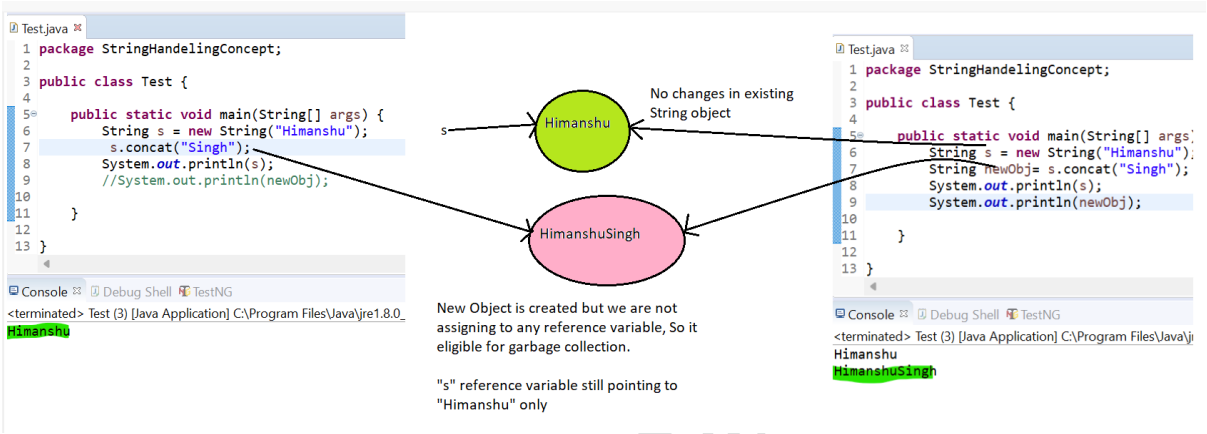
**********String Handling in JAVA**********

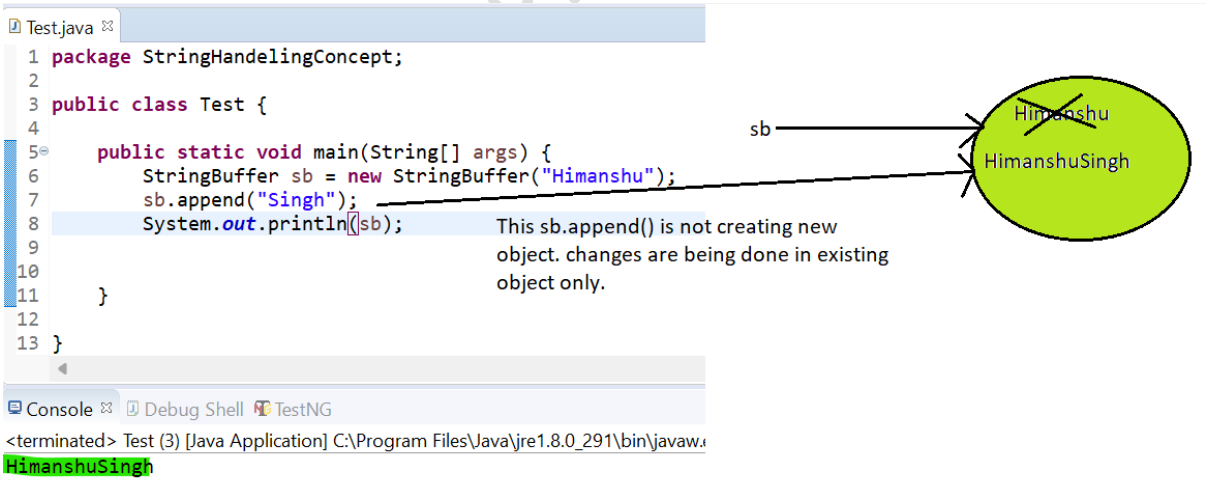
String: String objects are immutable. Once we create any object then we can't perform any changes in that object.

Immutability: Once we create a new object, we are not allowed to perform any changes in that object but if you are trying to perform any changes then with those changes a new object will be created. And in the existing object cannot perform any changes. This non changeable behaviour is nothing but immutability concept.



String Buffer: String Buffer objects are mutable. Means we can perform any changes in String buffer objects.

