String class

String is different in Java as compared to c language.

In c language we use char[] to store string

char[] c = {‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘\0’};

or

char[] c = {“abcde”};

‘\0’ determines end of the string.

String is not a literal but an Object in Java.

“This is literal”

A String object is an individual instance of **java.lang.String class**.

First thing to burn in mind while working with String is that **Strings are immutable Objects.** That is once the String object is created it can never change. So a question arises, what happens if we try to change it? That is interesting theory so let us understand few basics for that.

**Strings are immutable Objects**

What is String used for? Well, we can store name of person, car, dog, etc. We can store address. In Java each character is 16-bit Unicode character. Because 16-bit is a rich and international set of characters represented in Unicode.

In Java, Strings are objects and just like other objects, you can create an instance of String using new keyword. Below line of code defines a new String and assigns it to a reference variable s.

String s = **new** String();

Now let us provide a value to String reference variable s.

s = "Java";

We can also use String class constructor

String s=**new** String("Java");

We can also use define a string in different way

String s = "Java";

Now there is difference between using String class constructor and providing a literal to String object that means

String s=**new** String("Java");

and

String s = "Java";

are different not same. We will see later why and how they are different.

**Right now we will concentrate on Immutable nature of String.**

Immutable means that once you assign a value to String its value will never change.

String s1="Java";

s1=s1.concat(" Language");

Didn’t you just said that String object is immutable. Its value cannot change? Well yes. It is.

Then why are you assigning a new value to String object s1? Because String object is immutable, its reference variable is not immutable.

Ok, Now I understood. But, doesn’t it destroy immutability as it is concatenating a different String into the previous? Nope it does not because concat method returns new String object.

A new String object, but why? Because String is immutable.

Then what happens exactly?