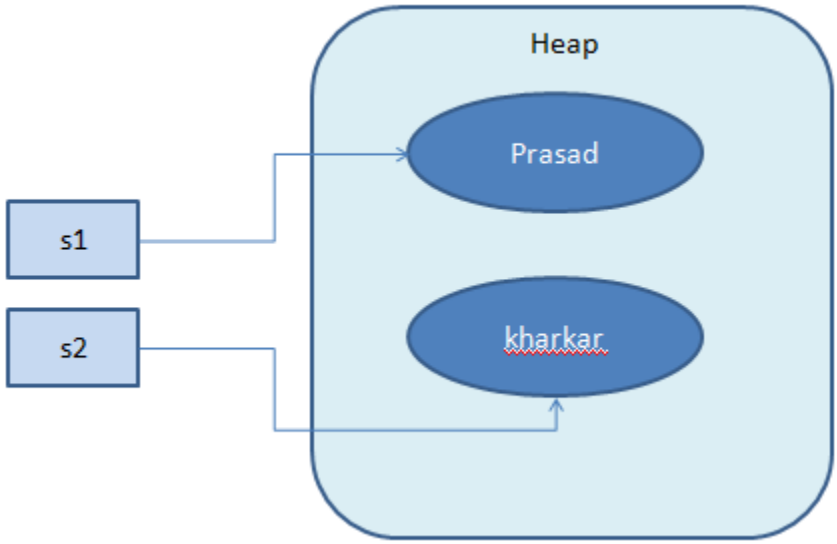
2.4.2 String immutable

Hello all, In this tutorial we are going to learn something interesting about strings in java. The String immutability
So what is it exactly? Instead of learning this part theoretically, I will write a program and you have to guess the output.

```
1  public class StringImmutability {
2
3  public static void main(String [] args)
4  {
5  String s1 = new String("prasad");
6  String s2 = new String("kharkar");
7  s1.concat(s2);
8  System.out.println(s1);
9  }
10
11 }
```

You would say, “Duh! Are you kidding man? This is a simple string concatenation program, whats the big deal? The output should be ‘`prasadkh arkar`‘”. Well are you sure? Let us run this program and the output I get by running this is `prasad`

Hmmm…What happened here? we have called the concat method on s1 right? so why doesn’t it output “`prasadkharkar`“?.
Have we done what is necessary to output the concatenated string? Let us understand what is happening diagrammatically when we are declaring and initializing variables.

