# **String class in Java**

The string is a sequence of characters. In Java, objects of String are immutable which means a constant and cannot be changed once created. It is automatically thread safe.

### **Creating a String**

There are two ways to create string in Java:

#### ***1. String literal***

String s = “Hello World”;

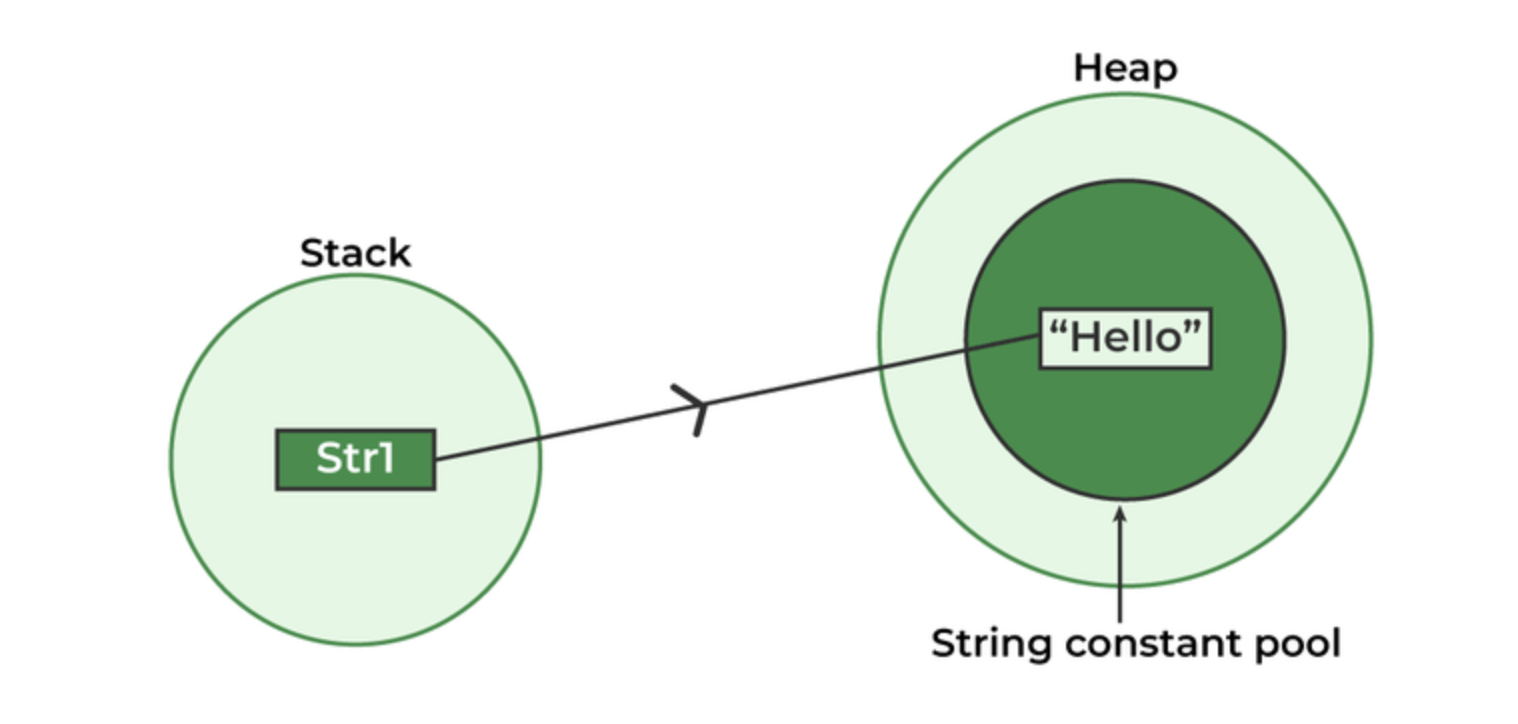
#### **2. Using *new* keyword**

String s = new String (“Hello World”);

## **What is Java String Pool?**

A Java String Pool is a place in heap memory where all the strings defined in the program are stored. A separate place in a stack is there where the variable storing the string is stored. Whenever we create a new string object, JVM checks for the presence of the object in the String pool, If String is available in the pool, the same object reference is shared with the variable, else a new object is created.

