5. Cadenas de caracteres

La **clase String** de Java combina algunas de las posibilidades que nos ofrecen los arrays (acceder a una posición concreta, conocer su longitud, etcétera) con otras opciones adicionales que irás conociendo a lo largo de este apartado. La clase String está incluida en uno de los paquetes que Java carga por defecto, así que no hace falta que la importemos en nuestras aplicaciones, sino que la podemos usar directamente.

Para crear una cadena de caracteres, tan solo tenemos que declarar una variable de tipo String. Hay que tener en cuenta, sin embargo, que cuando creamos una cadena sin asignarle un valor, dicha cadena no estará vacía: su valor será **null** (objeto sin valor).

```
String cadena; // cadena a null (sin inicializar)
```

Por ello, es posible que el sistema nos dé un error si intentamos acceder a su contenido sin haberlo definido anteriormente. Por eso, para crear una cadena vacía que no sabemos si más adelante se rellenará o no, la mejor opción es:

```
String cadena = ""; // cadena vacía
```

Si, en el momento de la creación de la cadena, le queremos dar un valor por defecto, lo podemos hacer como con cualquier otro tipo de dato:

```
String cadena = "Hola"; // cadena con contenido inicial
```

Para pedir el contenido de una cadena por consola, disponemos de los métodos next() y nextLine() de la clase Scanner. La diferencia es que, con next(), obtenemos el texto introducido hasta el siguiente espacio (una palabra), mientras que con nextLine() obtenemos el texto hasta el final de línea (varias palabras y espacios en blanco).

IMPORTANTE

Si al pedir el valor de un String con next() se introducen espacios, todos los caracteres a partir del primer espacio no se guardarán en la variable, pero quedarán almacenados en el buffer y serán leídos automáticamente por el siguiente next() o nextLine() que aparezca en la aplicación. Hay que estar pendientes de esto, puesto que puede generar un mal funcionamiento de la aplicación.

Una particularidad de las cadenas de caracteres es que, al ser objetos de la clase String, la comparación de su contenido no debe hacerse con el operador «==». Como veremos más adelante, dicho operador aplicado a objetos no compara su contenido, sino el objeto en sí. Para comparar dos cadenas de caracteres y ver si tienen el mismo contenido, usaremos una funcionalidad (un método, en realidad, como veremos al tratar la programación orientada a objetos) de la clase String llamado «equals», que devuelve TRUE si el contenido de ambas cadenas coincide.

Comparación del contenido de dos cadenas de caracteres en Java	
~~If(s1==s2)…~~	If(s1.equals(s2))…
Incorrecto	Correcto

Tabla 4.1. Comparación correcta del contenido de dos cadenas de caracteres

IMPORTANTE

El método **equals()** distingue entre mayúsculas y minúsculas, por lo que, si comparamos dos cadenas como «Madrid» y «madrid» devolverá FALSE. Si no queremos diferenciar entre mayúsculas y minúsculas, debemos usar en su lugar el método **equalsIgnoreCase()**.

Otros métodos de la clase String usados con frecuencia son:

Método	Descripción	Ejemplo

length()	Devuelve la longitud de la cadena.	`String s = "Hola mundo";`
		`int l = s.length();`
		`// devuelve 10`

charAt(pos)	Devuelve el carácter que ocupa la posición requerida (la primera posición es la 0).	`String s = "Hola mundo";` `char c = s.charAt(3); // devuelve 'a'`

indexOf(subcadena)	Devuelve la posición donde empieza la subcadena dentro de la cadena principal (-1 en caso de que no se encuentre).	`String s = "Hola mundo";`
		`int p = s.indexOf("mundo");`
		`// devuelve 5`

substring(pos1, pos2)
substring(pos1)
	Devuelve la parte de la cadena entre las posiciones pos1 (incluida) y pos2 (no incluida). Si se omite pos2, devuelve la parte de la cadena desde pos1 hasta el final de la cadena.	`String s = "Hola mundo";`
		`String s1 = s.substring(5);`
		`// devuelve "mundo"`
		`String s2 = s.substring(5,7);`
		`// devuelve "mu"`

toUpperCase()	Devuelve la cadena convertida en mayúsculas.	`String s = "Hola mundo";`
		`String s1 = s.toUpperCase();`
		`// devuelve "HOLA MUNDO"`

toLowerCase()	Devuelve la cadena convertida en minúsculas.	`String s2 = s.toLowerCase();` `// devuelve "hola mundo"`

trim()	Devuelve la cadena tras quitar los espacios del principio y del final de la cadena.	`String s = "  Hola mundo  ";`
		`String s1 = s.trim();`
		`// devuelve "Hola mundo"`

split(separador)	Separa la cadena en subcadenas a partir del carácter definido como separador. Devuelve un array de cadenas.	`String s = "Hola mundo";`
		`String[] palabras = s.split(" ");`
		`// palabras[0] guarda "Hola"`
		`// palabras[1] guarda "mundo"`

concat(cadena2)	Devuelve la concatenación de la cadena original con la cadena 2 (equivalente a usar el operador + con cadenas).	`String s = "Hola ";`
		`String s2 = s.concat("mundo");`
		`// s2 guarda "Hola mundo"`

Tabla 4. 2. Otros métodos de la clase String

Existen muchos otros métodos que, como te hemos comentado anteriormente, podemos explorar en la documentación oficial de Java.

https://docs.oracle.com/en/java/javase/16/docs/api/java.base/java/lang/String.html

Como ocurre con cualquier otro tipo de datos, podemos tener arrays o matrices cuyo contenido sean cadenas. Por ejemplo, supón que queremos pedir desde la consola 10 nombres y guardarlos. Más efectivo que crear 10 variables String y pedirlas una por una sería crear un array de String y pedir los datos con un bucle, tal como hemos visto en apartados anteriores.

El siguiente código pide 10 palabras, las guarda en un array de String y, posteriormente, las muestra por pantalla:

```
String[] arrayString = new String[10];
Scanner teclado = new Scanner(System.in);
for(int indice = 0; indice < arrayString.length ; indice++) {
     System.out.print("Introduce la palabra número " + (indice+1) + ": ");
     arrayString[indice] = teclado.next();
}
System.out.println("=== PALABRAS INTRODUCIDAS ===");
for(int indice = 0; indice < arrayString.length; indice++) {
     System.out.println("Palabra número " + (indice+1) + ": " +
arrayString[indice]);
}
```