# Cadena Java

https://docs.oracle.com/javase/7/docs/api/

En Java, la cadena es básicamente un objeto que representa la secuencia de valores de caracteres. Una matriz de caracteres funciona igual que una cadena Java. Por ejemplo:

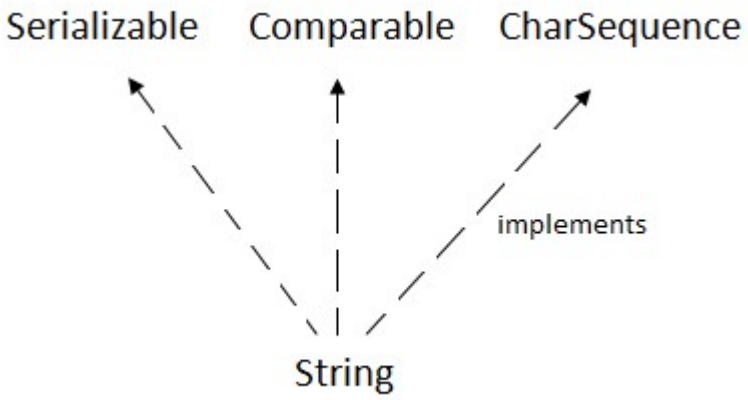
```
1.  char [] ch = { 'j' , 'q' , 'v' , 'r , 't' , 'j' , 'f' , 'd' , 'n' , 't' };
2.  Cadena s = nueva  Cadena (ch);
```

es igual que:

```
1.  Cadena s = "javatpoint" ;
```

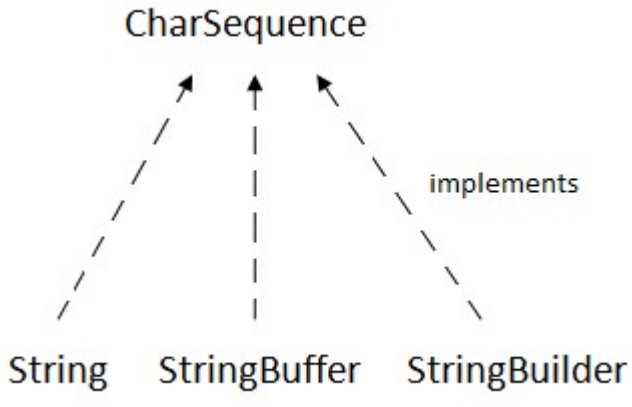
**La** clase **Java String** proporciona muchos métodos para realizar operaciones en cadenas como compare (), concat (), equals (), split (), length (), replace (), compareTo (), intern (), substring () etc.

La clase java.lang.String implementa interfaces *serializables* , *comparables* y *CharSequence* .




# Interfaz de secuencia de caracteres

La interfaz CharSequence se usa para representar la secuencia de caracteres. Las clases String, StringBuffer y StringBuilder lo implementan. Significa que podemos crear cadenas en Java usando estas tres clases.




La cadena de Java es inmutable, lo que significa que no se puede cambiar. Cada vez que cambiamos cualquier cadena, se crea una nueva instancia. Para cadenas mutables, puede usar las clases StringBuffer y StringBuilder.

Discutiremos la cadena inmutable más tarde. Primero, comprendamos qué es String en Java y cómo crear el objeto String.

# ¿Qué es una cadena en Java?

Generalmente, la cadena es una secuencia de caracteres. Pero en Java, la cadena es un objeto que representa una secuencia de caracteres. La clase java.lang.String se usa para crear un objeto de cadena.

## ¿Cómo crear un objeto de cadena?

Hay dos formas de crear un objeto String:

1. Por cadena literal
2. Por nueva palabra clave

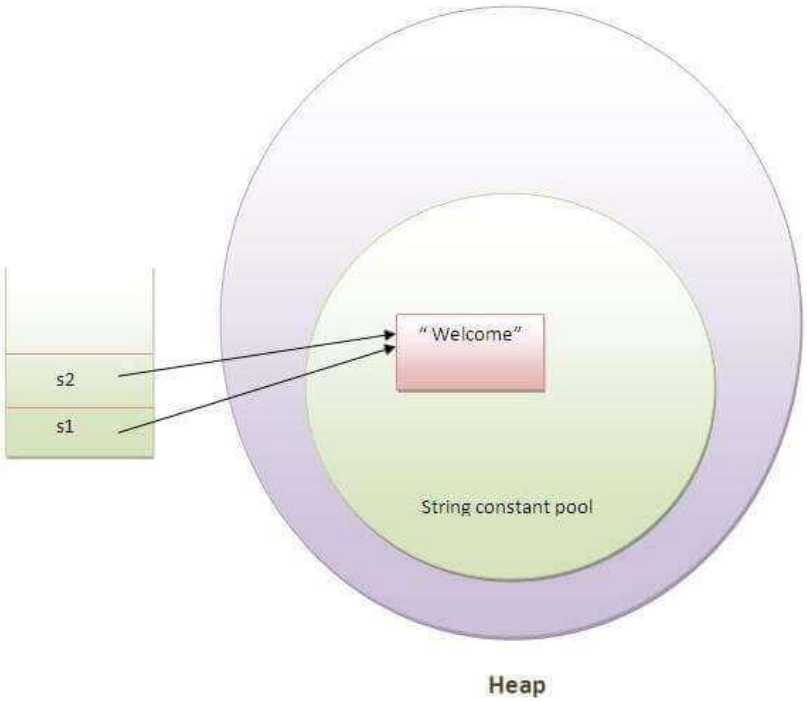
# 1) Cadena literal

El literal de cadena de Java se crea mediante comillas dobles. Por ejemplo:

```
1.	Cadena s = "bienvenido" ;
```

