

Immutable String in Java

In java, **string objects are immutable**. Immutable simply means unmodifiable or unchangeable.

Once string object is created its data or state can't be changed but a new string object is created.

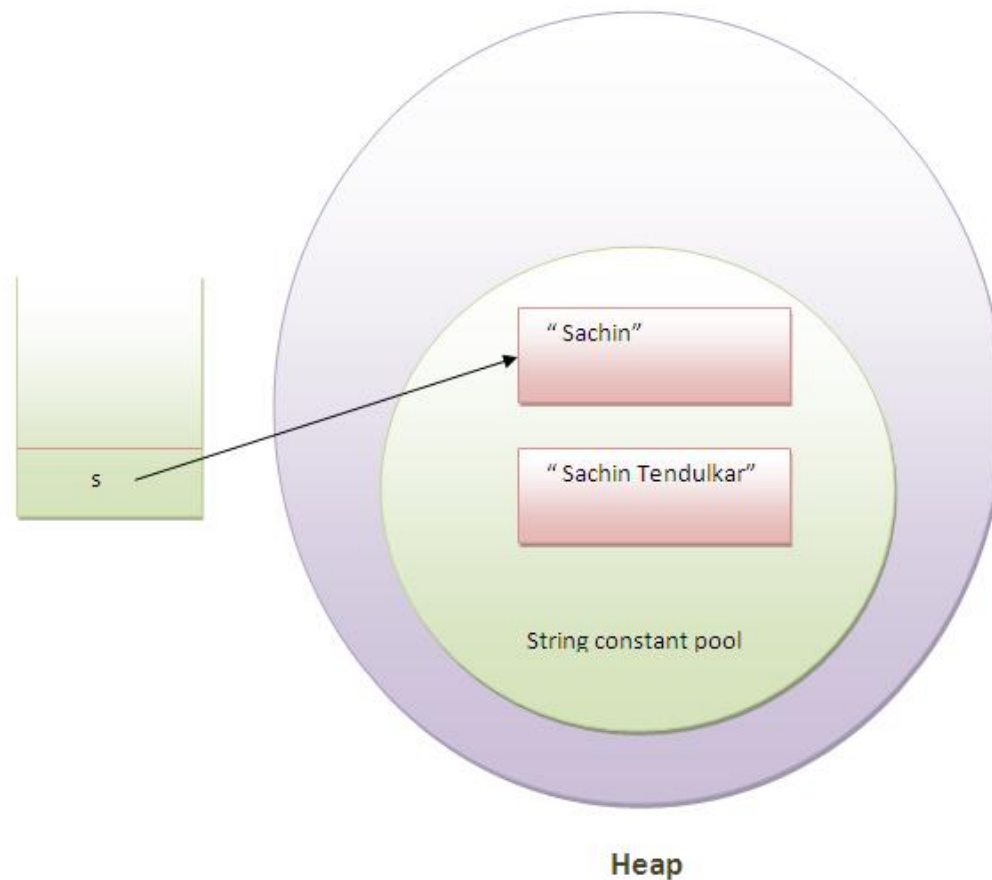
Let's try to understand the immutability concept by the example given below:

```
class Testimmutablestring{  
    public static void main(String args[]){  
        String s="Sachin";  
        s.concat(" Tendulkar");//concat() method appends the string at the end  
        System.out.println(s);//will print Sachin because strings are immutable objects  
    }  
}
```

Test it Now

Output:Sachin

Now it can be understood by the diagram given below. Here Sachin is not changed but a new object is created with sachintendulkar. That is why string is known as immutable.



As you can see in the above figure that two objects are created but `s` reference variable still refers to "Sachin" not to "Sachin Tendulkar".

But if we explicitly assign it to the reference variable, it will refer to "Sachin Tendulkar" object. For example:

```
class Testimmutablestring1{  
    public static void main(String args[]){  
        String s="Sachin";  
        s=s.concat(" Tendulkar");  
        System.out.println(s);  
    }  
}
```

Test it Now

Output:Sachin Tendulkar

In such case, `s` points to the "Sachin Tendulkar". Please notice that still `sachin` object is not modified.

Why string objects are immutable in java?

Because java uses the concept of string literal. Suppose there are 5 reference variables, all refer to one object "sachin". If one reference variable changes the value of the object, it will be affected to all the reference variables. That is why string objects are immutable in java.

[<< prev](#)[next >>](#)

Share this page



Latest 4 Tutorials

