6.B. Cadenas de caracteres.

1. Cadenas de caracteres.

1.5. Operaciones avanzadas con cadenas de caracteres (V).

¿Sabes cuál es el principal problema de las cadenas de caracteres? Su alto consumo de memoria. Cuando realizamos un programa que realiza muchísimas operaciones con cadenas, es necesario optimizar el uso de memoria.

En Java, `String` es un objeto inmutable, lo cual significa, entre otras cosas, que cada vez que creamos un `String`, o un literal de `String`, se crea un nuevo objeto que no es modificable. Java proporciona la clase `StringBuilder`, la cual es un mutable, y permite una mayor optimización de la memoria. También existe la clase `StringBuffer`, pero consume mayores recursos al estar pensada para aplicaciones multi-hilo, por lo que en nuestro caso nos centraremos en la primera.

Pero, ¿en que se diferencian `StringBuilder` de la clase `String`? Pues básicamente en que la clase `StringBuilder` permite modificar la cadena que contiene, mientras que la clase `String` no. Como ya se dijo antes, al realizar operaciones complejas se crea una nueva instancia de la clase `String`.

Veamos un pequeño ejemplo de uso de esta clase. En el ejemplo que vas a ver, se parte de una cadena con errores, que modificaremos para ir haciéndola legible. Lo primero que tenemos que hacer es crear la instancia de esta clase. Se puede inicializar de muchas formas, por ejemplo, partiendo de un literal de cadena:

```
StringBuilder strb=new StringBuilder ("Hoal Muuundo");
```

Y ahora, usando los métodos `append` (insertar al final), `insert` (insertar una cadena o carácter en una posición específica), `delete` (eliminar los caracteres que hay entre dos posiciones) y `replace` (reemplazar los caracteres que hay entre dos posiciones por otros diferentes), rectificaremos la cadena anterior y la haremos correcta:

1. `strb.delete(6,8)`; Eliminamos las '`uu`' que sobran en la cadena. La primera '`u`' que sobra está en la posición 6 (no olvides contar el espacio), y la última '`u`' a eliminar está en la posición 7. Para eliminar dichos caracteres de forma correcta hay que pasar como primer argumento la posición 6 (posición inicial) y como segundo argumento la posición 8 (posición contigua al último carácter a eliminar), dado que la posición final no indica el último carácter a eliminar, sino el carácter justo posterior al último que hay que eliminar (igual que ocurría con el método `substring`).
2. `strb.append ("!")`; Añadimos al final de la cadena el símbolo de cierre de exclamación.
3. `strb.insert (0,"¡");` Insertamos en la posición 0, el símbolo de apertura de exclamación.
4. `strb.replace (3,5,"la")`; Reemplazamos los caracteres 'al' situados entre la posición inicial 3 y la posición final 4, por la cadena '`la`'. En este método ocurre igual que en los métodos `delete` y `substring`, en vez de indicar como posición final la posición 4, se debe indicar justo la posición contigua, es decir 5.

`StringBuilder` contiene muchos métodos de la clase `String` (`charAt, indexOf, lenght, substring, replace, setCharAt,` etc.), pero no todos, pues son clases diferentes con funcionalidades realmente diferentes.

Debes conocer

En la siguiente página puedes encontrar más información (en inglés) sobre como utilizar la clase `StringBuilder`.

Uso de la clase StringBuilder.

Autoevaluación

Rotar una cadena es poner simplemente el primer carácter al final, y retroceder el resto una posición. Después de unas cuantas rotaciones la cadena queda igual. ¿Cuál de las siguientes expresiones serviría para hacer una rotación (rotar solo una posición)?

- ○ stb.delete (0,1); strb.append(stb.charAt(0));
- ○ strb.append(strb.charAt(0));strb.delete(0, 1);
- ○ strb.append(strb.charAt(0));strb.delete(0);
- ○ strb.append(strb.charAt(1));strb.delete(1,2);