Mutability: Once we create any StringBuffer object then we are allowed to perform any changes in that object. This changeable behaviour is nothing but mutability concept.



Operator == And equals(): == operator is always use for reference comparison for String & StringBuffer class. And equals() is also meant for reference comparison.

Basically, equals() method is present in Object class. And in String class it is overridden method and in StringBuffer class it is not overridden. So, for String class equals() is used for content comparison and in StringBuffer class it is used for reference comparison.

Test.java

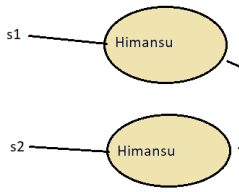
```
1  package StringHandelingConcept;
2
3  public class Test {
4
5      public static void main(String[] args) {
6          String s1 = new String("Himansu");
7          String s2 = new String("Himansu");
8          System.out.println(s1==s2);
9          System.out.println(s1.equals(s2));
10
11     }
12
13 }
```

Console | Debug Shell | TestNG

<terminated> Test (3) [Java Application] C:\Program Files\Java\jre1.8.0_

false
true

So, s1 & s2 both reference variables are pointing to the different objects. returning False.
equals() is overriden method of Object class. And here it is referring to String Objects and will do Content comparison. So, returning true.



Test.java

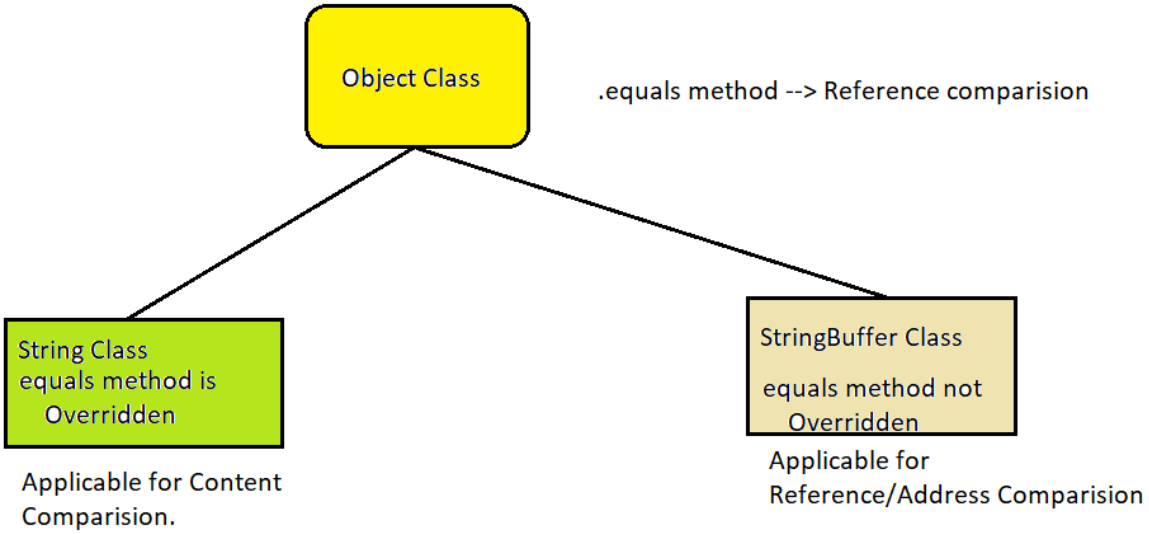
```
1  package StringHandelingConcept;
2
3  public class Test {
4
5      public static void main(String[] args) {
6          StringBuffer s1 = new StringBuffer("Himansu");
7          StringBuffer s2 = new StringBuffer("Himansu");
8          System.out.println(s1==s2);
9          System.out.println(s1.equals(s2));
10
11     }
12
13 }
```

Console | Debug Shell | TestNG

<terminated> Test (3) [Java Application] C:\Program Files\Java\jre1.8.0_291\bin\jav

false
false

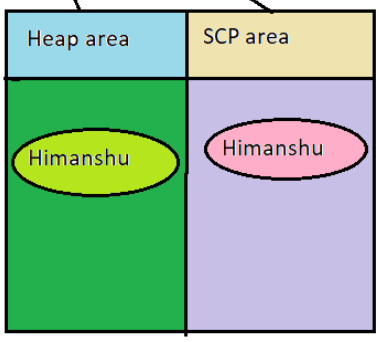
Always reference comparison.
equals() is not overriden method. So, will be reffering to reference comparison. So, returning false because reference variables/address is different.