**String str1 = "Hello";**

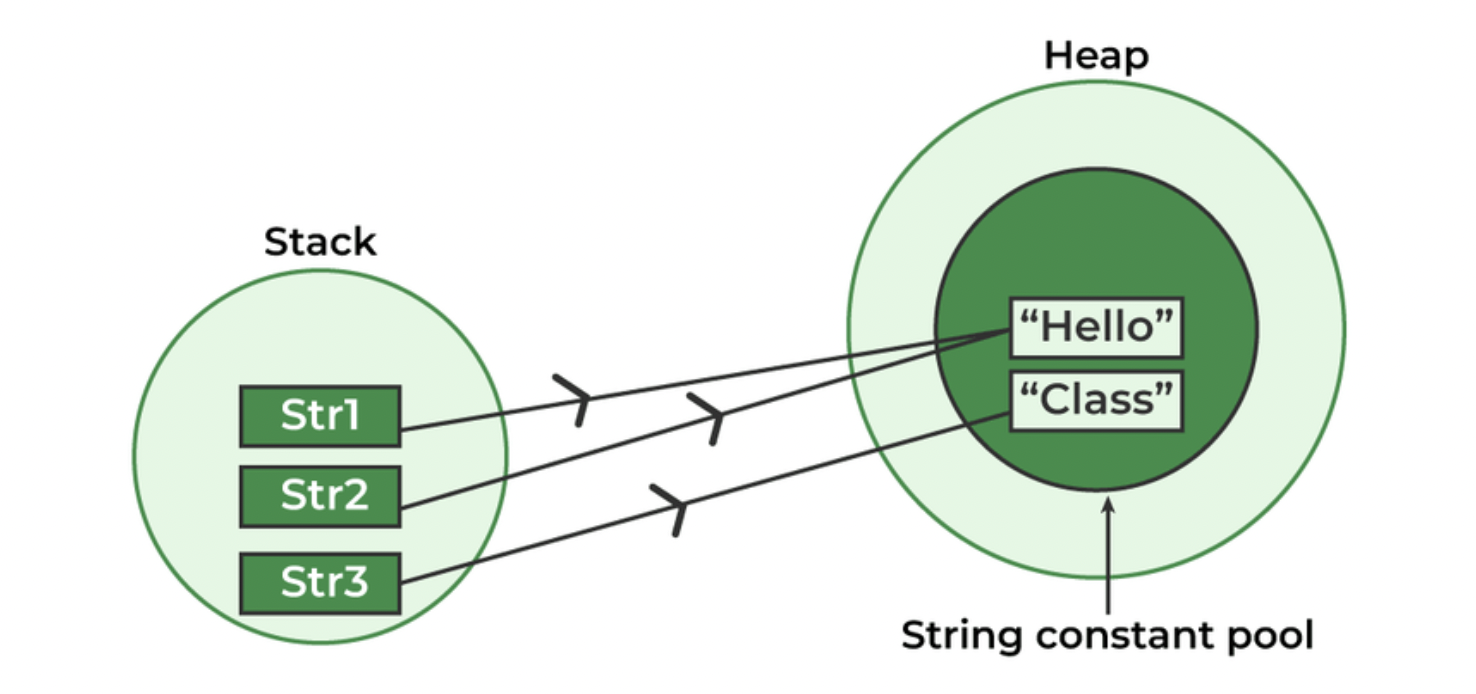


**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**String str1 = "Hello";**

**String str2 = "Hello";**

**String str3 = "Class";**



## **String Methods in Java**

### **1. int length()**

Returns the number of characters in the String.