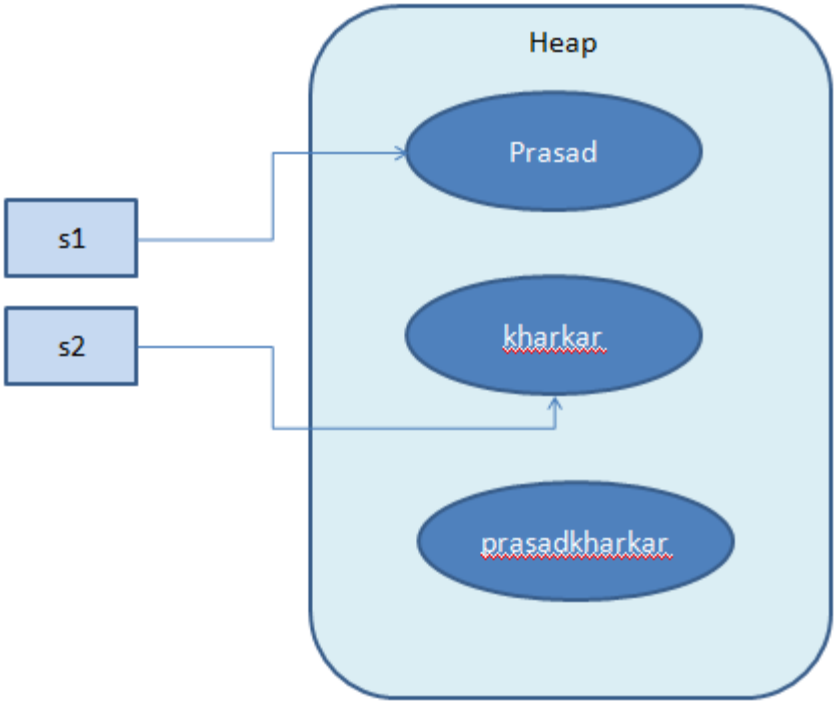


Two objects are created

A reference variable `s1` is created and the string object “`prasad`” is assigned to it.
A reference variable `s2` is created and the string object “`kharkar`” is assigned to it.
Now all the action goes when we are calling `s1.concat(s2)` .

- When you call s1.concat(s2) then a new string object will be created that is the result of concatenation, and the value of object will be `pra sadkharkar`
- Note that this does not alter the “`prasad`” object. It simply creates a new “`prasadkharkar`“.
- But after “`prasadkharkar`” is created, it is lost to our program as we are not referring to it by any variable. So the concatenation will happen but it will be lost to our program.

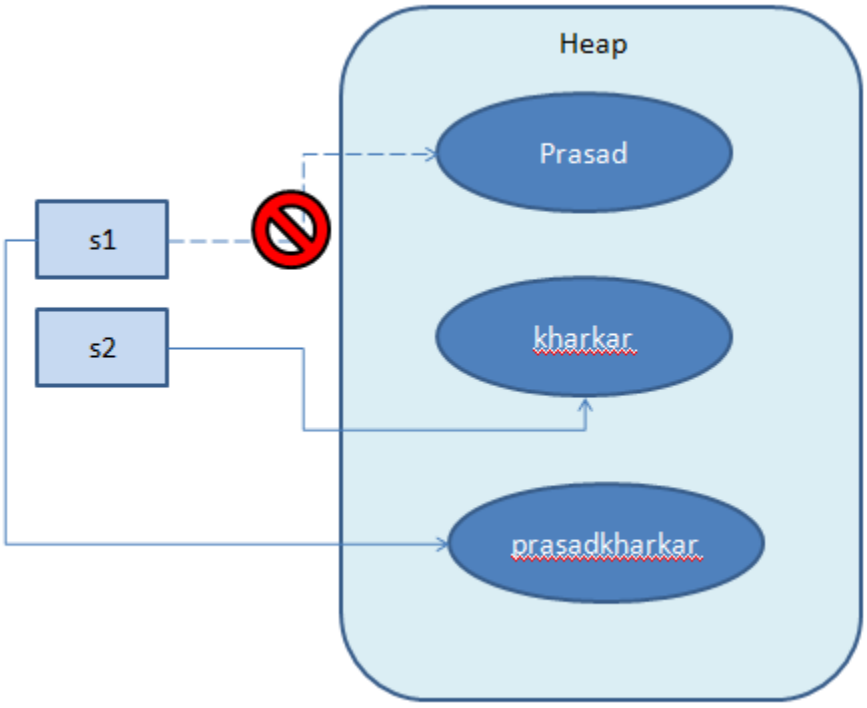
This is depicted in the image below



New String is created

Now we will change s1.concat(s2) to s1 = s1.concat(s2) . and re-run the program.
This will output `prasadkharkar`.

- s1.concat(s2) creates a new object on the heap
- s1 = s1.concat(s2) assigns the object to the reference variable s1
- Now s1 is referring to the “prasadkharkar” so its link to “prasad” is broken.



Object References Changed

In short, when you are performing some operations on string objects, then new string objects are created and the original strings remain unaffected. i.e. original string objects cannot be changed. This phenomenon is called as immutability of strings. But the

reference variables are not immutable. You can assign the same reference variable to multiple objects.

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.