 # Heap & String Constant Pool (SCP):

String s = new String("Himanshu")

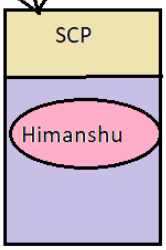
There are two objects will be created in above case. and s is pointing to the object created in Heap area. and reference variable is maintained by JVM internally for the object created in SCP area. So it is not eligible for garbage collection. it is for future purpose.



Heap & SCP Object creation.

String s = "Himanshu"

In this case only one object will be created in SCP area.

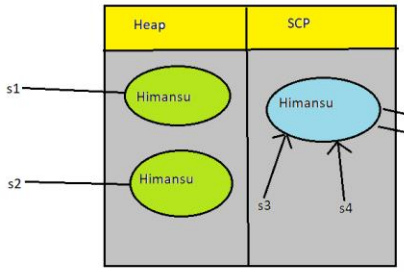


Note: Whenever we are using 'new' keyword then it is compulsory to create an object in Heap area. So, there may be a chance of two objects with the same content in heap area. And a copy will be maintained in SCP area with that content.

And in SCP area there is no chance of existence of two objects with the same content. The same copy will be used again and again.

Case: In the screenshot below, how many objects will be created?

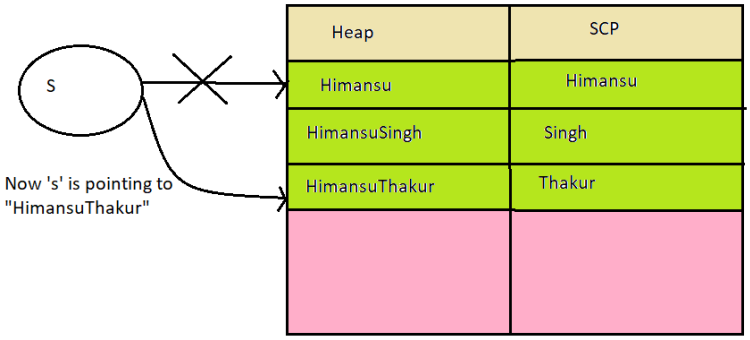
```java
public class Test {

    public static void main(String[] args) {
        String s1 = new String("Himansu");
        String s2 = new String("Himansu");
        String s3 = "Himansu";
        String s4 = "Himansu";

    }
}
```




```java
public class Test {

    public static void main(String[] args) {
        String s1 = new String("Himansu");
        String s2 = new String("Himansu");
        String s3 = "Himansu";
        String s4 = "Himansu";
        System.out.println(s3==s4);
    }
```

```
true  It is the proof because == operator always
      applicable for reference comparison.
```

Case: How many objects will be created?

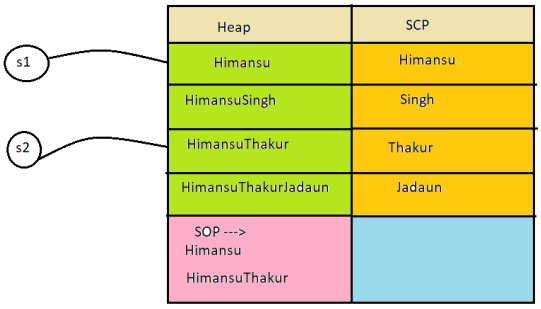



```
String s = new String("Himansu");
s.concat("Singh");
s = s.concat("Thakur");
```

Qus:- How many objects will be created ?

