1.What is a String?

- Generally, in c,c++ string is a sequence of characters.
- But in java,string is an Object that is used to represent the sequence of characters.

  Eg: "JamesGosling ",here it is a sequence of 12 characters represent in java

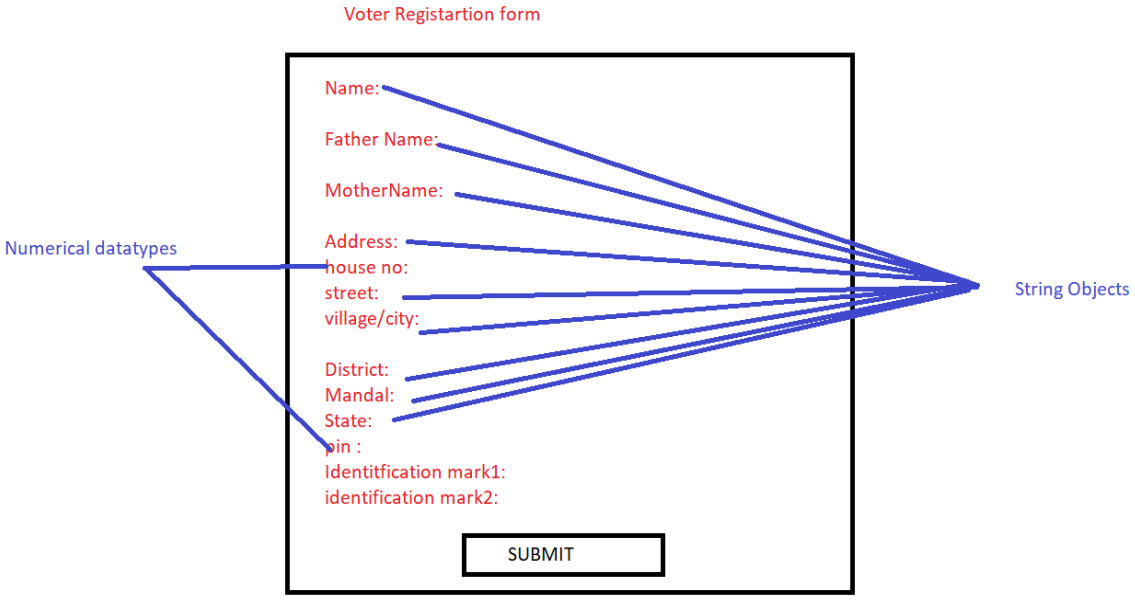
- String class is used to represent string Object.
- String class is immutable class that is present in java.lang package;
- So,here about immutable class we will discuss later.
- Basically, java memory is divided into 3 parts

  1.stack

  2.heap

  3.string pool

- So,In java String is so important in java to develop in any application there is maximum need of string .



2.String class declaration in java

The declaration of String class

public final class String

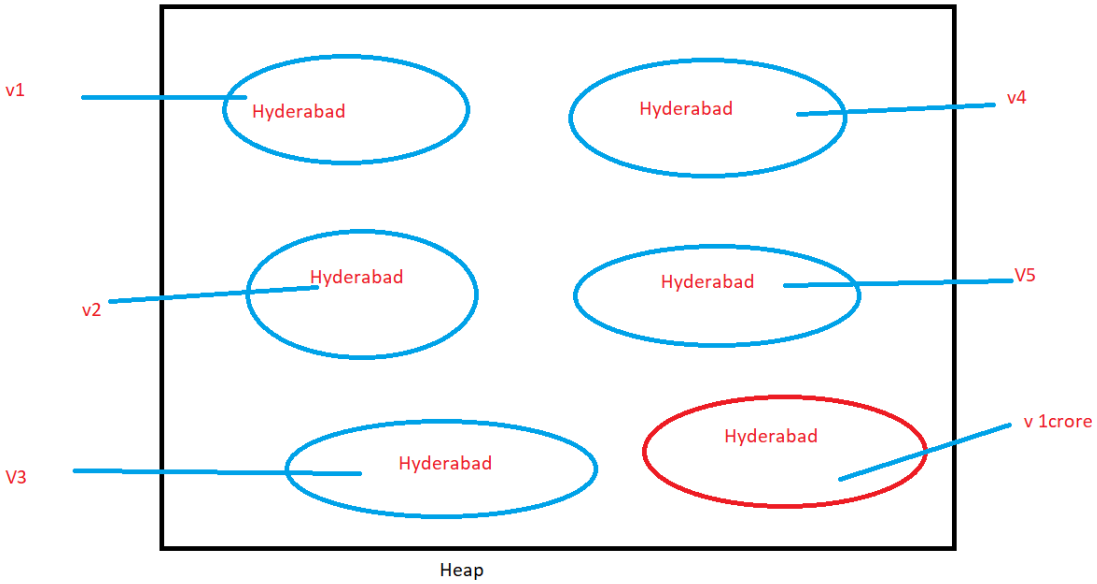
extends Object

implements Serializable, Comparable<String>, CharSequence

- So,in java every class extends object class
- So,in java Object is super class of all classes in java
- The above syntax says that string is a final class and object class is its superclass. The class implements three interfaces Serializable, Comparable, and CharSequence.
- The CharSequence is an interface that is used to represent the sequence of characters. String, StringBuffer, and StringBuilder class implement it.
- So,In java Strings comes under referenced datatype or non-primitive data types.
- In java any class is called referenced datatype because it uses reference variable to store data.

3.What is the need of string Constant pool area?

- If suppose we had one application that is voter registration form
- If suppose there are 1 crore voters in Hyderabad



- Observe the above diagram..so, in heap area every time creating new object with same string that is upto 1 crore times.
- So, every time creating object in heap for hyderabad then automatically the performance will become less.
- So, In java by using StringConstantpoolarea concept in string class
- It is possible to create one object of hyderabad and that object will be share among all these 1 crore members

- In Stringconstantpool when we declare a string Object then it will check whether that Object is there are not first
- If the object is not there then it will create that string Object .
- If the object is there then it will add reference to the existing Object.

4.How to create a String Object

There are three ways to create a string Object in java

1.By using literal way

2.By using new Keyword

3.By converting character arrays into Strings.

