|  |
| --- |
| String Pool in Java  In Java, [**String**](https://www.javatpoint.com/java-string) is the most important topic. There is a number of concepts related to String but the **string pooling concept** is one of them. The **String pooling concept in Java** is a bit tricky. So, in this section, we are going to discuss the **String Pool** or **String Intern concept**. What is the string pool? **String pool** is nothing but a storage area in [Java heap](https://www.javatpoint.com/java-heap) where string literals stores. It is also known as **String Intern Pool** or **String Constant Pool**. It is just like object allocation. By default, it is empty and privately maintained by the [**Java String**](https://www.javatpoint.com/java-string) class. Whenever we create a string the string object occupies some space in the heap memory. Creating a number of strings may increase the cost and memory too which may reduce the performance also.  The JVM performs some steps during the initialization of string literals that increase the performance and decrease the memory load. To decrease the number of String objects created in the JVM the String class keeps a pool of strings.  When we create a string literal, the JVM first check that literal in the String pool. If the literal is already present in the pool, it returns a reference to the pooled instance. If the literal is not present in the pool, a new String object takes place in the String pool. Creating String in Java There are two ways to create a string in Java: Using String Literal String str1 = "Python";  String str2 = "Data Science";  String str3 = "Python"; Using new Keyword In Java, a [new keyword](https://www.javatpoint.com/new-keyword-in-java) is also used to create String, as follows:  String str1 = new String ("Java");  String str2 = new String ("C++");  String str3 = new String ("Data Science");  Let's understand what is the difference between them. Let's compare the string literals' references.  s1==s3 //true  s2==s3 //false  Let's see how we found that equal or not.  String Pool in Java  First, we have created a string literal **Python** and it takes place in the pool. After that, the string **Data Science** is created, it also takes place in the pool. At last, again we have created the string **Python**. But at this time, JVM checks for the string and found that string literal is already present. Instead of taking a new instance in the String pool, it returns the reference of the pooled instance i.e. **str1**.  Similarly, when we use the new keyword to create the string literals, it also takes place in the String pool. We have created three strings literals [**Java**](https://www.javatpoint.com/java-tutorial)**,**[**C++**](https://www.javatpoint.com/cpp-tutorial)**,** and [**Data Science**](https://www.javatpoint.com/data-science). We see that the string literals [**Java** and **C++**](https://www.javatpoint.com/cpp-vs-java) are the new ones. But the **Data Science** literal is already existing in the pool. At this time, [JVM](https://www.javatpoint.com/jvm-java-virtual-machine) allocates space for the literal **Data Science** in Java Heap. Remember that all the String literals created with the new keyword take place in the Java heap, not in the String pool.  **StringPoolExample.java**   1. **public** **class** StringPoolExample 2. { 3. **public** **static** **void** main(String[] args) 4. { 5. String s1 = "Java"; 6. String s2 = "Java"; 7. String s3 = **new** String("Java"); 8. String s4 = **new** String("Java").intern(); 9. System.out.println((s1 == s2)+", String are equal."); // true 10. System.out.println((s1 == s3)+", String are not equal."); // false 11. System.out.println((s1 == s4)+", String are equal."); // true 12. } 13. }   **Output:**  String Pool in Java  In the above example, we have seen that whenever we use a **new** operator to create a string it creates a new string object in the Java heap. We can forcefully stop this feature by using the **intern()** method of the String class. Java String.intern() Method The **[String.intern()](https://www.javatpoint.com/java-string-intern)**[method](https://www.javatpoint.com/java-string-intern) puts the string in the String pool or refers to another String object from the string pool having the same value. It returns a string from the pool if the string pool already contains a string equal to the String object. It determines the string by using the **[String.equals(Object)](http://javatpoint.com/java-string-equals)** method. If the string is not already existing, the String object is added to the pool, and a reference to this String object is returned.  For any two strings say **str1** and **str2, str1.intern() = = str2.intern()** will be true if and only if the statement **str1.equals(str2)** will be true. |