"GoodMorning".length(); // returns 11

### **2. Char charAt(int i)**

Returns the character at ith index.

"GoodMorning".charAt(3); // returns ‘d’

### **3. String substring (int i)**

Return the substring from the ith index character to end.

"GoodMorning".substring(3); // returns “dMorning”

### **4. String substring (int i, int j)**

Returns the substring from i to j-1 index.

"GoodMorning".substring(2, 5); // returns “odM”

### **5. String concat( String str)**

Concatenates specified string to the end of this string.

String s1 = ”Good”;

String s2 = ”Morning”;

String output = s1.concat(s2); // returns “GoodMorning”

### **6. int indexOf (String s)**

Returns the index within the string of the first occurrence of the specified string.

If String s is not present in the input string then -1 is returned as the default value.

1. String s = ”Learn Share Learn”;

int output = s.indexOf(“Share”); // returns 6

2. String s = "Learn Share Learn"

int output = s.indexOf(“Play”); // return -1

### **7. int indexOf (String s, int i)**

Returns the index within the string of the first occurrence of the specified string, starting at the specified index.

String s = ”Learn Share Learn”;

int output = s.indexOf("ea",3);// returns 13

### **8. Int lastIndexOf( String s)**

Returns the index within the string of the last occurrence of the specified string.

If String s is not present in the input string then -1 is returned as the default value.

1. String s = ”Learn Share Learn”;

int output = s.lastIndexOf("a"); // returns 14

2. String s = "Learn Share Learn"

int output = s.indexOf(“Play”); // return -1

### **9. boolean equals( Object otherObj)**

Compare this string to the specified object.

Boolean out = “Morning”.equals(“Morning”); // returns true

Boolean out = “Morning”.equals(“morning”); // returns false

### **10. boolean equalsIgnoreCase (String anotherString)**

Compares string to another string, ignoring case considerations.

Boolean out= “Morning”.equalsIgnoreCase(“Morning”); // returns true

Boolean out = “Morning”.equalsIgnoreCase(“morning”); // returns true

### **11. String toLowerCase()**

Converts all the characters in the String to lowercase.

String word1 = “HeLLo”;

String word3 = word1.toLowerCase(); // returns “hello"

### **12. String toUpperCase()**

Converts all the characters in the String to uppercase.

String word1 = “HeLLo”;

String word2 = word1.toUpperCase(); // returns “HELLO”

### **13. String trim()**

Returns the copy of the String, by removing whitespaces at both ends. It does not affect whitespaces in the middle.

String word1 = “ Learn Share Learn “;

String word2 = word1.trim(); // returns “Learn Share Learn”

### **14. String replace (char oldChar, char newChar)**

Returns new string by replacing all occurrences of *oldChar* with *newChar.*

String s1 = “RunRunRun“;

String s2 = s1.replace(‘u’ ,’a’); // returns “RanRanRan”

***Note:*** *s1 is still* RunRunRun *and s2 is* RanRanRan

### **15.** boolean contains(string) :

Returns true if string contains contains the given string

String s1="geeksforgeeks";

String s2="geeks";

s1.contains(s2) // return true

### **16.** Char[] toCharArray():

Converts this String to a new character array.

String s1="FineMorning";

char []ch=s1.toCharArray(); // returns [ 'F', 'i' , 'n' , 'e' , 'M' , 'o', 'r', 'n' , 'i' , 'n' , 'g' ]

### **17.** boolean starsWith(string):

Return true if the string starts with this prefix.

String s1="geeksforgeeks";

String s2="geeks";

s1.startsWith(s2) // return true

String Equality:

In Java, you should always use the equals method to compare strings for equality, rather than the == operator. Here's an example:

String str1 = "Hello";

String str2 = new String("Hello");

boolean equal = str1.equals(str2); // true

In this example, str1 and str2 have the same value, but they are not the same object in memory, so == would return false.