1.By using literal way:

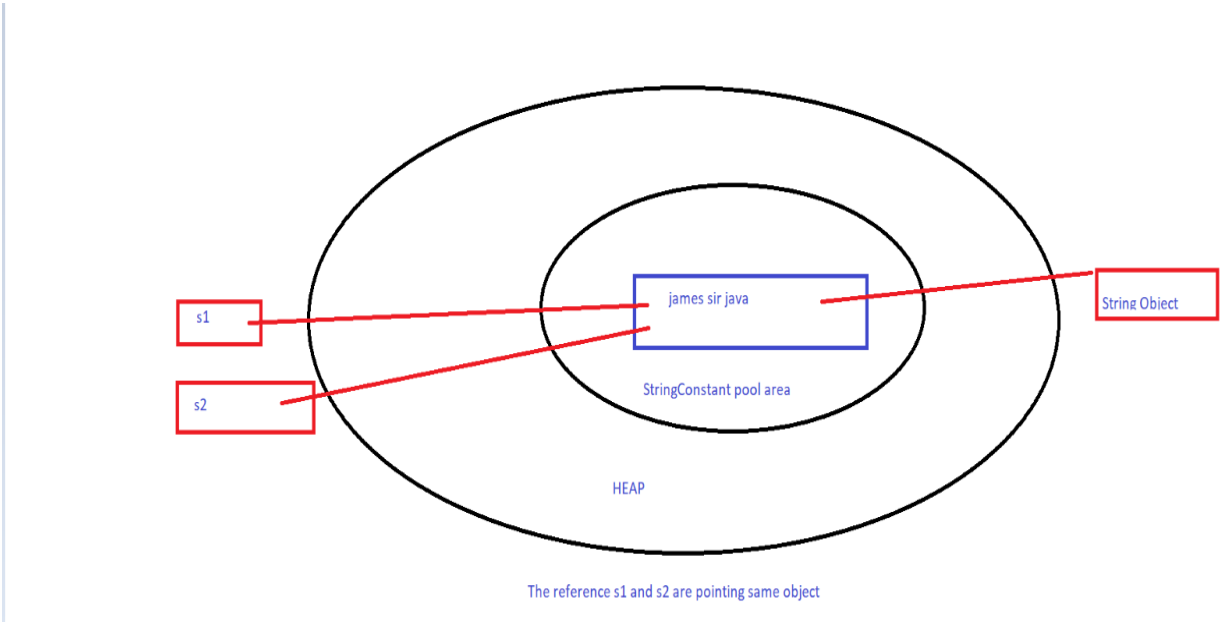
- String literal in Java is created by using double quotes.
- For example:
- String s = "Hello";
- The string literal is always created in the string constant pool. In Java, String constant pool is a special area that is used for storing string objects.
- Whenever we create a string literal in Java, JVM checks string constant pool first. If the string already exists in string constant pool, no new string object will be created in the string pool by JVM.
- JVM uses the same string object by pointing a reference to it to save memory. But if string does not exist in the string pool, JVM creates a new string object and placed in the pool.

For example:

1.1.Two String Objects having same content

String s1="James sir java";

String s2="james sir java";



The reference s1 and s2 are pointing same object

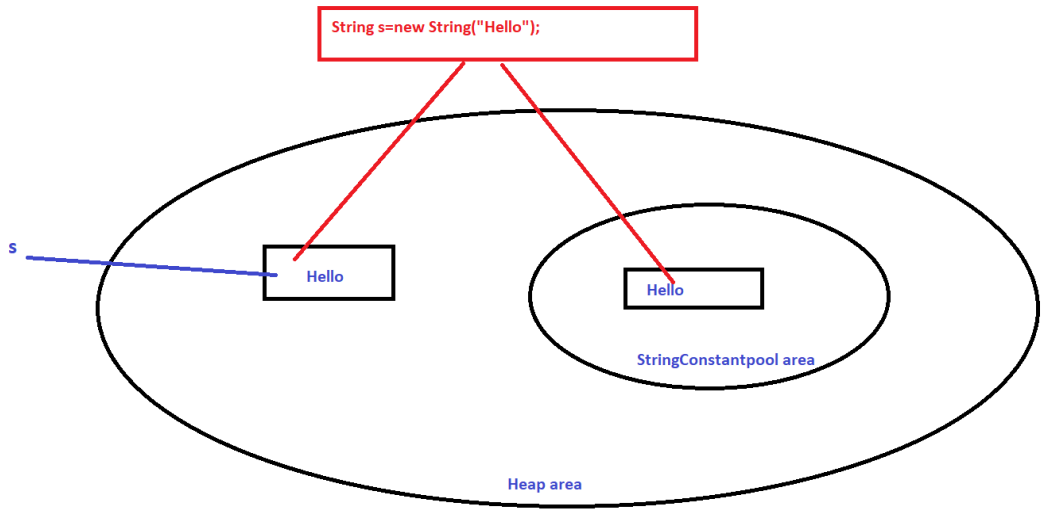
1.2.Two string Objects having Different content:

String s1="james Gosling";

String s2="Java";

2.By using new Keyword

- This is the second way of creating a string.
- It is just like creating an object of any class. It can be declared as follows:
- String s=new String("Hello");
- Whenever we will create an object of string class using the new operator, JVM will create two objects. First, it will create an object in the heap area and store string "Hello" into the object.
- After storing data into memory, JVM will point a reference variable s to that object in the heap. Allocating memory for string objects is shown in the below .



- In above diagram in heap area one hello object is created with reference s.
- In StringConstantPool area hello object is created but with no explicit reference.

- It means two copies of Objects are created by using new keyword and In StringConstantPool area an implicit reference is created by internally by JVM.
- The object created in StringConstant pool is not eligible for Garbage Collection.

3.By converting character arrays into Strings.

- The third way to create strings is by converting the character arrays into string. Let's take a character type array: arr[ ] with some characters as given below:
- char arr[ ] = {'j','a','v','a'};
- Now create a string object by passing array name to string constructor like this:
- String s = **new** String(arr);

Program 1:

```
package com.strings;


public class StringCreation {


public static void main (String [] args) {

// TODO Auto-generated method stub

char arr[]= {'j','a','v','a'};

String s=new String(arr);

System.out.println(s);

}


}
```

Output:

java

Program 2:

If you do not want all the characters of the array into string then you can also mention which character you need, like this:

1

String s = **new** String(arr, 1,3);

```
package com.strings;


public class StringCreation {


public static void main (String [] args) {

// TODO Auto-generated method stub

char arr[]= {'j','a','v','a'};

String s=new String(arr,1,3);

System.out.println(s);

}


}
```

Output:

ava

Notepoint:

