

6.B. Cadenas de caracteres.

1. Cadenas de caracteres.

1.4. Operaciones avanzadas con cadenas de caracteres (IV).

¿Cómo puedo comprobar si dos cadenas son iguales? ¿Qué más operaciones ofrece Java sobre las cadenas? **Java ofrece un montón de operaciones más sobre. En la siguiente tabla puedes ver las operaciones más importantes.** En todos los ejemplos expuestos, las variables `cad1`, `cad2` y `cad3` son cadenas ya existentes, y la variable `num` es un número entero mayor o igual a cero.

Métodos importantes de la clase String.	
Método.	Descripción
<code>cad1.compareTo(cad2)</code>	Permite comparar dos cadenas entre sí lexicográficamente. Retornará 0 si son iguales, un número menor que cero si la cadena (<code>cad1</code>) es anterior en orden alfabético a la que se pasa por argumento (<code>cad2</code>), y un número mayor que cero si la cadena es posterior en orden alfabético.
<code>cad1.equals(cad2)</code>	Cuando se comparan si dos cadenas son iguales, no se debe usar el operador de comparación "==", sino el método <code>equals</code> . Retornará <code>true</code> si son iguales, y <code>false</code> si no lo son.
<code>cad1.compareToIgnoreCase(cad2)</code> <code>cad1.equalsIgnoreCase(cad2)</code>	El método <code>compareToIgnoreCase</code> funciona igual que el método <code>compareTo</code> , pero ignora las mayúsculas y las minúsculas a la hora de hacer la comparación. Las mayúsculas van antes en orden alfabético que las minúsculas, por lo que hay que tenerlo en cuenta. El método <code>equalsIgnoreCase</code> es igual que el método <code>equals</code> pero sin tener en cuenta las minúsculas.
<code>cad1.trim()</code>	Genera una copia de la cadena eliminando los espacios en blanco anteriores y posteriores de la cadena.
<code>cad1.toLowerCase()</code>	Genera una copia de la cadena con todos los caracteres a minúscula.
<code>cad1.toUpperCase()</code>	Genera una copia de la cadena con todos los caracteres a mayúsculas.
<code>cad1.indexOf(cad2)</code> <code>cad1.indexOf(cad2, num)</code>	Si la cadena o carácter pasado por argumento está contenida en la cadena invocante, retorna su posición, en caso contrario retornará -1. Opcionalmente se le puede indicar la posición a partir de la cual buscar, lo cual es útil para buscar varias apariciones de una cadena dentro de otra.
<code>cad1.contains(cad2)</code>	Retornará <code>true</code> si la cadena pasada por argumento está contenida dentro de la cadena. En caso contrario retornará <code>false</code> .

Métodos importantes de la clase String.

Método.	Descripción
<code>cad1.startsWith(cad2)</code>	Retornará <code>true</code> si la cadena comienza por la cadena pasada como argumento. En caso contrario retornará <code>false</code> .
<code>cad1.endsWith(cad2)</code>	Retornará <code>true</code> si la cadena acaba por la cadena pasada como argumento. En caso contrario retornará <code>false</code> .
<code>cad1.replace(cad2,cad3)</code>	Generará una copia de la cadena <code>cad1</code> , en la que se reemplazarán todas las apariciones de <code>cad2</code> por <code>cad3</code> . El reemplazo se hará de izquierda a derecha, por ejemplo: reemplazar <code>"zzz"</code> por <code>"xx"</code> en la cadena <code>"zzzzz"</code> generará <code>"xxzzz"</code> y no <code>"zzxx"</code> .

Autoevaluación

¿Cuál será el resultado de ejecutar `cad1.replace("l","j").indexOf("ja")` si `cad1` contiene la cadena `"hojalata"`?

- ☒ 2.
☐ 3.
☐ 4.
☐ -1.