**Creation of String**:

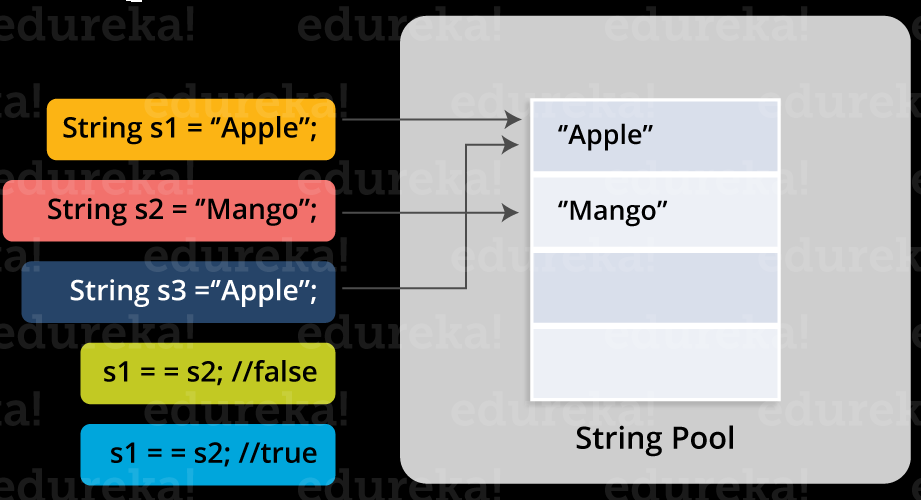
-----------------------

There are two ways to create a String object:

1. **By string literal** : Java String literal is created by using double quotes.  
   For Example: String s=“Welcome”;
2. **By new keyword** : Java String is created by using a keyword “new”.  
   For example: String s=new String(“Welcome”);    
   It creates two objects (in String pool and in heap) and one reference variable where the variable ‘s’ will refer to the object in the heap.

Now, let us understand the concept of Java String pool.

**Java String Pool:**Java String pool refers to collection of Strings which are stored in heap memory. In this, whenever a new object is created, String pool first checks whether the object is already present in the pool or not. If it is present, then same reference is returned to the variable else new object will be created in the String pool and the respective reference will be returned. Refer to the diagrammatic representation for better understanding:



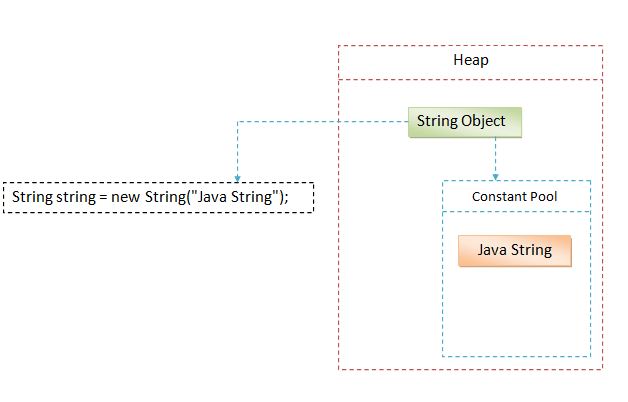
In the above image, two Strings are created using literal i.e “Apple” and “Mango”. Now, when third String is created with the value “Apple”, instead of creating a new object, the already present object reference is returned. That’s the reason Java String pool came into the picture.

Before we go ahead, One key point I would like to add that unlike other data types in Java, Strings are immutable. By immutable, we mean that Strings are constant, their values cannot be changed after they are created. Because String objects are immutable, they can be shared.

String is immutable in java programming. Once you crated an String object, you can not modify it’s content again on the same object.  
  
Let’s understand it by creating a simple String object.

String string = new String("Java String");

Let’s discuss what’s happening inside when we are creating a new string in java and it’s memory structure.

[](http://www.ashtpoint.com/wp-content/uploads/2017/07/what-is-string-in-java.jpg)

Loot at the above diagram how String object is being created in Heap area.  
  
