In Java String is an object represented as sequence of characters. Handling "strings" of characters is a fundamental aspect of most programming languages. Implementing strings as built-in objects allows Java to provide a full complement of features that make string handling convenient. For example, Java has methods to compare two strings, search for a substring, concatenate two strings, and change the case of letters within a string. String class has 11 constructors and over 40 utility methods. String objects are immutable means once a String object has been created, you cannot change the characters that comprise that string. This is not limitation of String class but it will be very helpful in multithreaded application. You can perform all the operation on your String object only things changed is new String object will be created each time. This approach is used because fixed, immutable strings can be implemented more efficiently than changeable ones.

Creating String Object

In Java, strings are objects. Just like other objects, you can create an instance of a String with the new keyword, as follows:

Step 1

String name = new String(“Amit”);

String name = "Amit";

This line of code creates a new object of class String, and assigns it to the reference variable name. There are some vital differences between these options that we'll discuss later, but what they have in common is that they both create a new String object, with a value of "Amit", and assign it to a reference variable name. Now let's say that you want a second reference to the String object referred to by s, and we are calling concat method on name object, this will create new String object with value “Amit Himani”, while reference s2 still points to object “Amit”.

Step 2

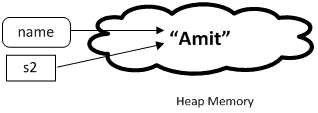
String s2 = name; // refer s2 to the same String as name

Step 3

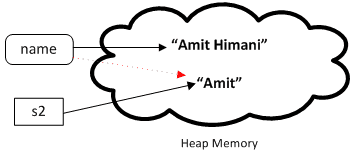
name = name.concat(" Himani"); // the concat() method 'appends' a literal to the end

Let’s understand this concept graphically,

**Step 1 and Step 2**



**Step 3**



As we know that the strings within objects of type String are unchangeable means that the contents of the String instance cannot be changed after it has been created. However, a variable declared as a String reference can be changed to point at some other String object at any time.

Comparing String Objects Reference

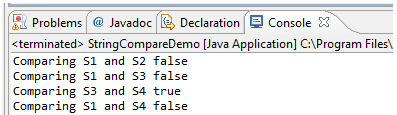
As we have seen earlier String object can be created using constructor using new keyword as well as we can use String literals. There is fundamental difference between both ways of String creation. Let’s understand this with help of Java program. We are creating two String object using literals and two objects by calling String class constructor. We are comparing String objects using == operator.

**Java Code:** [Go to the editor](http://www.w3resource.com/java-tutorial/the-string-class.php#EDITOR)

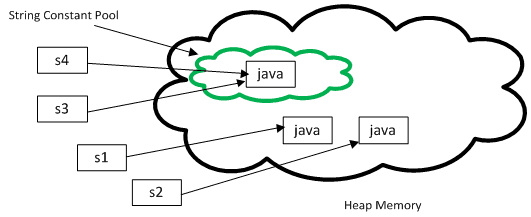
[view plaincopy to clipboardprint?](http://www.w3resource.com/java-tutorial/the-string-class.php)

1. **public** **class** StringDeepCompareDemo {
2. **public** **static** **void** main(String[] args) {
4. String s1 = **new** String("java");
5. String s2 = **new** String("java");
6. String s3 = "java";
7. String s4 = "java";
9. System.out.print("Comparing S1 and S2 ");
10. System.out.println(s1==s2);
11. System.out.print("Comparing S1 and S3 ");
12. System.out.println(s1==s3);
13. System.out.print("Comparing S3 and S4 ");
14. System.out.println(s3==s4);
15. System.out.print("Comparing S1 and S4 ");
16. System.out.println(s1==s4);
17. }
19. }

Output



As shown above, String objects created using literals are passing equality test rest all are failing. For the “literal” String assignment if the assignment value is identical to another String assignment value created then a new String object is not created. A reference to the existing String object is returned. Below picture explain this concept.



One of the key goals of any good programming language is to make efficient use of memory. As applications grow, it's very common for String literals to occupy large amounts of a program's memory, and there is often a lot of redundancy within the universe of String literals for a program. To make Java more memory efficient, the JVM sets aside a special area of memory called the "String constant pool." When the compiler encounters a String literal, it checks the pool to see if an identical String already exists. If a match is found, the reference to the new literal is directed to the existing String, and no new String literal object is created.

Comparing String Object Values

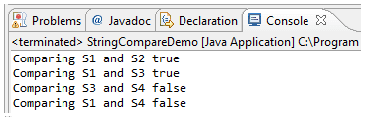
String class provides many utility methods one of them is equals method. Equals method is used to compare object’s value. There is slight difference between “==” operator and equals() method. Comparison using “==” is called shallow comparison because == returns true, if the variable reference points to the same object in memory. Comparison using equals() method is called deep comparison because it will compares attribute values. Let’s understand this concept by java program.

**Java Code:** [Go to the editor](http://www.w3resource.com/java-tutorial/the-string-class.php#EDITOR)

[view plaincopy to clipboardprint?](http://www.w3resource.com/java-tutorial/the-string-class.php)

1. **public** **class** StringCompareDemo {
2. **public** **static** **void** main(String[] args)
3. {
4. String s1 = **new** String("Hello");
5. String s2 = **new** String("Hello");
6. String s3 = "Hello"; String s4 = "Java";
7. System.out.print("Comparing S1 and S2 ");
8. System.out.println(s1.equals(s2));
9. System.out.print("Comparing S1 and S3 ");
10. System.out.println(s1.equals(s3));
11. System.out.print("Comparing S3 and S4 ");
12. System.out.println(s3.equals(s4));
13. System.out.print("Comparing S1 and S4 ");
14. System.out.println(s1.equals(s4));
15. }
16. }

Output



Summary:

In java String is an object represented by sequence of characters

A sequence of characters enclosed within double quotation marks is a literal string. You can create a String object by using the keyword new and the String constructor.

String object comparison can be done using == operator which is called shallow comparison. String object comparison using equals() method is called deep comparison.