String StringBuffer StringBuilder

String:- String is used to represent a group of characters or character array enclosed within the double quotes.

String str=new String(“HELLO”);

String class objects are immutable objects, where Immutable objects mean we cannot perform any modifications over existing content, if we are trying to perform modifications over existing content then operations are allowed, but, the resultant data will not be stored original object, a new object will be created with modified data.

Example:-

```
public class Test1
{
public static void main(String[] args)
{
String str1 = new String("gla");
str1.concat("university");
System.out.println(str1);//gla
String str2="gla";
str2.concat("uni");
System.out.println(str2);//gla
}
}
```

Result:-

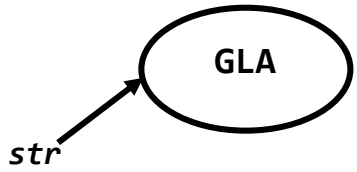
```
Gla
Gla
```

There are two ways to create String objects in the java.

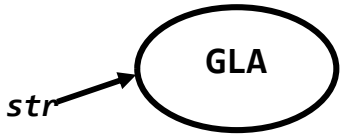
1) By using a new operator String str = new String(“GLA”);

2) Without using the new operator String str=”GLA”;

In the first way ***String str = new String(“GLA”);*** *Two* objects will be Created, One is in the Heap and the Other is in String Constant Pool (SCP) area and str is always pointing to Heap Object.

HEAP AREA	*SCP AREA*
	

In the second way String ***str="GLA";*** *One object will be created in SCP are and str* always refers to that Object.

HEAP AREA	*SCP AREA*
	

Note:-Whenever we are used New Operator to create a String object, then a new object will create in the Compulsory heap area.

There may be a Chance of existing two Objects with the Same Content on the Heap memory but No Chance of existing two Objects with the Same Content in the SCP memory i.e. Duplicate Objects are Possible in Heap memory but there is No Chance of Duplicate Objects existing in SCP memory.

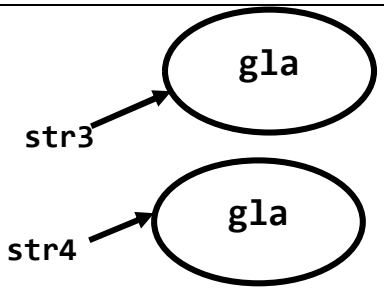
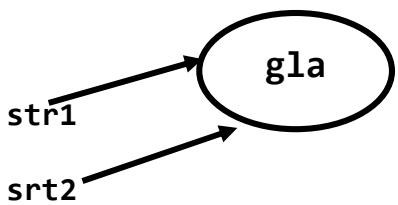
Example:-1

```
class Test
{
public static void main(String[]args)
{
String str1="gla";
String str2="gla";
String str3=new String("gla");
String str4=new String("gla");
System.out.println(str1==str2);
System.out.println(str2==str3);
System.out.println(str3==str4);
}
}
```

Result:-

```
true
false
false
```

Memory Structure of the above program:-

HEAP AREA	*SCP AREA*
	

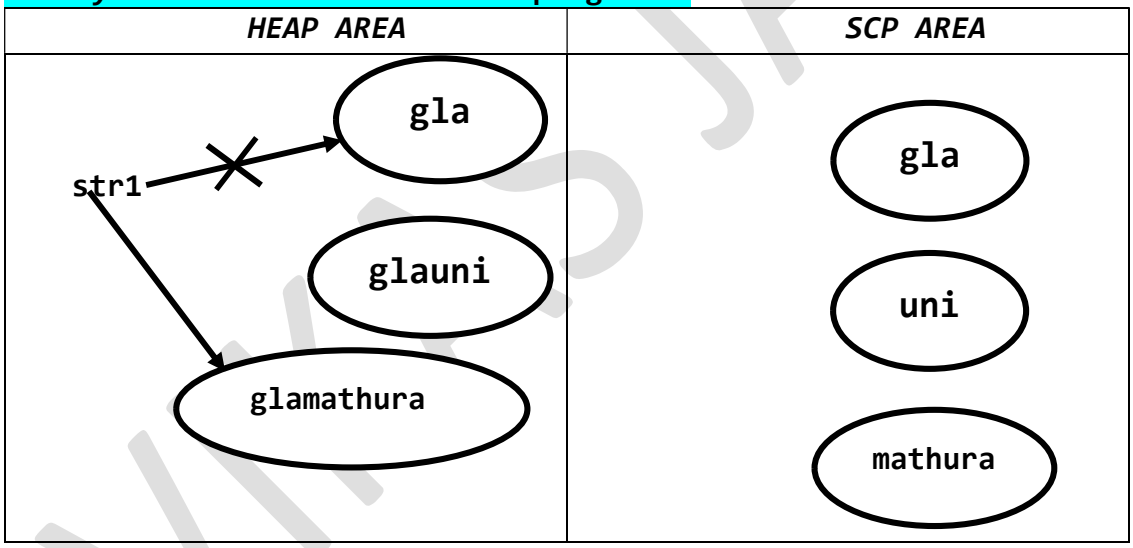
Example:-2 For every String Constant one object will be created in SCP. Because of run time operation if an object is required to create compulsory that object should be placed on the heap but not SCP.

```
class Test
{
public static void main(String[]args)
{
String str1=new String("gla");
System.out.println(str1);
str1.concat("uni");
System.out.println(str1);
str1=str1.concat("mathura");
System.out.println(str1);
}
}
```

Result:-

```
gla
gla
glamathura
```

Memory Structure of the above program:-



String Class Constructors:

1) String str = new String(); Creates an Empty String Object.

