Java String Pool

Sameera Nelson
Aug 2, 2018 · 3 min read

Image Credit

What is Java String Pool?

String Pool in Java corresponds to an allocation of memory in Java heap memory. It consists of a collection of String objects, which are shared and reused among several String object references for same String content.

Note : This capability is gained through the immutability nature of Java String.

Why Java String Pool is useful?

The Java String Pool avoids creating unnecessary String objects as it reuses the existing String objects. Therefore the Java String Pool has a significant impact on improving the memory and the application performance/ utilization of Java String objects.

Immutability of Strings and Java String Pool

The implementation of the Java String Pool in Java has been possible mainly because of the immutable nature of Java String objects.

Does every Java String object reside in Java String Pool?

However every String object, which is created in Java does not reside in the Java String Pool. For further clarification, it is better to have an idea on different approaches, which can be employed to create a String object in Java.

Mainly there are two different common ways, which can be used to create a new String object.

- Using String literals
- Using new keyword

Using String literals

Here is an example of creating a new String with String literals.

String literalHelloWorld = "Hello World";

Whenever this approach is used to create a new String, following steps will be followed.

First it will look for a String object with similar content in Java String Pool. If there is an already created String object in the Java String Pool equal to the String content of the new String, then the already existing String object in the Java String Pool will be reused.

The reference to the already existing String object in the Java String Pool will be returned as the newly created String object, rather than creating a new String object.

Otherwise a new String object will be created with the required String content and it will be added to the Java String Pool enabling to reuse it in future, if required.

Whenever two String objects created with this approach compared with equal to operator (i.e. ==), it will return true, as both of them are pointing to the same String object in the Java String Pool.

Using new keyword

Here is an example of creating a new String using new keyword.

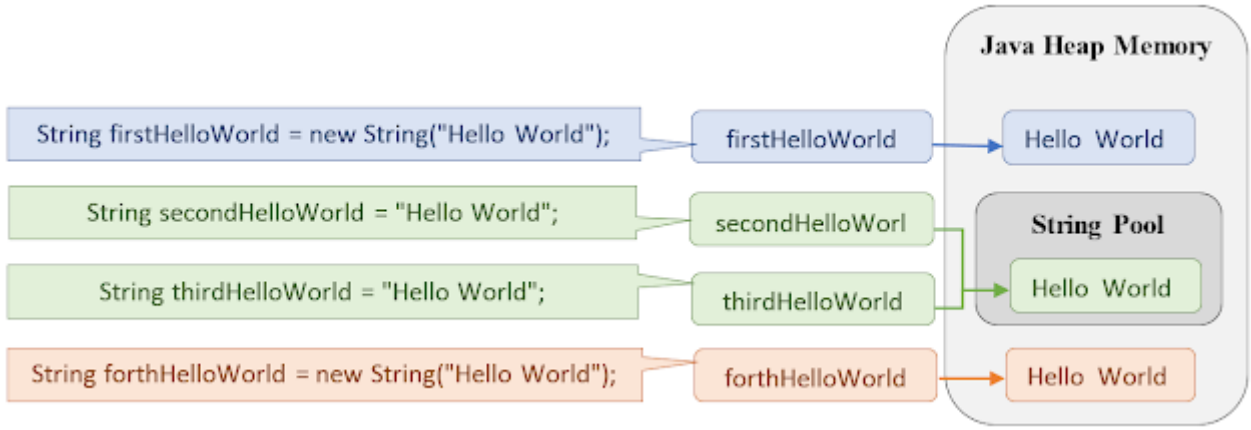
String newHelloWorld = new String("Hello World");

Every time, when this approach is used to create a String object, a new String object will be created in the **Java heap memory**, regardless of whether there are already

created String objects with same String content in the Java heap memory.

Therefore creating String objects using this approach is not recommended, as this approach is not memory efficient.

On the other hand, whenever two String objects created with this approach compared with equal to operator (i.e. ==), it will return false, as each of them point to two different String objects in Java heap memory.



firstHelloWorld == secondHelloWorld => false

firstHelloWorld == thirdHelloWorld => false

firstHelloWorld == forthHelloWorld => false

secondHelloWorld == thirdHelloWorld => true

secondHelloWorld == forthHelloWorld => false

Reference

- http://www.codedjava.com/2017/09/java-string-pool.html
- https://www.devdummy.com/2017/09/what-is-java-string-pool.html
- Joshua Bloch, "**Effective Java**", Addison-Wesley, 2008
- https://docs.oracle.com/javase/7/docs/api/java/lang/String.html

Java String