Case: How many objects will be created and where?

```
String s1 = new String("Himansu");
s1.concat("Singh");
String s2 = s1.concat("Thakur");
s2.concat("Jaduan");
SOP(s1);
SOP(s2);
```




```java
package StringHandelingConcept;

public class Cases {

    public static void main(String[] args) {
        String s1 = new String("Himansu");
        s1.concat("Singh");
        String s2 = s1.concat("Thakur");
        s2.concat("Jaduan");
        System.out.println(s1);
        System.out.println(s2);

    }
}
```

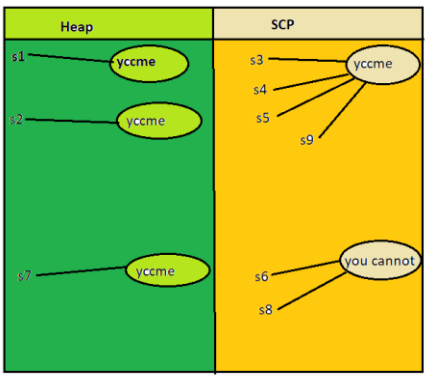
```
<terminated> Cases [Java Application] C:\Program Files\Java\jre1.8.0
Himansu
HimansuThakur
```

Case: How many objects will be created and where? (Proof)

If both are constants then operation will be performed by JVM at Compile time. And Object will be created at **SCP** area.
If there is atleast one variable then operation will be performed by at Run time. And Object will be created at **Heap** area.

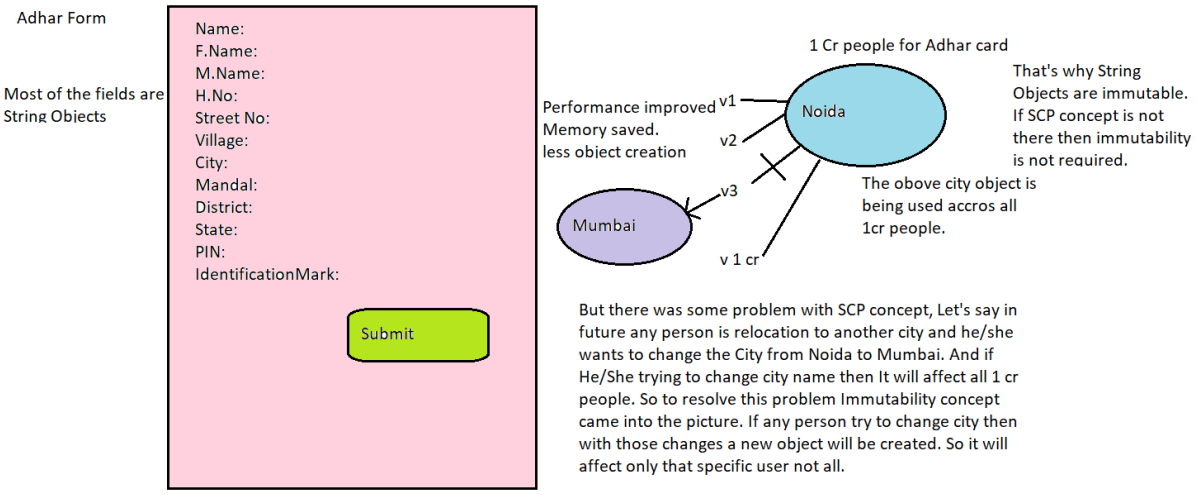
```java
package StringHandelingConcept;

public class Cases {

    public static void main(String[] args) {
        String s1 = new String("You cannot change me");
        String s2 = new String("You cannot change me");
        System.out.println(s1==s2);
        String s3 = "You cannot change me";
        System.out.println(s1==s3);
        String s4 = "You cannot change me";
        System.out.println(s3==s4);
        String s5 = "You cannot" + " change me";
        System.out.println(s4==s5);
        String s6 = "You cannot";
        String s7 = s6 + " change me";
        System.out.println(s4==s7);
        final String s8 = "You cannot";
        String s9 = s8 + " change me";
        System.out.println(s4==s9);

    }
```

```
<terminated> Cases [Java Application] C:\Program Files\Java\jre1.8.0_291\bin\javaw
false
false
true
true
false
true
```



This is the proof which shows objects creation & comparision with respect to Heap & SCP area.

Advantage/Importance of SCP:



StringBuilder : Non-Synchronized version of String Buffer is known as String Builder. In String Buffer all the methods are synchronized. Means only one thread is allow at a time. And in String Builder there is no such type of restriction. Multiple threads are allowed at a time.