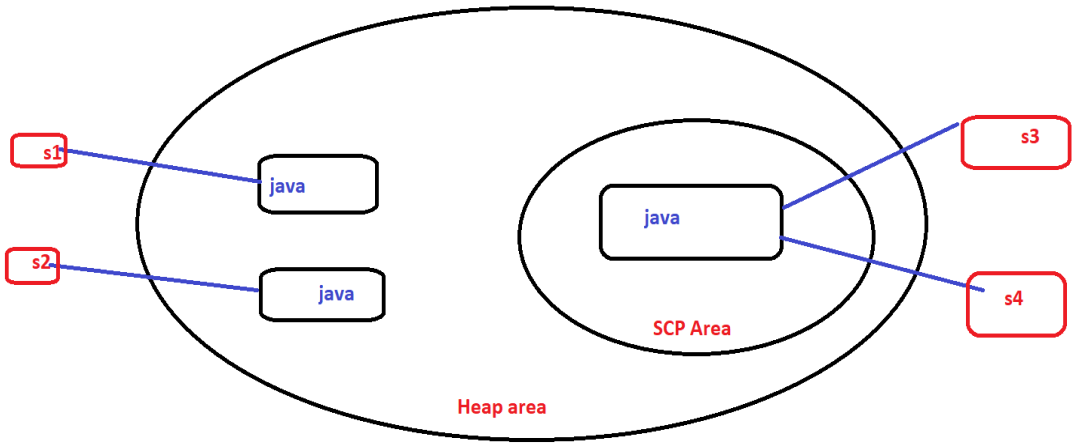
So, the counting will start from 0 i.e 0th character in the array is 'j' and the first character is 'a'. Starting from 'a', total of three characters 'aja' will copy into string s.

How many total objects will be created in memory for following string objects?

Program:

```
package com.strings;

public class StringMemory {

public static void main (String [] args) {

String s1 = new String("java");

String s2 = new String("java");

String s3 = "java";

String s4 = "java";

}

}
```

Memory allocation:



1. During the execution of first statement using new operator, JVM will create two objects, one with content in heap area and another as a copy for literal "java" in the SCP area for future purposes as shown in the figure.

The reference variable s1 is pointing to the first object in the heap area.

2. When the second statement will be executed, for every new operation, JVM will create again one new object with content "java" in the heap area.

But in the SCP area, no new object for literal will be created by JVM because it is already available in the SCP area. The s2 is pointing to the object in the heap area as shown in the figure.

3. During the execution of third and fourth statements, JVM will not create a new object with content "java" in the SCP area because it is already available in string constant pool.

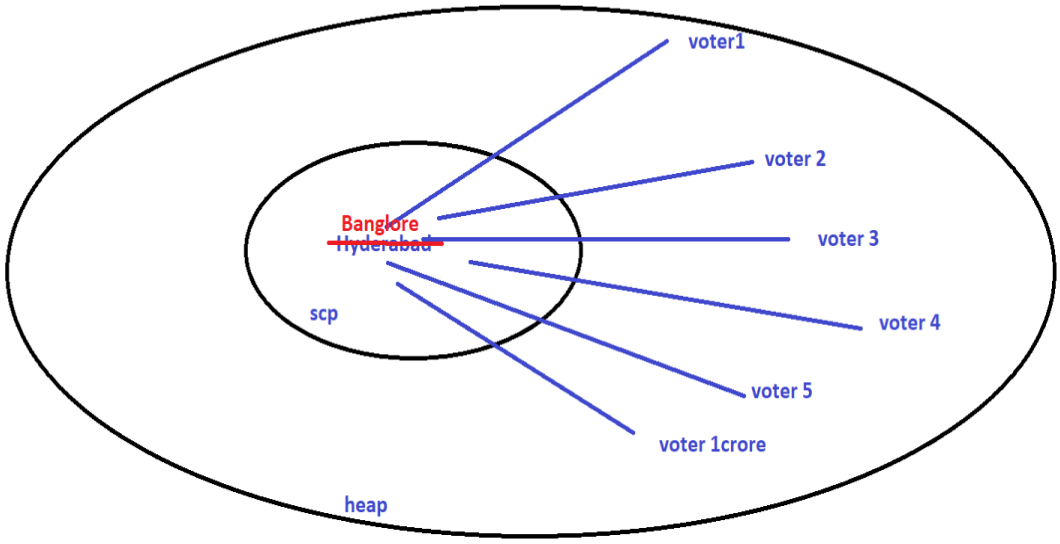
It simply points the reference variables s3 and s4 to the same object in the SCP. They are shown in the above figure.

Thus, a total of three objects is created, two in the heap area and one in string constant pool.

Why in java Strings are immutable?

Need of immutability:

- If suppose there is voter registration form
- In that Scp area is used to create 1 crore voters in Hyderabad
- In scp same hyderabad object is used by all 1 crore voters
- If voter5 reallocated in Bangalore and he changed the hyderabad Object to Bangalore then it will affect on all other voters also.
- It means for the rest of voters means that 1 crore voters also the string object modified to Bangalore.
- So,here abnormal problem will be happening
- So, to overcome this immutability concept is introduced.
- Immutability means not changeable.



About Immutability:

- So, if anyone want to change the current object it is not possible in existing object
- If you made changes then the new Object is created with the changes
- So, this concept is called immutability

```
package com.strings;


public class StringsImmutability {


public static void main(String[] args) {

// TODO Auto-generated method stub

String s=new String("jamesGosling");

System.out.println(s);

System.out.println("After concatination of java to String");

s.concat("java");

System.out.println(s);


}
```

}

Output:

jamesGosling

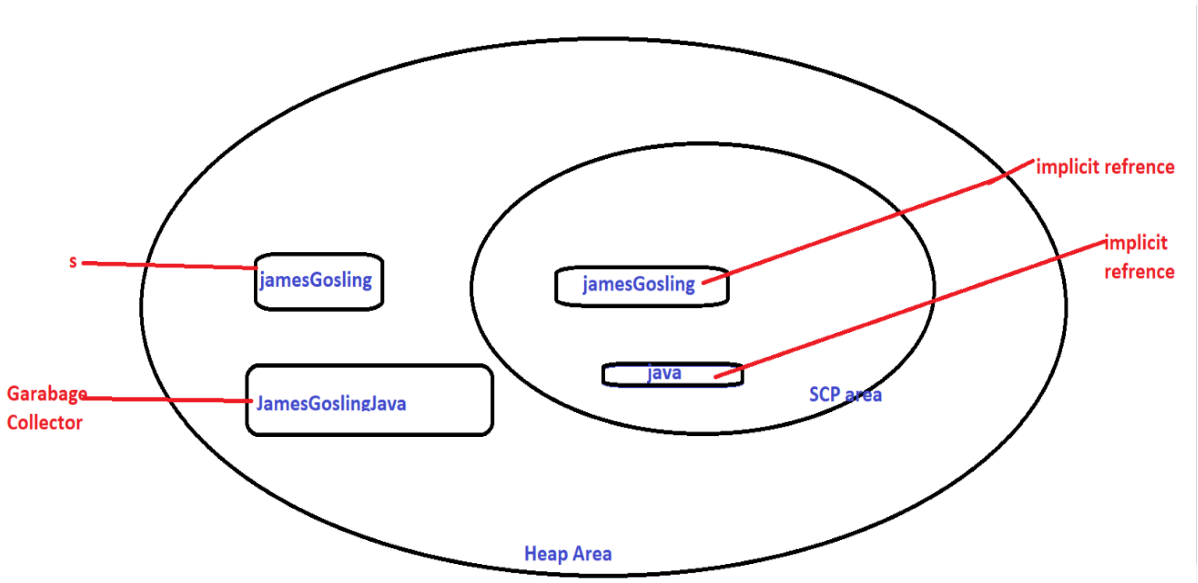
After concatination of java to String

JamesGosling

Notepoint:

- When you Observe the above program there is no change in Existing String s.
- When we try to print same string after concatination.
- There is no change in String.
- This is called Immutability of String.