2) String str = new String(String literal); Creates an Equivalent String Object for the given String Literal.

Example:-

String str = new String("VIKAS");

3) String str = new String(StringBuffer sb); Creates an Equivalent String Object for the given StringBuffer.

Example:-

```
StringBuffer sb=new StringBuffer("VIKAS");
String str = new String(sb);
```

4) String str = new String(char [] ch); Creates an Equivalent String Object for the given char array.

Example:-

```
char[] ch = {'a', 'b', 'c', 'd'};
String str = new String(ch);
System.out.println(str); //abcd
```

5) String s = new String(byte[] b); Creates an Equivalent String Object for the given byte array.

Example:-

```
byte[] b = {100, 101, 102, 103, 104};
String str = new String(b);
System.out.println(str); //defgh
```

A most important method of String class:-

1. public char charAt(int index):-Returns the character located at the specified index.
2. public String concat(String str):-we can use this method to add a new string value.
3. public boolean equals(Object o):-we can use this method to check two string objects contain the same content or not.
4. public boolean equalsIgnoreCase(String s):- we can use this method to check whether two string objects contain the same content or not where the case is not important.
5. public String substring(int begin):-This method returns the substring from the beginning index to the end of the string.
6. public String substring(int begin, int end):-This method returns the substring from beginning index to the end-1 index.
7. public int length():-This method returns the number of characters present in the string.
8. public String replace(char old, char new):-By using this method we replace every old character with a new character.

9. public String toLowerCase():-By using this method we converts the all characters of the string to lowercase.

10. public String toUpperCase():-By using this method we converts the all characters of the string to uppercase.

11. public String trim():-We can use this method to remove blank spaces present at beginning and end of the string but not blank spaces present at middle of the String.

12. public int indexOf(char ch):-This method returns index of 1st occurrence of the specified character if the specified character is not available then return -1.

13. public int lastIndexOf(Char ch):-This method returns index of last occurrence of the specified character if the specified character is not available then return -1.

StringBuffer:- StringBuffer we also used to represent a group of characters or character array enclosed within the double quotes.

StringBuffer str=new StringBuffer(“HELLO”);

StringBuffer class objects are mutable objects, where mutable objects mean we can perform any modifications over existing content.

If the content will change frequently then it is not recommended to use a String object because for every change a new object will be created internally overcome this problem use StringBuffer because in StringBuffer all required changes will be performed in the existing object only instead of creating a new object.

Example:-

```
public class Test1
{
public static void main(String[] args)
{
StringBuffer sb1 = new StringBuffer("gla");
sb1.append(" university");
System.out.println(sb1);
}
}
```

Result:- gla university

StringBuffer class constructor:-

1. StringBuffer sb=new StringBuffer():-Creates an empty StringBuffer object with default initialcapacity "16".

Once StringBuffer object reaches its maximum capacity a new StringBuffer object will be created with

Newcapacity=(currentcapacity+1)*2.

StringBuffer sb=new StringBuffer();

2. StringBuffer sb=new StringBuffer(int initialcapacity):-Creates an empty StringBuffer object with the specified initial capacity.

StringBuffer sb=new StringBuffer(19);

3. StringBuffer sb=new StringBuffer(String s):-Creates StringBuffer object with given String.

StringBuffer sb=new StringBuffer("ashok");

Most important method of StringBuffer:-

1. public int length():-This method return the no of characters present in the StringBuffer.
2. public int capacity():-This method returns the total no of characters that can hold StringBuffer object.
3. public char charAt(int index):-This method returns the character located at a specified index.
4. public void setCharAt(int index, char ch):-This method replace the character located at the specified index with the provided character.
5. public StringBuffer insert(int index,String str):-To insert at the specified location.
6. public StringBuffer append(String str):-By using this append string value.
7. public StringBuffer reverse();
8. public void trimToSize():-To deallocate the extra allocated free memory such that capacity and size are equal.

9. public void ensureCapacity(int initialcapacity):-To increase the capacity dynamically(fly) based on our requirement.

StringBuilder:-

Every method present in StringBuffer is synchronized hence at a time only one thread is allowed to operate on the StringBuffer object, this creates performance problems, to overcome this problem we use StringBuilder.

StringBuffer allow allow multithreading.

StringBuffer Vs StringBuilder:-

StringBuilder is exactly the same as StringBuffer (including constructors and methods) except for the following differences :

StringBuffer	StringBuilder
1. Every method present in StringBuffer is synchronized.	1.No method present in StringBuilder is synchronized.
2. At a time only one thread is allowed to operate on the StringBuffer object hence StringBuffer object is the Thread safe.	2.At a time Multiple Threads are allowed to operate simultaneously on the StringBuilder object hence StringBuilder is not Thread safe.
3. It increases the waiting time of the Thread and hence relative performance is low.	3. Threads are not required to wait and hence relatively performance is high
4. Introduced in 1.0 version.	4. Introduced in 1.5 versions

Note:-1 If the content is fixed and won't change frequently then we should go for String.

Note:-2 If the content will change frequently but Thread safety is required then we should go for StringBuffer.

Note:-3 If the content will change frequently and Thread safety is not required then we should go for StringBuilder.