

Bloques de texto

Bloques de texto

- Es posible representar un texto que contenga caracteres especiales, como saltos de línea y comillas, delimitándolo entre triples comillas: `"""` y `"""`
- Muy útil, por ejemplo, para JSON:

```
String json="""  
    {  
        "nombre":"pepito",  
        "telefonos":[1111,2222]  
    }""";
```

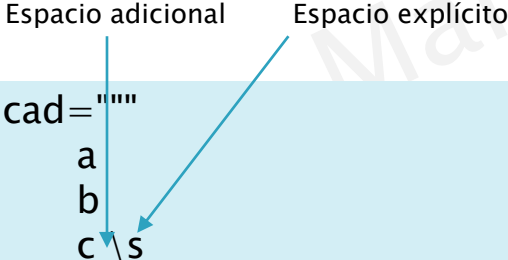
Delimitador de
inicio en línea
independiente

- Se respetan todos los caracteres de la cadena, incluidos saltos de línea

Espacios

➤ En bloques de texto multilínea, los espacios antes del salto de línea son eliminados automáticamente, a no ser que se indiquen explícitamente (carácter \s)

➤ Ejemplo:



```
String cad=""  
a  
b  
c\s  
""",  
System.out.println(cad); // a  
                                b  
                                c  
System.out.println(cad.length()); // 8
```

Los espacios que hubiera después de a, b y \s son eliminados automáticamente

Eliminación de saltos de línea

- Si no queremos que se introduzca un salto al final de cada línea, se utilizará el símbolo (\)
- No podrá indicarse ningún carácter después de dicho símbolo
- Ejemplo:

```
String cad=""  
    a \  
    b\  
    c  
    """,  
System.out.println(cad)); // a bc  
System.out.println(cad.length()); // 5
```

Hay que contar el salto de línea después del carácter c, que no se ha eliminado

Revisión conceptos



Dado el siguiente bloque de código, ¿Cuál será el valor mostrado?:

```
String cadena=""  
    r  
    \s\  
    \  
    """,  
System.out.println(cadena.length());
```

- a. 3
- b. 4
- c. 5
- d. 6

Respuesta

La respuesta es la **a**. La cadena tiene un total de 3 caracteres: r, \n y \s. Los saltos de línea de las líneas segunda y tercera son eliminados

Indentación

- En Java 12 la clase `String` incorpora el método `indent(int n)` para añadir espacios en una cadena
- Añade tantos espacios al principio de cada línea como se indique en el número, si este es negativo los elimina
- Incluye un salto de línea final si no existe:

```
String mycad=""  
    x  
    y  
    z"";  
System.out.println(mycad.length()); //5  
System.out.println(mycad.indent(1).length()); //9  
System.out.println(mycad.indent(-1).length()); //6
```

hay que
contar el salto
de línea final

Método stripIndent()

➤ Método incorporado a la clase String en Java 15 que, tras una concatenación, elimina los espacios existentes antes de un salto de línea.

```
String c1="x \n";  
String c2=" y \n";  
System.out.println((c1+c2).stripIndent().length()); //5
```

Elimina estos espacios



Este espacio no es
eliminado



Revisión conceptos



Dado el siguiente bloque de código, ¿Cuál será el valor mostrado?:

```
String c1=" a \n";  
String c2=""  
    b  
    c\  
    d  
    "";  
System.out.println((c1+c2).stripIndent().indent(-1).length());
```

- a. 5
- b. 8
- c. 11
- d. 12

Respuesta

La respuesta es la **b**. Al aplicar `stripIndent()` eliminamos el espacio entre la a y la b. Con `indent()`, eliminamos los espacios delante de a y de c, pero no de d, puesto que forma parte de la línea anterior. Por tanto la cadena resultante es: `a\nb\nc d\n`