Memory allocation:



Notepoint:

- What is happening here when we use to concate the jamesGosling Object with Java Object then a new object is created with the Object JamesGoslingJava.
- And no Object reference is allocated to it in heap area.
- So, garbage collector takes the responsibility of that particluar Object.

What is the need of String Buffer and String Builder In java?

Need of String Buffer:

- Strings are Immutable because of StringConstant Pool area in Strings.
- So,in String Buffer there is no concept of StringConstantPool area.
- So,In stringBuffer the objects are mutable.

Need of String Builder:

- The only difference between String Buffer and String Builder is
- String Builders are not synchronized it means at a time multiple threads are run.
- But in case of StringBuffer it is synchronized it means at a time it can run only one thread.
- StringBuilder is mutable and there is no concept of StringConstant pool area in StringBuilder also
- StringBuilder is introduced in 1.5 version whereas StringBuffer is introduced in 1.0 Version.

Constructors in String Class:

1.Create an empty String Object

String s=new String ();

2.create equivalent string for given String Literal

String s=new String(String literal);

Eg:String s=new String("varshi");

3.Create equivalent string to StringBuffer Object

String s=new String(StringBuffer sb);

4.Create Equivalent String to StringBuilder Object

String s=new String (StringBuilder sb);

5.Create Equivalent String for given char Array Object.

String s=new String (char [] Ch);

6.Create Equivalent String for given Byte Array Object

String s=new String(byte [] b);

So,in byte we give the byte values and when we are converting to String then it changed by using ASCII values.

Program:

```
package com.strings;


public class StringConstructors {

public static void main (String [] args) {


//              1.creating an empty String

String s=new String ();


//              2.Creating a equivalent string for String Literal

String s1=new String("String literal");
```

```
//              3.creating equivalent string to StringBuffer Object
StringBuffer sbuf=new StringBuffer("StringBuffer object");
String s2=new String(sbuf);

//              4.creating equivalent string to StringBuilder Object
StringBuilder sbuild=new StringBuilder("StringBuilder object");
String s3=new String(sbuild);

//              5.creating equivalent string to char array Object
char arr[]= {'c','h','a','r','a','r','r','a','y'};
String s4=new String(arr);

//              6.creating equivalent string to int array Object
byte arr1[]= {97,98,99,100,101};
String s5=new String(arr1);


//              prinitng all the string values
System.out.println(s);
System.out.println(s1);
System.out.println(s2);
System.out.println(s3);
System.out.println(s4);
System.out.println("bytearray:"+s5);

}
```

}

Output:

String literal

StringBuffer object

StringBuilder object

chararray

bytearray: abcde

Notepoint:

So,in output the first statement is printing empty characters it means it is printing the empty string.

String Comparison:

- While creating any object in memory it is assigned with reference variable following with the address
- Means if s1->pointing to varshi (then one address is also there for reference variable that is like 135f3e like that the reference address will be there)

String Comparison is done by three ways:

1.By using (==) double equals operator.

2.By using equals method.

3.By using compareTo method.

1.By using (==) double equals operator

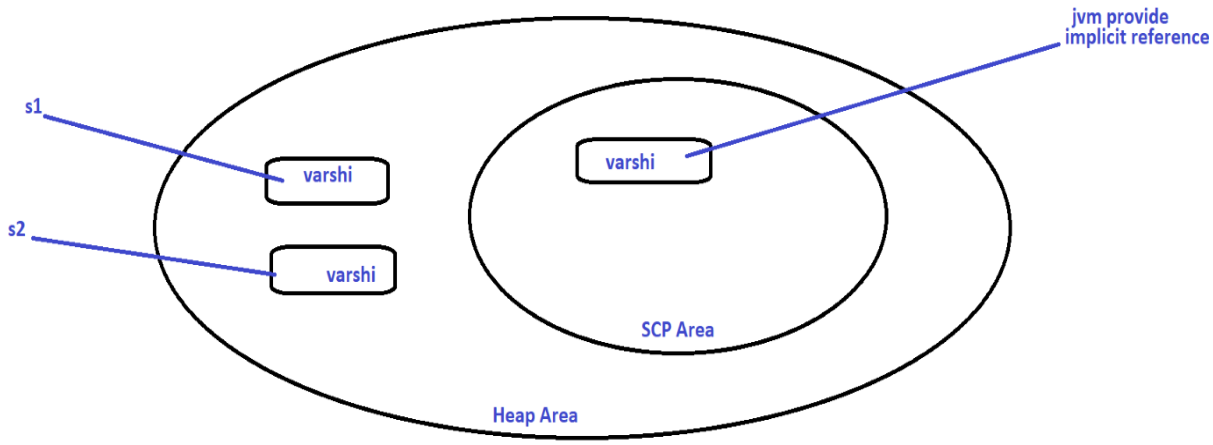
- Double equals in String always used for reference Comparisions/address Comparisions
- If both the Objects referring to same Object only it returns true
- If reference pointing to another Objects, then automatically false.

Program:

```
package com.strings;


public class StringComparision {
```

```
public static void main (String [] args) {

String s1=new String("varshi");

String s2=new String("varshi");

System.out.println(s1==s2);

}

}
```

Memory allocation:



Output:

False

Notepoint:

Since varshi object is referring two separate object one is s1 and another one is s2.

So, both are pointing two different reference the output will be false because (==) double equals are used for reference comparisons.

2.By using equals method

- Generally, equals method is an Object class method
- In object class equals method it is used for reference Comparisions only
- But in stringclass equals method it is overridden to content Comparisions.