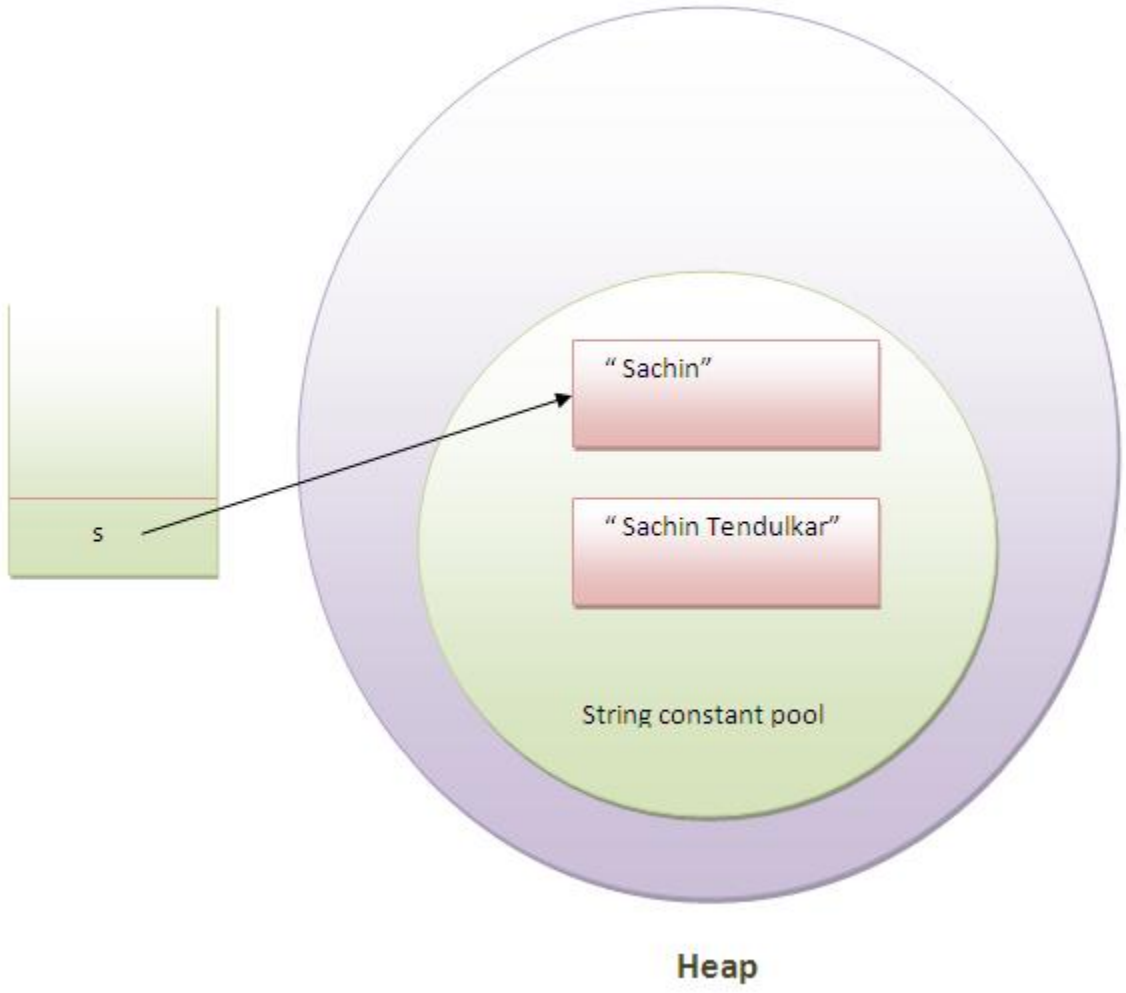
Let's try to understand the immutability concept by the example given below:

```
1.  class Testimmutablestring{
2.   public static void main(String args[]){
3.     String s="Sachin";
4.     s.concat(" Tendulkar");//concat() method appends the string at the end
5.     System.out.println(s);//will print Sachin because strings are immutable objects
6.   }
7.  }
```

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 explicitely assign it to the reference variable, it will refer to "Sachin Tendulkar" object.For example:

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

```
6.  }
7.  }
```

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 referes 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.