaje:
		`La frase reserva espacio para 84 caracteres.`
		
		Y sin embargo, si los cuentas, verás que sólo contiene 60 caracteres. El resto son posiciones adicionales para modificar el objeto StringBuffer sin necesidad de realojarlo en memoria.

TIPO DEVUELTO	NOMBRE Y SIGNATURA DEL MÉTODO	DESCRIPCIÓN DE SU FUNCIÓN.

StringBuffer	`delete(int start, int end)`	Borra todos los caracteres entre las posiciones indicadas por los parámetros `start` y `end-1`.
		
		Ejemplo:
		Continuamos con el de las casillas anteriores:
		`fraseModificable.delete(25,43);`
		`System.out.println(fraseModificable);`
		
		Como resultado se escribe la frase:
		`En un lugar de La Mancha quiero acordarme.`

StringBuffer	`deleteCharAt(int index)`	Borra el carácter de la posición indicada del `StringBuffer`.
		
		Ejemplo: (continuando los anteriores)
		
		`fraseModificable.deleteCharAt(41);`
		`System.out.println(fraseModificable);`
		
		Como resultado se escribe la frase:
		`En un lugar de La Mancha quiero acordarme`

StringBuffer	`insert(int offset, String str)`
	
	`insert(int offset, double d)`
	
	`insert(int offset, int i)`
		Todos los distintos métodos `insert()` sobrecargados, realizan la misma funcionalidad: insertar la representación como `String` del valor dado como argumento en la llamada en el valor que tenga el `StringBuffer`, empezando en la posición indicada por `offset`. Se obtiene por tanto un objeto `StringBuffer` más largo.
		
		Existen en total 12 métodos `insert()` sobrecargados, y cada uno inserta la representación como `String` de un tipo diferente.
		
		Ejemplo (continuación de los anteriores):
		
		`fraseModificable.insert(6, ”determinado ”);`
		`System.out.println(fraseModificable);`
		
		Como resultado se escribe la frase:
		`En un determinado lugar de La Mancha quiero acordarme`

int	`indexOf(String str)`
	`indexOf(String str, int fromIndex)`
		Devuelve la posición del `StringBuffer` en la que se ha encontrado la primera ocurrencia del `String str` pasado como parámetro.
		Si adicionalmente recibe el parámetro `fromIndex`, empieza la búsqueda en esa posición.
		
		Ejemplo: (continuando con el anterior)
		
		`int posicion=fraseModificable.indexOf(“de”);`
		`int posicion2=fraseModificable.indexOf(“de”,10);`
		`System.out.println( “posicion=”+posicion +`
		`“posicion2=”+posicion2);`
		
		El resultado es:
		`posicion=6 posicion2=24`

TIPO DEVUELTO	NOMBRE Y SIGNATURA DEL MÉTODO	DESCRIPCIÓN DE SU FUNCIÓN.

int	`lastIndexOf(String str)`	Devuelve la posición del `StringBuffer` en la que se ha encontrado la última ocurrencia (la ocurrencia más a la derecha) del substring `str` pasado como parámetro.
		
		Ejemplo: (continuando con el anterior)
		
		`posicion=fraseModificalble.lastIndexOf("rm");`
		`System.out.println("posicion="+posicion);`
		
		El resultado es:
		`posicion=46`

int	`length()`	MUY USADO: Devuelve un entero que es la longitud o número de caracteres que realmente contiene el `StringBuffer` para el que se ejecuta. Necesitamos usarlo frecuentemente cuando recorremos un `StringBuffer` para realizar alguna manipulación. Este método es similar al visto para la clase `String`

StringBuffer	`replace(int start,`
	`int end,`
	`String str)`
		Devuelve un `StringBuffer` en el que se ha sustituido el substring que empieza en la posición `start` y termina en la posición `end-1` por el `String` indicado como parámetro, `str`.
		
		Ejemplo: (continuando con los ejemplos anteriores)
		`System.out.println(fraseModificable.replace(6,16,`
		`"encantador");`
		El resultado es :
		
		`En un encantador lugar de La Mancha quiero acordarme`

StringBuffer	`reverse()`	Invierte el contenido del `StringBuffer` para el que se ejecuta.
		
		Ejemplo:
		`StringBuffer nombre=new StringBuffer("arroz");`
		`nombre.reverse();`
		`System.out.println(nombre);`
		
		Como resultado se escribirá:
		`Zorra`

void	`setLength(int newLength)`	Permite reservar un nuevo espacio de almacenamiento determinado para el `StringBuffer` para el que se ejecuta. Realmente realoja el objeto en una nueva zona de memoria, en la que se le reserva más espacio libre, en concreto el que se indique con el parámetro newLength.
		
		Si la nueva longitud fijada es mayor que la anterior, los nuevos espacios se llenan con el carácter nulo.
		
		Ejemplo:
		`fraseModificable.setLength(200);`
		`System.out.println(" Espacio reservado o longitud total de la frase: " + fraseModificable.capacity());`
		El resultado será:
		`Espacio reservado o longitud total de la frase: 200`

TIPO DEVUELTO	NOMBRE Y SIGNATURA DEL MÉTODO	DESCRIPCIÓN DE SU FUNCIÓN.

String	`substring(int start, int end)`	Devuelve el substring o subcadena que comienza en la posición `start` y que termina en la posición `end-1,` ambas indicadas como parámetros.
		
		Debemos tener cuidado de no salirnos de los límites del `String`, ya que si lo hacemos se produce una Excepción, un error que hace que aborte el programa. También hay varias modalidades de métodos `substring().` Míralos en la API de Java
		
		Ejemplo:
		`System.out.println(fraseModificable.substring(6,22));`
		
		El resultado sería:
		
		`encantador lugar`