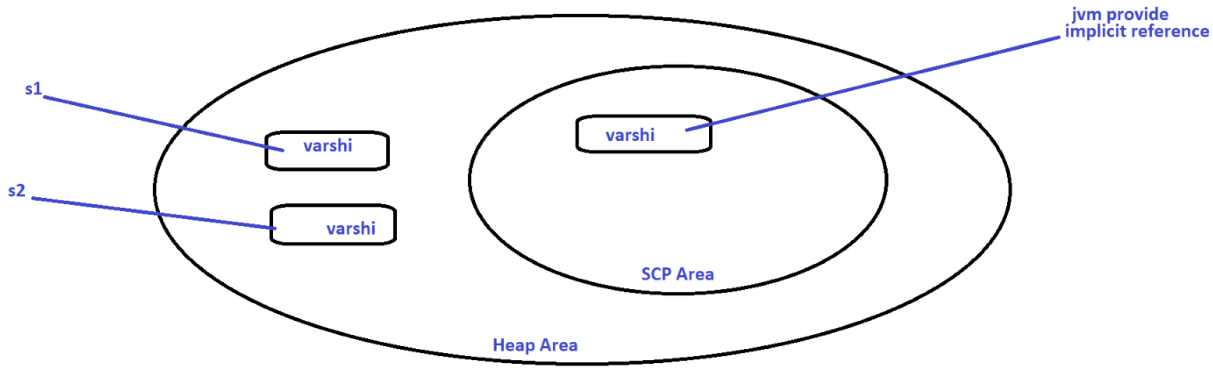
Program:

```
package com.strings;


public class StringComparision {
public static void main (String [] args) {
String s1=new String("varshi");
String s2=new String("varshi");
System.out.println(s1. equals(s2));
}
}
```

Output:

True

Memory Allocation:



Notepoint:

Since s1 and s2 are referring separate reference but equals method will check whether the content is same or not.

So, it will return true as output for above program.

3.By using CompareTo method:

- The string compareTo() method compares the current string object with specified string object in the method argument lexicographically and returns an integer value.
- Basically, it compares two strings on the basis of the Unicode value of each character in the string. Each character of both strings is converted into a Unicode value for comparison.

The general form of this method is given below:

public int compareTo(String str)

Here, str is a specified string object being compared with calling string object.

For example:

1. s1.compareTo(s2);

where s1 and s2 are two reference variables of string literals.

- If s1 = = s2, this method returns zero. It means that two strings are equal.
- If s1 > s2, +ve value. It means that first string is lexicographically greater than second string.
- If s1 < s2, -ve value.It means that first string is lexicographically less than second string.

2: s1.compareTo("Hello Java");

where s1 is a string literal and its value is compared with string specified in the method argument.

Program:

```
package stringPrograms;

public class CompareToMethodTest
```

```
{
public static void main (String args[])
{
String s1 = "mumbai";
String s2 = "mumbai";
String s3 = "ranchi";
String s4 = "pune";
String s5 = " ";

System.out.println(s1.compareTo(s2));
System.out.println(s1.compareTo(s3));
System.out.println(s1.compareTo(s4));

System.out.println(s3.compareTo(s4));
System.out.println(s4.compareTo(s5));
}
}
```

Output:

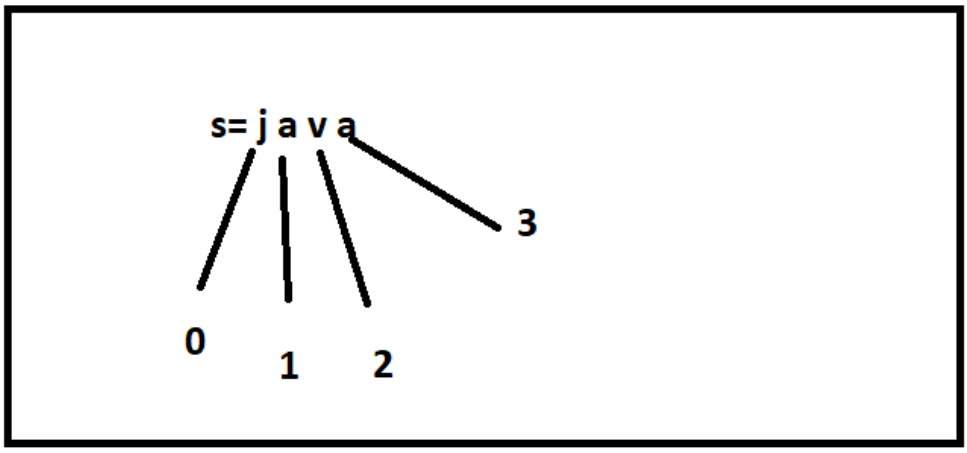
```
 0
 -5
-3
 2
 4
```

Important methods Of String Class:

1.public char charAt(int Index):

- The charAt() method return type is char
- So,here whenever we are creating java object
- So,in CharAt method if you give index means 1 or 2..
- The index start from '0'
- It will return the particular character at particular index mentioned.

Eg:String s="java";



Program:

```
package com.strings;


class Stringmethod {

public static void main (String [] args) {

String s = "java";

System.out.println(s.charAt(3));

System.out.println(s.charAt(5));

}

}
```

Output:

a

Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String index out of range: 5

at java.base/java.lang.StringLatin1.charAt(StringLatin1.java:48)

at java.base/java.lang.String.charAt(String.java:712)

at JavaBasics/com.strings.Stringmethod.main(Strinmethod.java:7)

Notepoint:

- So,here the first print statement will return 'a' because at third index a character is present
- But the second print statement will raise runtime Exception because that string contains index from 0 to 3 when we are trying to print 5^{th} index then it will raise runtime exception that is StringIndexOutOfBound Exception.

2.public String Concat(String s);

String concatination is done by 2 ways

1.By using concat method

String s="varshi";

s.concat("software");

System.out.println(s);

2.By using "+" operator also we can concatinate String

Program:

```
package com.strings;


public class StringConcatination {

public static void main (String [] args) {

//                1.By using concat method
```

```
String s="jamesGosling";
System.out.println(s.concat(" java"));

//              2.By using "+" string concatination
String s1="varshi";
System.out.println(s1+" "+"java");

}
}
```

Output:

```
jamesGosling java
varshi java
```

3.public boolean equals (Object O)

- In string class equals method is overridden for content Comparisions.
- To check equality of String Objects
- So,here case is valid
- If both the content are in same case only then it returns true otherwise false.