Generally we create lot of string objects in our project or program during run time. So to make this process faster, Java though let’s create a separate area where we can create String objects faster to make our program response time faster. This separate space is called “String Constant Pool” or we can say “String Literal Pool”.  
  
**String constant pool basically stores the String literals which can be reused by other String objects.**  
  
Let’s take the above syntax to understand how String is created.

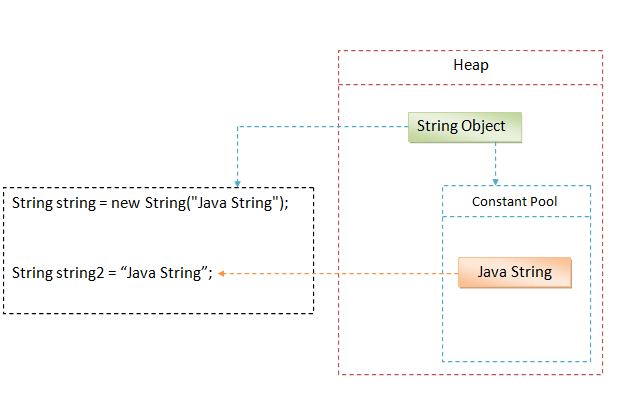
1. First, we created a reference variable of type String which is “string”. This object is empty for now.
2. Then we created a new String by simple using the “new” keyword.
3. We passed the string “Java String” to the constructor of String class.
4. Now JVM, first will look for the existing String “Java String” in constant pool. JVM finds that there no such string created before in Constant Pool.
5. So Java will crate a new String in normal Heap (Non pool) memory area and reference of it will be passed to reference variable of String type.
6. And at the same time, String literal “Java String” will be placed in the String constant pool.
7. So basically two objects will be created. First it will create a string in constant pool and then other will be created in the Heap area.
8. After that reference of the newly created String will be passed to the String reference variable.

Now let’s modify the above example and create another String object and see what’s happens now.

String string = new String("Java String");

String string2 = "Java String";

Loot at the below diagram what’s happening internally.

[](http://www.ashtpoint.com/wp-content/uploads/2017/07/String-in-java.jpg)

What’s happening here.

1. Now, JVM will first look for the String literal “Java String” in string constant pool.
2. So there will no new object will be created for the “string2”.
3. JVM will simple pass the reference of the String literal already created in the String constant pool.
4. So there will still be only two objects in memory.

If String is immutable then what’s happening in below example.

String string = "Hello ";

string = string + "World";

System.out.println(string); // Hello World

We were saying that String in immutable in Java, once crated can not be modified. But as we can see above, we have modified the string which is giving us the concatenated string.  
  
Remember, What you are seeing above is actually not happening internally. We are adding another string to the existing string. So JVM will take both of the String and will create a new String. And the reference of the newly crated String will be given to the existing String reference variable. So it will output the combined string and existing String literals will be removed from the memory.    
  
So “Hello ” will be removed from the memory and now String constant pool will contain only “Hello World”.

**How to calculate the space taken by a String in java**

Before we calculate the space a String takes, We need to first understand what actually String contains as below.

* **char array**— A character array which contains the actual characters of a given String.
* **integer offset** Offset from where String actually starts.
* **length** Length of the given string.
* **hash code**Some space for the calculation of the hash code.

From above it is clear that whether you create an empty String or not. String object will need some default space in memory.  
  
That means for if we have an empty String so the memory calculation will be done as below.

Minimum String memory usage (bytes) = 8 \* (int) (((*(no of chars)* \* 2) + 45) / 8)

1. Char array will take 4 bytes.
2. Then, 3\*4=12 bytes for the three int fields.
3. Additionally 8 bytes for the object header.
4. Memory taken 24 bytes (24 is a multiple of 8 so padding is not required).
5. Again it will take up to 12 bytes of Space (Extra 4 bytes to store the length of string).
6. Additionally 4 bytes will be added for padding to bright the memory used by the char object up to multiple of 16.
7. So an empty string uses 40 bytes of space in memory.