**Introduction :**

* In C/C++ - **strings** appear to be an array of characters. But in Java - **strings** are objects of pre-defined class.
* **strings** in Java, do not have a null character at the end.
* **strings** are a sequence of Unicode characters.
* **string** is a pre-defined class present in **java.lang** package.
* It is a final class that means **string** class cannot be inherited.
* **string** is the child class of an Object class and implements the CharSequence, Comparable, Serializable interfaces.
* Each quoted string is an object of the String class and thus it occupies memory from heap, in the following example – **s** which is a reference of **String** class, must have a **HASH Code** value.

**class demo**

**{**

**public static void main(String args[])**

**{**

**String s = “Hello”;**

**System.out.println(s1.hashCode());**

**}**

**}**

i.e., when **string s = “hello”**, JVM implicitly invokes the string class constructor to allocate memory to heap. (Such a statement implicitly invokes a new operator to allocate memory from heap.)

* **string objects** are created inside the string pool.
* **string class** is immutable i.e., **strings** in Java, once created and initialized, cannot be changed on the same reference. If one changes the content of the original string object, then an entirely new string object is created.
* The **java.lang.String** class is final. This implies that no class can extend it.
* The **java.lang.String** class implements **Serializable**, **Comparable** and **CharSequence** interfaces.



**Note:** In Java, string is basically an object that represents sequence of char values. An array of characters works same as Java string.

|  |  |
| --- | --- |
| **char[] ch={'h','e','l','l','o'};**  **String s=new String(ch);** | **String s="hello";** |

In this above example, both are same

**How to create a string object?**

There are two ways to create String object -

* **By string literal**
* **By new keyword**

**By string literal -**

Java String literal is created by using double quotes.

**String s = "hello";**

Each time you create a string literal, the JVM checks the "**string constant pool**" first. If the string already exists in the pool, a reference to the pooled instance is returned. If the string doesn't exist in the pool, a new string instance is created and placed in the pool.

**String s1 = "hello";**

**String s2 = "hello"; //It doesn't create a new instance**

**By new keyword -**

**String s=new String("hello"); //creates two objects and one reference variable**

In such case, JVM will create a new string object in normal (**non-pool**) heap memory, and the literal "hello" will be placed in the string constant pool. The variable s will refer to the object in a heap (non-pool).

**Example –**

* **Converting char array to string**

**class demo**

**{**

**public static void main(String args[])**

**{**

**char arr[] = {‘h’,’e’,’l’,’l’,’o’};**

**String s = new String(arr);**

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

**}**

**}**

* **Creating string by java string literal**

**class demo**

**{**

**public static void main(String args[])**

**{**

**String s = “hello”;**

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

**}**

**}**

* **Creating java string by new keyword**

**class demo**

**{**

**public static void main(String args[])**

**{**

**String s=new String("hello"); System.out.println(s);**

**}**

**}**