4.public boolean equalsIgnoreCase(String s)

- To Check equals of String Objects where case is ignored
- Eg:I am taking one real time example that Gmail account
- So,in mailId you can use either lowercase or uppercase it is not problem
- But in case of password compulsory important to provide uppercase or lowercase is mandatory

Program:

```
package com.strings;
import java.util.Scanner;

public class Gmail {
```

```
public static void main (String [] args) {
Scanner sc=new Scanner (System.in);
System.out.println("enter mail id:");
String mailId=sc.next();
System.out.println("enter password:");
String pswd=sc.next();
if(mailId.equalsIgnoreCase("java@gmail.com")) {
if(pswd.equals("JAVA123")) {
System.out.println("login Successful");
}
else {
System.out.println("password is incorrect");
}
}


}
}
```

Output1:

```
enter mail id:
JAVA@GMAIL.COM
enter password:
JAVA123
login Successful
```

Notepoint:

Since here the original user id is in lowercase but when I am validating I had entered in Uppercase

It is accepting both uppercase and lowercase because here it is using equalsIgnoreCase() Method.

Output2:

enter mail id:

java@gmail.com

enter password:

Java123

password is incorrect

Notepoint:

- So,here the password I given is in lowercase but the original password is in uppercase
- So,here it is displaying password is incorrect because of equals method
- In equals method it also checks the case also if the case also matched only it is accepted.

5.public Boolean isEmpty()

This method is used to check whether the String is empty or not

If it is empty, it will return true Otherwise it will return False

6.public int length ():

If the string is not empty if we want to know the no: of characters present in String then we go for length method.

Don't be confused between length variable and length method

Length variable for arrays concept

Int [] x={1,2,3,4};

System.out.println(x.Length); //4

System.out.println(X.length()); // Compile time error because length method is applicable for strings only

Program:

```
package com.strings;


public class StringLength {
public static void main (String [] args) {
String s="";
System.out.println("String s is empty or not:"+s.isEmpty());
String s1="varshi";
if (s1. isEmpty())
{
System.out.println("String is empty");
}
else {
System.out.println("The string s1 is not empty and the length is:"+s1.length ());
}
}
}
```

Output:

```
String s is empty or not: true
The string s1 is not empty and the length is:6
```

7.Public String replace (char old, char new):

To replace every Occurrence of old character with new character we go for replace method

8. public String substring(int begin):

These methods return substring from begin index to end index.

9.public String substring (int begin,int end)

This method returns substring from begin index to end index.

For example:

System.out.println(3,6);

Then it will return substring from 3 to 5^{th} index.

10.public int indexOf(char Ch);

It will return the index of specified character

If that specified character is not available then it returns –1.

If the same character available at multiple times, then it will return the first occurrence of the index.

11.public int lastIndexOf(char Ch)

If the specified character occurrence is Mutiple times then lastIndexOf() method will return the last occurrence of the specified char.

Now you will get a doubt for second index, third index is there any method

So,here Api is available for most commonly used methods if you want 2^{nd},3^{rd} occurrence you should implement the code for it

12.Public String toLowerCase();

Whatever the String contains it all converts to lowerCase

13.public String toUpperCase();

Whatever the string contains it all converts to uppercase

Here look at on example

```
import java.util.Scanner;


public class Sample {

public static void main(String[] args) {
```

```
Scanner sc=new Scanner(System.in);
System.out.println("enter your cityName");
String name=sc.nextLine();
if(name.equals("hyderabad")) {
System.out.println("Hi Hyderabadians");
}
else if(name.equals("chennai")) {
System.out.println("Hello madrasian");
}
else if(name.equals("banglore")) {
System.out.println("hii banglore");
}
else {
System.out.println("Must use valid city name");
}
}
}
```

In above application there are several problems are there

So, who is responsible to use this application?

End-user will use this application

But end user doesn't no following things

1.The end-user doesn't know the city name should be only in lowercase letters.

So,to Overcome this problem we want make changes in application

For first problem there are two ways to solve casesensitive problem

One is by using equalsIgnoreCase() Method in each condition

The second way is that converting the given string into Lowercase and checking the condition

So,here the second way is better because once we can convert into lowerCase and we can use through the application

But while using first way every time in condition we should apply equalsIgnoreCase() method.

2.So, some users while entering the city name they will give space before entering it is also not accepting here.

So,to Overcome this problem we want make changes in application

So,to remove empty places before the String we can use

public string trim ()

This method will remove all the white spaces before and after String

But trim method can not remove white spaces between the Strings.

```
import java.util.Scanner;


public class Sample {
public static void main(String[] args) {
Scanner sc=new Scanner(System.in);
System.out.println("enter your cityName");
String name=sc.nextLine().toLowerCase().trim();
if(name.equals("hyderabad")) {
System.out.println("Hi Hyderabadians");
}
else if(name.equals("chennai")) {
System.out.println("Hello madrasian");
}
```

```
else if(name.equals("banglore")) {
System.out.println("hii banglore");
}
else {
System.out.println("Must use valid city name");
}
}
}
```

So, now Once Observe Output:

enter your cityName

HyderabaD

Hi Hyderabadians

Notepoint:

So,here when the enduser is entering the city name with whitespaces and in uppercase letter it is working now.

So, whenever we are developing an application developer is responsible to solve any problem in the code.

How to create our own immutable class:

```
package com.strings;

final class Test {
private int i;
Test(int i){
this.i=i;
}
public Test modify(int i) {
```

```
if(this.i==i) {
return this;
}
else {
return new Test(i);
}

}

public static void main (String [] args) {
Test t1=new Test (10);
Test t2=t1.modify(100);
Test t3=t1.modify(1000);
System.out.println(t2==t3);
System.out.println(t1==t2);
}
}
```

Output:

False

False

Notepoint:

So,when we declared final to the variable no one override to change the code .

So,here I am modifying 10 to 100 .

So,if here when the change is occuring then I want to create new Object.

if there is no change in content existing object will be used .

So,here because of modify method the class will became immutable.

So,string class every method is implemented like this.

Final versus immutability:

Final is different from immutability

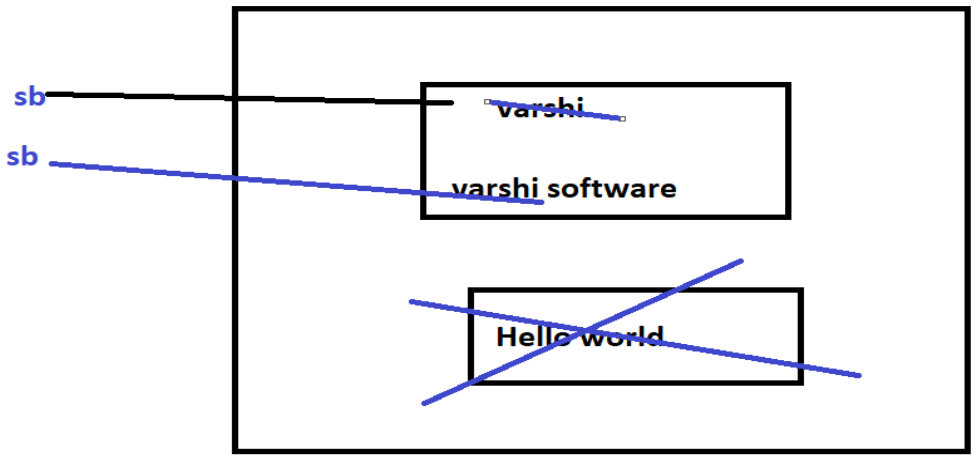
By declaring final to the reference, we cannot get the immutability to the class.

