# STRING COMO CADENAS

Una cadena es una secuencia de caracteres. Las cadenas son una parte fundamental de la mayoría de los programas, así pues Java tiene varias características incorporadas que facilitan la manipulación de cadenas. Java tiene una clase incorporada en el paquete java.lang que encapsula las estructuras de datos de una cadena. Esta clase, llamada String es la representación como objeto de una matriz de caracteres que no se puede cambiar. Hay una clase que la acompaña, llamada StringBuffer, que se utiliza para crear cadenas que pueden ser manipuladas después de ser creadas.

**Creación de cadenas**

Muchos Strings se crean a partir de cadenas literales. Cuando el compilador encuentra una serie de caracteres entre comillas (" y "), crea un objeto String cuyo valor es el propio texto. El esquema general es el siguiente: String nombre="cadena"; Cuando el compilador encuentra la siguente cadena, crea un objeto String cuyo valor es Hola Mundo.

**"Hola Mundo"**

o también

**String s = "Hola Mundo"**

También se pueden crear objetos String como se haría con cualquier otro objeto Java: utilizando new.

**String s = new String("Hola Mundo.");**

El constructor anterior es equivalente, pero es mucho más eficiente el primer método ya que, el segundo método crea dos objetos String en vez de sólo uno.

Se pueden utilizar cadenas literales en cualquier lugar donde se pueda utilizar un objeto String. Por ejemplo, System.out.println() acepta un argumenteo String, por eso se puede utilizar una cadena literal en su lugar:

**System.out.println("Hola Mundo!");**

**Uso de cadenas**

**Concatenación de cadenas**

Java permite concatenar cadenas facilmente utilizando el operador +. El siguiente fragmento de código concatena tres cadenas para producir su salida:

"La entrada tiene " + contador + " caracteres."

Dos de las cadenas concatenadas son cadenas literales: "La entrada tiene " y " caracteres.". La tercera cadena - la del medio- es realmente un entero que primero se convierte a cadena y luego se concatena con las otras.

**Longitud de cadenas**

uno de los métodos mas habituales que se utilizan en un String es length, que devuelve el n§. de caracteres de una cadena:

**String s = "abc";**

**System.out.println(s.length());**

El resultado de ejecutar el código anterior, sería la impresión de 3, que se corresponde con la longitud llamada s.

Un punto interesante en Java es que se crea una instancia de objeto para cada literal String, por lo que se puede llamar a los métodos directamente con una cadena entre comillas, como si fuera una referencia a objeto, con este ejemplo se volvería a imprimir un 3:

**String s = "abc";**

**System.out.println("abc".lenght());**

**Extracción de caracteres**

Para extraer un único carácter de una cadena, se puede referir a un carácter indexado mediante el método charAt, la sintaxis es la siguiente Objeto_cadena.charAt(índice);

**"abc".charAt(1)**

**Devolverá 'b'**

Si se necesita extraer más de un carácter a la vez, puede utilizar el método getChars, que le permite especificar el índice del primer carácter y del último más uno que se desean copiar, además de la matriz char donde se desean colocar dichos caracteres.

```
String s = "Esto no es una canción";

char buf[] = new char[2];

s.getChars(5, 7, buf, 0);
```

buf ahora tendrá el valor 'no'

También existe una función útil llamada toCharArray, que devuelve una matriz de char que contiene la cadena completa.

**Comparación de cadenas**

Si se desean comparar dos cadenas para ver si son iguales, puede utilizar el método equals de String. Devolverá true si el único parámetro está compuesto de los mismos caracteres que el objeto con el que se llama a equals. Una

forma alternativa de equals llamada equalsIgnoreCase ignora si los caracteres de las cadenas que se comparan están en mayúsculas o minúsculas.

**Objeto_cadena1.equals(Objeto_cadena2);**

**Objeto_cadena1.equalsIgnoreCase(Objeto_cadena2);**

Veamos ahora un **ejemplo:**

```
String cadena1="pepe";

String cadena2="juan";

if (cadena1.equals(cadena2)) {

//Ambas cadenas son iguales (no es el caso)

}

{  //Ambas cadenas son iguales (es el caso)

}
```

**Igualdad**

El método equals y el operador = = hacen dos pruebas completamente diferentes para la igualdad. Mientras que el método equals compara los caracteres contenidos en una String, el operador = = compara dos referencias de objeto para ver si se refieren a la misma instancia. Por tanto, no podemos usar el signo == por que esta sería una comparación binaria de punteros a memoria y no nos devolvería el valor correcto

**Comparación con CompareTo**

Si lo que queremos es comparar cadenas para ordenarlas, una opción es usar el método compareTo() de la clase String. Este método devuelve 0 si ambas cadenas tienen el mismo contenido, negativo si el String es menor -va antes- que el parámetro que se le pasa y positivo si es mayor. Es decir:

```
if (cadena1.compareTo(cadena2) == 0)

System.out.println("cadena1 y cadena2 son iguales");

else
```

```
if (cadena1.compareTo(cadena2) < 0)

System.out.println ("cadena1 va antes que cadena2");

else

if (cadena1.compareTo(cadena2) > 0)

System.out.println("cadena2 va después que cadena1");
```

**Búsqueda en la cadena**

Presenta los siguientes métodos para buscar caracteres o subcadenas en la cadena, y devuelven el índice que han encontrado o el valor –1 si la búsqueda no ha sido satisfactoria:

**int indexOf( char ch, int start ):** Devuelve el índice correspondiente a la primera aparición del carácter ch en la cadena, comenzando a buscar desde el carácter start (si no se especifica se busca desde el principio).

**int indexOf( String str ):** Devuelve el índice correspondiente al carácter en que empieza la primera aparición de la subcadena str.

**int lastIndexOf( char ch, int start ):** Devuelve el índice correspondiente a la última aparición del carácter ch en la cadena, comenzando a buscar desde el carácter start (si no se especifica se busca desde el final).

**int lastIndexOf( String str ):** Devuelve el índice correspondiente al carácter en que empieza la última aparición de la subcadena str.

```
Objeto_cadena.indexOf(‘carácter’);

Objeto_cadena.indexOf(“cadena”);

Objeto_cadena.lastIndexOf(‘carácter’);

Objeto_cadena.lastIndexOf(“cadena”);
```

**Conversión de minúsculas a mayúsculas de una cadena**

```
Objeto_cadena.toLowerCase(); // Lo convierte a minúsculas.

Objeto_cadena.toUpperCase(); // Lo convierte a mayúsculas.
```