Cada vez que crea un literal de cadena, la JVM comprueba primero el "grupo constante de cadena". Si la cadena ya existe en el grupo, se devuelve una referencia a la instancia agrupada. Si la cadena no existe en el grupo, se crea una nueva instancia de cadena y se coloca en el grupo. Por ejemplo:

```
1.	Cadena s1 = "Bienvenido" ;
2.	Cadena s2 = "Bienvenido" ; // No crea una nueva instancia
```




En el ejemplo anterior, solo se creará un objeto. En primer lugar, JVM no encontrará ningún objeto de cadena con el valor "Bienvenido" en el grupo de constante de cadena, es por eso que creará un nuevo objeto. Después de eso, encontrará la cadena con el valor "Bienvenido" en el grupo, no creará un nuevo objeto pero devolverá la referencia a la misma instancia.

> Nota: Los objetos de cadena se almacenan en un área de memoria especial conocida como "grupo de constante de cadena".

## ¿Por qué Java usa el concepto de cadena literal?

Para que Java sea más eficiente en la memoria (porque no se crean objetos nuevos si ya existe en el grupo de cadenas constantes).

# Métodos de clase Java String

La clase java.lang.String proporciona muchos métodos útiles para realizar operaciones en la secuencia de valores de caracteres.

| No. | Method | Description |
|---|---|---|
| 1 | char charAt(int index) | returns char value for the particular index |
| 2 | int length() | returns string length |
| 3 | static String format(String format, Object... args) | returns a formatted string. |
| 4 | static String format(Locale l, String format, Object... args) | returns formatted string with given locale. |
| 5 | String substring(int beginIndex) | returns substring for given begin index. |
| 6 | String substring(int beginIndex, int endIndex) | returns substring for given begin index and end index. |
| 7 | boolean contains(CharSequence s) | returns true or false after matching the sequence of char value. |
| 8 | static String join(CharSequence delimiter, CharSequence... elements) | returns a joined string. |
| 9 | static String join(CharSequence delimiter, Iterable<? extends CharSequence> elements) | returns a joined string. |
| 10 | boolean equals(Object another) | checks the equality of string with the given object. |
| 11 | boolean isEmpty() | checks if string is empty. |
| 12 | String concat(String str) | concatenates the specified string. |

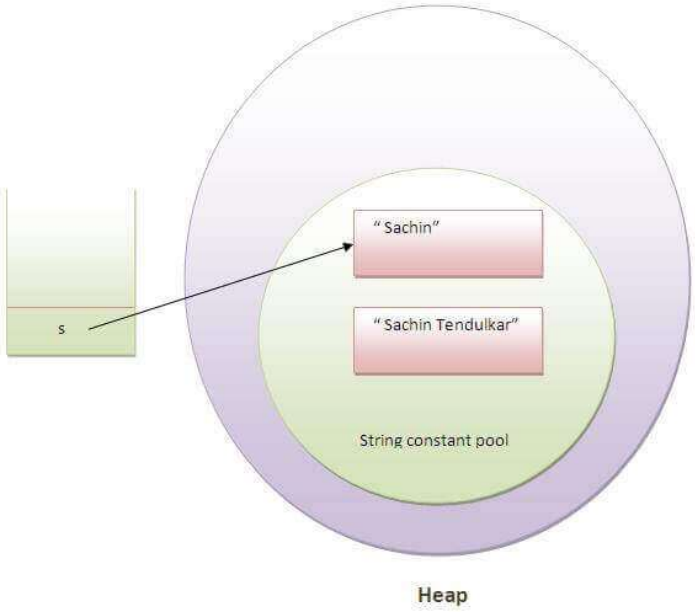
| | | |
|---|---|---|
| 13 | String replace(char old, char new) | replaces all occurrences of the specified char value. |
| 14 | String replace(CharSequence old, CharSequence new) | replaces all occurrences of the specified CharSequence. |
| 15 | static String equalsIgnoreCase(String another) | compares another string. It doesn't check case. |
| 16 | String[] split(String regex) | returns a split string matching regex. |
| 17 | String[] split(String regex, int limit) | returns a split string matching regex and limit. |
| 18 | String intern() | returns an interned string. |
| 19 | int indexOf(int ch) | returns the specified char value index. |
| 20 | int indexOf(int ch, int fromIndex) | returns the specified char value index starting with given index. |
| 21 | int indexOf(String substring) | returns the specified substring index. |
| 22 | int indexOf(String substring, int fromIndex) | returns the specified substring index starting with given index. |
| 23 | String toLowerCase() | returns a string in lowercase. |
| 24 | String toLowerCase(Locale l) | returns a string in lowercase using specified locale. |
| 25 | String toUpperCase() | returns a string in uppercase. |
| 26 | String toUpperCase(Locale l) | returns a string in uppercase using specified locale. |
| 27 | String trim() | removes beginning and ending spaces of this string. |
| 28 | static String valueOf(int value) | converts given type into string. It is an overloaded method. |

# Cadena inmutable en Java

En java, **los objetos de cadena son inmutables** . Inmutable simplemente significa inmodificable o inmutable.