we can perform any changes in final variable but we cannot create new Object because we had declared final to that project one assigned can not be reinitialized in java

```
package com.strings;


public class SamplefinalAndImmutability {

public static void main (String [] args)
  {
     final StringBuffer sb = new StringBuffer("varshi");


     // Even though reference variable sb is final

     // We can perform any changes

     sb.append("software");


     System.out.println(sb);


     // Here we will get Compile time error

     // Because reassignment is not possible for final variable

     sb1 = new StringBuffer("Hello World");
```

```
        System.out.println(sb);
    }
}
```

Memory Allocation:



Output:

Exception in thread "main" java.lang.Error: Unresolved compilation problem:

sb1 cannot be resolved to a variable

at JavaBasics/com.strings.SamplefinalAndImmutability.main(SamplefinalAndImmutability.java:16)

StringBuffer class:

Already String is there what is the need of StringBuffer.

There are some situations where we cannot use String.

Assume if that content is fixed and the content is not going to change at runtime then we go for StringConcept.

If the content is not fixed and keep on changing the never recommended to use String Concept

This is because of immutability of string because of StringConstantpool Area.

so,to make the strings mutable StringBuffer concept is introduced.

If suppose if you want to change the object 10 times then 10 objects will be created and unnecessary performance is going down in strings

If content keep on changing then we go for StringBuffer so,in that all required changes will performed by existing Object only.

Why Stringbuffer is mutable?

It does not contains the concept of StringConstantPool area.

```
package com.strings;


public class StringBufferMutability {

public static void main(String[] args) {

StringBuffer s=new StringBuffer("james Gosling");

System.out.println(s);

s.append(" java");

System.out.println("After appending java to StringBuffer");

System.out.println(s);

}

}
```

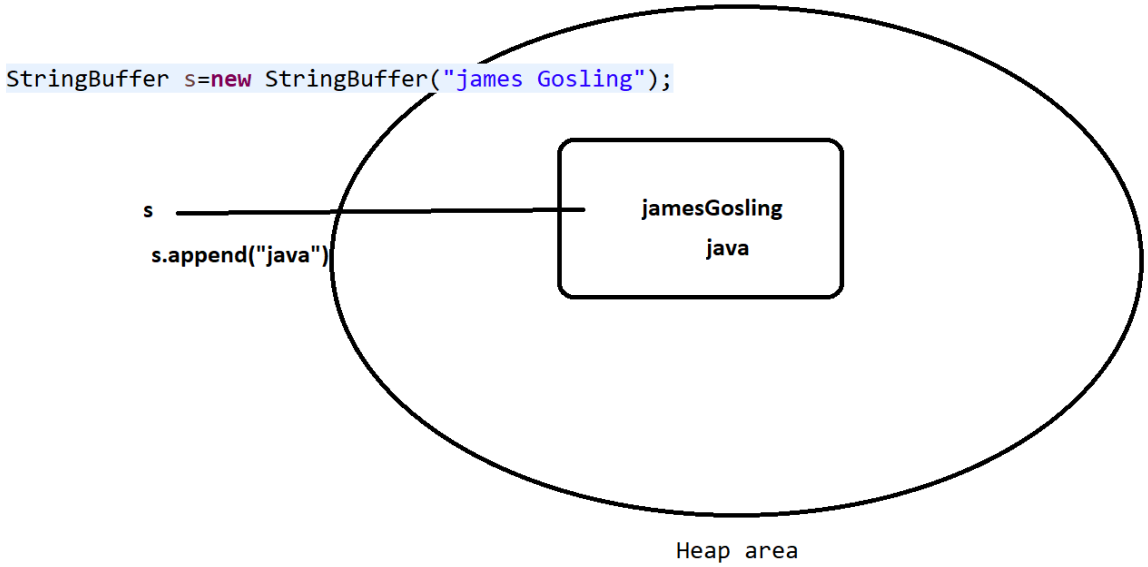
Output:

james Gosling

After appending java to StringBuffer

james Gosling java

Memory allocation



Constructors of StringBuffer:

1.StringBuffer sb=new StringBuffer();

So, the default capacity of StringBuffer is 16.

Once the StringBuffer is full of its capacity then new StringBuffer Objects will be created

With "34".

new capacity=(currentcapacity+1)*2;

new capacity=(16+1)*2=34;

If stringBuffer reaches 34 maximum capacity then bigger StringObject is created.

(34+1)*2=70

So,now 70 capacity StringBuffer Object is Created.

2. **StringBuffer sb=new StringBuffer(int initialCapacity);**

suppose in some cases I know I want to add 1000 characters in that string.

So,first 16->34->70->142>286

So,here every time performance is going to down.

So,we can create bigger StringBuffer.

StringBuffer sb=new StringBuffer(1000);

So, based on our requirement we can create bigger String Object at beginning only.

Once stringBuffer reaches 1000 then the formula is applied i.e.,

Defaultcapacity=(oldcapacity+1)* 2

3. StringBuffer sb=new StringBuffer(String s);

For given string I want to create equivalent stringBuffer Object.

StringBuffer sb=new StringBuffer("varshi");

So,here the capacity is 16+length of string=16+6=22 is its capacity.

Example:

```
package com.strings;


public class StringBufferConstructors {
public static void main (String [] args) {
//              creating empty stringbuffer with default capacity
StringBuffer sb=new StringBuffer();
System.out.println("empty StringBuffer Constructor capacity:"+sb.capacity());
//              creating StringBuffer with initial capacity
StringBuffer sb1=new StringBuffer(1000);
System.out.println("StringBuffer Constructor capacity:"+sb1.capacity());
//              for given String creating equivalent string object
```

```
StringBuffer sb2=new StringBuffer("varshi");
System.out.println("stringbuffer capacity for equivalent string:"+sb2.capacity());
}
}
```

Output:

```
empty StringBuffer Constructor capacity:16
StringBuffer Constructor capacity:1000
stringbuffer capacity for equivalent string:22
```

Methods of StringBuffer class:

1.public int length ();

Here length method returns no: of characters in that StringBuffer.

Length always starts from the index 1.

2.public int capacity ();

Here it will return the capacity of that particular StringBuffer.

Length is different from capacity

Length means no:of characters in that particular string.

