String Immutable:

String Objects are immutable once we create modification is not possible

Every time we are creating new objects s1, s2, s3 they are stored in SCP area but orginal string is not changing

```
String s1 = new String("Java");
String s2 = s1.concat("Python");
String s3 = s2.concat("JavaScript");

System.out.println(s1); // Java
System.out.println(s2); // JavaPython
System.out.println(s3); // JavaPythonJavaScript
```

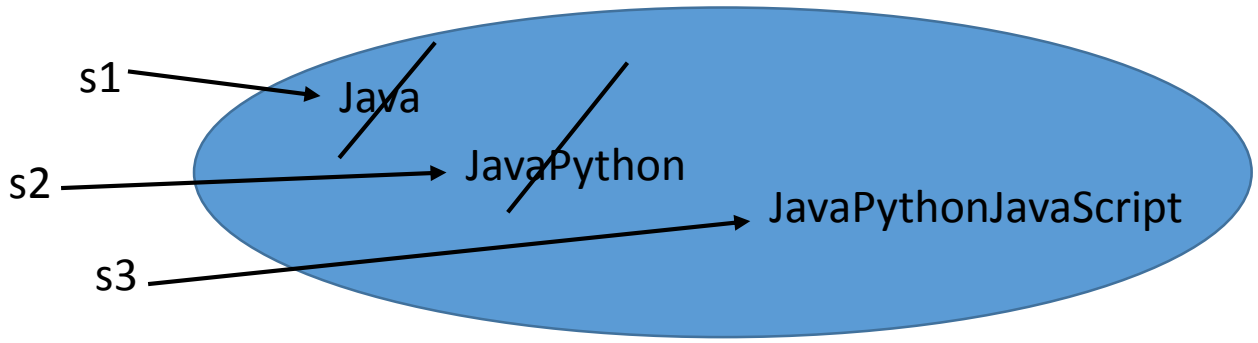
In the above string is immutable its not changing orginal object data

StringBuffer:

StringBuffer is mutable we can change the content

```
StringBuffer sb1 = new StringBuffer("Java");
StringBuffer sb2 = sb1.append("Python");
StringBuffer sb3 = sb1.append("JavaScript");
System.out.println(sb1); //JavaPythonJavaScript
System.out.println(sb2); //JavaPythonJavaScript
System.out.println(sb3); //JavaPythonJavaScript
```

In the below stringbuffer is mutable its changing orginal object data directly

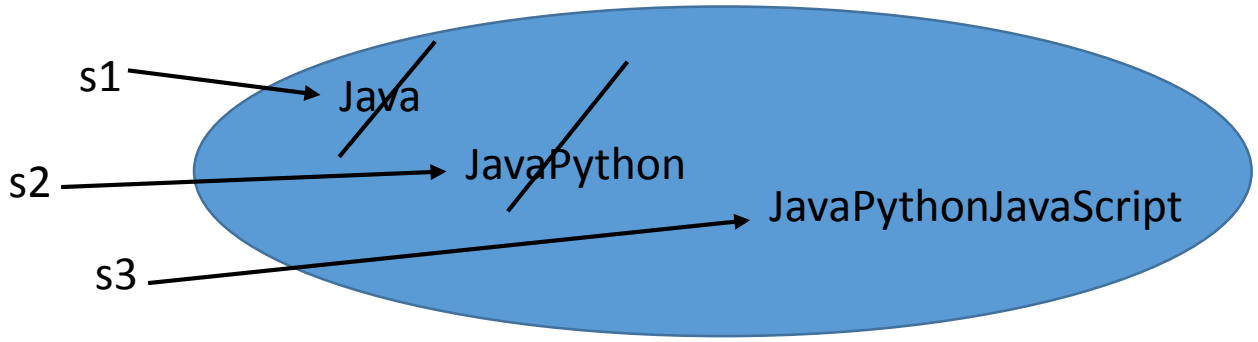


StringBuilder:

StringBuilder is mutable we can change the content

```
StringBuilder sb1 = new StringBuilder("Java");
StringBuilder sb2 = sb1.append("Python");
StringBuilder sb3 = sb1.append("JavaScript");
System.out.println(sb1); //JavaPythonJavaScript
System.out.println(sb2); //JavaPythonJavaScript
System.out.println(sb3); //JavaPythonJavaScript
```

stringbuilder is mutable its changing orginal object data directly



String	String Buffer 1.0	String Builder 1.5
String is **non-synchronized** i.e. not thread safe. It means two threads can call the methods of String simultaneously.	StringBuffer is **synchronized** i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously.	StringBuilder is **non-synchronized** i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously.
Immutable	Mutable	Mutable
Memory Allocation in String Pool Area	**Memory Allocation** in Heap Area	**Memory Allocation** in Heap Area
If we keep on changing the String object continiously **new String object is created** and allocates more memory and **performance becomes slow**	If we keep on changing the StringBuffer object continuously **no new String object is created** and consumes less memory so **performanceis high compared to String**	If we keep on changing the StringBuffer object **continuously no new String object is created** and consumes less memory so **performanceis high compared to String Buffer**
If **data is not required to modify** then use String	If **data is required to modify** then use String Buffer (Thread Safety)	If **data is required to modify** then use String Builder (No Thread Safety)