Una vez que se crea el objeto de cadena, sus datos o estado no se pueden cambiar, pero se crea un nuevo objeto de cadena.

Tratemos de entender el concepto de inmutabilidad con el ejemplo que se muestra a continuación:




Como puede ver en la figura anterior, se crean dos objetos pero la variable de referencia de s todavía se refiere a "Sachin", no a "Sachin Tendulkar".

Pero si lo asignamos explícitamente a la variable de referencia, se referirá al objeto "Sachin Tendulkar". Por ejemplo:

## ¿Por qué los objetos de cadena son inmutables en Java?

Debido a que Java utiliza el concepto de cadena literal. Supongamos que hay 5 variables de referencia, todas se refieren a un objeto "sachin". Si una variable de referencia cambia el valor del objeto, se verá afectado por todas las variables de referencia. Es por eso que los objetos de cadena son inmutables en Java.

# Comparación de cadenas de Java

Podemos comparar cadenas en java en función del contenido y la referencia.

Se utiliza en **autenticación** (por el método equals ()), **clasificación** (por el método compareTo ()), **coincidencia de referencias** (por == operador) etc.

Hay tres formas de comparar cadenas en java:

1. Por el método equals ()
2. Por = = operador
3. Por el método compareTo ()

## 1) Comparación de cadenas por el método equals ()

El método String equals () compara el contenido original de la cadena. Compara valores de cadena para igualdad. La clase de cadena proporciona dos métodos:

- **public boolean equals (Object another)** compara esta cadena con el objeto especificado.
- **public boolean equalsIgnoreCase (String another)** compara esta cadena con otra cadena, ignorando mayúsculas y minúsculas.

## 2) Comparación de cadenas por == operador

## 3) Comparación de cadenas por el método compareTo ()

El método String compareTo () compara valores lexicográficamente y devuelve un valor entero que describe si la primera cadena es menor, igual o mayor que la segunda cadena.

Supongamos que s1 y s2 son dos variables de cadena. Si:

- **s1 == s2** : 0
- **s1> s2** : valor positivo
- **s1 <s2** : valor negativo

# Concatenación de cadenas en Java

En java, la concatenación de cadenas forma una nueva cadena *que es* la combinación de múltiples cadenas. Hay dos formas de concatenar cadenas en Java:

1. Por operador + (concatenación de cadenas)
2. Por el método concat ()

## 1) Concatenación de cadenas por operador + (concatenación de cadenas)

El operador de concatenación de cadenas Java (+) se usa para agregar cadenas. Por ejemplo:

El **compilador de Java transforma el** código anterior a esto:

```
1.  String s = ( new  StringBuilder ()). Append ( "Sachin" ) .append ("Tendulkar) .toString ();
```

En java, la concatenación de cadenas se implementa a través de la clase StringBuilder (o StringBuffer) y su método append. El operador de concatenación de cadenas produce una nueva cadena al agregar el segundo operando al final del primer operando. El operador de concatenación de cadenas puede concatenar no solo cadenas sino también valores primitivos. Por ejemplo:

> Nota: Después de un literal de cadena, todos los + serán tratados como operador de concatenación de cadenas.

## 2) Concatenación de cadenas por el método concat ()

El método String concat () concatena la cadena especificada al final de la cadena actual. Sintaxis:

Cadena **pública** concat (Cadena otra)

Veamos el ejemplo del método String concat ().

# Subcadena en Java

Una parte de la cadena se llama **subcadena** . En otras palabras, la subcadena es un subconjunto de otra cadena. En caso de subcadena, startIndex es inclusivo y endIndex es exclusivo.

> Nota: El índice comienza desde 0.

Puede obtener una subcadena del objeto de cadena dado mediante uno de los dos métodos:

1. **subcadena de cadena pública (int startIndex):** este método devuelve un nuevo objeto de cadena que contiene la subcadena de la cadena dada de startIndex especificado (inclusive).
2. **Subcadena de cadena pública (int startIndex, int endIndex):** este método devuelve un nuevo objeto String que contiene la subcadena de la cadena dada desde startIndex especificado a endIndex.

En caso de cuerda:

- **startIndex:** inclusive
- **endIndex:** exclusivo

Vamos a entender el startIndex y endIndex por el código que figura a continuación.

```
1.  Cadena s = "hola" ;
2.  System.out.println (s.substring ( 0 , 2 )); //él
```

En la subcadena anterior, 0 puntos a h pero 2 puntos a e (porque el índice final es exclusivo).