String, StringBuilder y StringBuffer

StringBuffer y StringBuilder

Además de String, existen otras clases para la construcción de cadenas de caracteres, como pueden ser **StringBuffer** y **StringBuilder** que resultan de interés porque facilitan cierto tipo de trabajos y aportan mayor eficiencia en determinados contextos.

StringBuilder vs String

La clase StringBuilder es similar a la clase String pero presenta algunas diferencias relevantes:

- Su tamaño y contenido pueden modificarse a diferencia con los String.
- Debe crearse con alguno de sus constructores asociados. No se permite instanciar directamente a una cadena como sí permiten los String.
- Un StringBuilder está indexado. Cada uno de sus caracteres tiene un índice: 0 para el primero,1 para el segundo, etc.
- Los métodos de StringBuilder no están sincronizados. Esto implica que es más eficiente que StringBuffer siempre que no se requiera trabajar con múltiples hilos (threads), que es lo más habitual (en caso de trabajar con hilos se recomienda StringBuffer).

Constructores de StringBuilder

Constructor	Descripción	Ejemplo
StringBuilder()	Construye un StringBuilder vacío y con una capacidad por defecto de 16 caracteres.	StringBuilder s = new StringBuilder();
StringBuilder(int capacidad)	Se le pasa la capacidad (número de caracteres) como argumento.	StringBuilder s = new StringBuilder(55);
StringBuilder(String str)	Construye un StringBuilder en base al String que se le pasa como argumento.	StringBuilder s = new StringBuilder("hola");

Métodos principales de StringBuilder

Retorno	Método	Explicación
StringBuilder	append(...)	Añade al final del StringBuilder a la que se aplica, un String o la representación en forma de String de un dato asociado a una variable primitiva
int	capacity()	Devuelve la capacidad del StringBuilder
int	length()	Devuelve el número de caracteres del StringBuilder
StringBuilder	reverse()	Invierte el orden de los caracteres del StringBuilder
void	setCharAt(int indice,char ch)	Cambia el carácter indicado en el primer argumento por el carácter que se le pasa en el segundo
char	charAt(int índice)	Devuelve el carácter asociado a la posición que se le indica en el argumento
void	setLength(int nuevaLongitud)	Modifica la longitud. La nueva longitud no puede ser menor

Métodos principales de StringBuilder (2)

Retorno	Método	Explicación
String	toString()	Convierte un StringBuilder en un String
StringBuilder	insert(int indiceIni,String cadena)	Añade la cadena del segundo argumento a partir de la posición indicada en el primero
StringBuilder	delete(int indiceIni,int indiceFin)	Borra la cadena de caracteres incluidos entre los dos índices indicados en los argumentos
StringBuilder	deleteCharAt(int indice)	Borra el carácter indicado en el índice
StringBuilder	replace(int indiceIni, int indiceFin,String str)	Reemplaza los caracteres comprendidos entre los dos índices por la cadena que se le pasa en el argumento
int	indexOf (String str)	Analiza los caracteres de la cadena y encuentra el primer índice que coincide con el valor deseado
String	subString(int indiceIni,int indiceFin)	Devuelve una cadena comprendida entre la posición inicial incluida y la final (no incluida)

StringBuffer

La clase StringBuffer es similar a la clase StringBuilder, con los mismos métodos y constructores, pero sus métodos están sincronizados, permitiendo trabajar con múltiples hilos threads.

Métodos de StringBuilder vs métodos de String

El uso de los métodos de StringBuilder y StringBuffer difiere un poco de los String, ya que como estos últimos eran inmutables, devolvían el resultado del método, pero no modificaban el objeto en sí, por ejemplo:

cad.toUpperCase();

no modificaba cadena así que hay que llamarla así:

cad=cad.toUpperCase();

Esto no ocurre con los StringBuilder/StringBuffer, los métodos sí modifican el contenido del objeto, así podríamos hacer directamente:

StringBuilder sb = new StringBuilder ("abcdefg");
sb.delete(3,5);

Métodos de StringBuilder vs métodos de String (2)

Es frecuente convertir String a StringBuider si necesitamos un método de una de las clases que está disponible en la otra. Por ejemplo, si necesitamos eliminar en la tercera posición de una cadena, podríamos hacer:

```
String cadena = "abcdef";
StringBuilder sb = new StringBuilder(cadena);
sb.deleteCharAt(3);
cadena = sb.toString();
```

abreviando:

```
cadena = new StringBuilder(cadena).deleteCharAt(3).toString();
```