Capacity means no:of characters it can able to store

3.public char charAt(int index)

```
StringBuffer sb=new StringBuffer("varshi");
System.out.println(sb.charAt(3));
```

Here it will return the character present at 3rd index.

```
System.out.println(sb.charAt(30));
```

Here there is no 30th character in string

So,here stringIndexOutOfBound Exception will raise(whether it may be string or StringBuffer same exception will raise)

4.public void setCharAt(int index,char newChar);

So,here whatever character placed in particular index replace it with new Character.

StringBuffer sb=new StringBuffer("java");

Sb.setCharAt(0,"y");

Then output will be yava.

5.public StringBuffer append(String s)

This append method is overloaded method it contains

public StringBuffer append (byte b)

public StringBuffer append (int i)

public StringBuffer append (long l)

public StringBuffer append (float f)

So, there are multiple append methods are available for StringBuffer

StringBuffer sb=new StringBuffer();

Sb.append("PI value is");

Sb.append(3.14); //double

Sb.append("it is exactly");

Sb.append(true);

System.out.println(Sb);

Here upto know we are adding string at last position

If I want to add string at specified index we go for index () method.

6.public StringBuffer insert (int index, String s)

public StringBuffer insert (int index, double d)

public StringBuffer insert (int index,boolean b)

public StringBuffer insert(int index,character ch)

So,in insert method also multiple overloaded methods are there

StringBuffer sb=new StringBuffer("abcdefgh");

Sb.insert(2,"xyz");

System.out.println(sb);//abxyzcdefgh

So,in second position xyz is added

7.public StringBuffer delete(int begin,int end)

It deletes characters from begin index to end-1 index.

StringBuffer sb=new StringBuffer("abcdefgh");

Sb.delete(2,5);

System.out.println(sb);

If suppose I want to delete character which is located at specified index means at 4th position.

8.public StringBuffer deleteCharAt(int index)

StringBuffer sb=new StringBuffer("abcdefgh");

Sb.deleteCharAt(3);

System.out.println(Sb); //abcefgh

9.If I want to reverse the content we go for

Public StringBuffer reverse ()

In strings there is no reverse method but in case of StringBuffer, it is there

Whenever we create reverse ()->then the no: of characters will be reversed.

StringBuffer sb=new StringBuffer("varshi");

Sb.reverse();

System.out.println(Sb);

package com.strings;

public class StringBufferMethods {

public static void main (String [] args) {

```
// TODO Auto-generated method stub

StringBuffer sb=new StringBuffer("varshi java");

System.out.println("To find length of stringBuffer we use this length
method:"+sb.length());

System.out.println("To find capacity of StringBuffer we use capcity
method:"+sb.capacity());

System.out.println("To find char at specified position we use this method
charAtmethod:"+sb.charAt(5));

sb.setCharAt(0, 'a');

System.out.println("after replacing character by using setChar method the result is:"+sb);

sb.setCharAt(0, 'v');

sb.append(" and jamesGosling java");

System.out.println("after append the value the string is:"+sb);

sb.insert(7, "Core");

System.out.println("after inserting value by using insert method the string is:"+sb);

sb.delete(13, 35);

System.out.println("after deleting string from specified index to specified index by using
delete method the string is:"+sb);

sb.deleteCharAt(6);

System.out.println("delete char at specified index:"+sb);

sb.reverse();

System.out.println("after reversing a string:"+sb);



}


}
```

Output:

To find length of stringBuffer we use this length method:11

To find capacity of StringBuffer we use capcity method:27

To find char at specified position we use this method charAtmethod:i

after replacing character by using setChar method the result is:aarshi java

after append the value the string is:varshi java and jamesGosling java

after inserting value by using insert method the string is:varshi Corejava and jamesGosling java

after deleting string from specified index to specified index by using delete method the string
is:varshi Corejava

delete char at specified index:varshiCorejava

after reversing a string:avajeroCihsrav

10.public void setLength(int length)

StringBuffer sb=new StringBuffer("jamesGosling");

Sb.setLength(8);

Here whenever I am setting the length to 8 then only 8 characters are allowed and remaining by default will be gone.

11.public void ensureCapacity(int capacity)

StringBuffer sb=new StringBuffer();

So,here what happens stringBuffer with capacity 16 is created

After creating I know that I want to create 1000 characters we are going to add

So,if you want to insert the capacity dynamically, we go for ensureCapacity()

Sb.ensureCapacity(1000);

12. Public StringBuffer trimToSize();

I thought that 1000 characters are going to add in the StringBuffer .so, I created bigger String Object.

StringBuffer sb=new StringBuffer(1000);

Sb.append(“abc”);

System.out.println(sb.capacity);

Sb.trimToSize();

So,here upto whenever we apply trimToSize() method then only 3 characters only it is going to Consider and remaining 97-character capacity is deallocated.

```
package com.strings;


public class StringBufferOtherMethods {

public static void main(String[] args) {

StringBuffer sb=new StringBuffer("jamesGosling");

sb.setLength(8);

System.out.println("after setting length by using setLength method the string is:"+sb);

sb.ensureCapacity(1000);

System.out.println("after dynamically entering capacity by using ensureCapacity method is:"+sb.capacity());

sb.trimToSize();

System.out.println("after triming the size:"+sb.capacity());

}

}
```

Output:

after setting length by using setLength method the string is:jamesGos

after dynamically entering capacity by using ensureCapacity method is:1000

after triming the size:8

Need Of StringBuilder and difference with StringBuffer:

Already StringBuffer is there what is the need of StringBuilder

What is the problem with StringBuffer?

So, every method present inside StringBuffer is synchronized

If a method is synchronized at a time we can run only one thread

After completion of one thread only other thread is started

Here threads are executed one by one after in StringBuffer

So,it increases the waiting time of thread and performance is going to be down.

So,problem with StringBuffer is every method present in StringBuffer is synchronized.

To overcome this in 1.5 version java people developed one StringBuilder Class

So,Non-synchronized version of StringBuffer is nothing but StringBuilder.

StringBuffer class	**StringBuilder Class**
Most of the threads present in stringBuffer are synchronized	No methods present inside StringBuilder is synchronized
At a time only one thread is allowed to operate on StringBuffer object and hence it is thread-safe	At a time multiple threads are allowed to operate on StringBuilder Object and hence it is not thread safe.
Threads are required to wait to operate on StringBuffer Object and hence relative performance is low	Threads are not required to wait to operate on StringBuilder Object and hence relative performance is high.
Introduced in 1.0 version	Introduced in 1.5 version

StringBuilder class:

Java StringBuilder class is used to create mutable (modifiable) string. The Java StringBuilder class is same as StringBuffer class except that it is non-synchronized. It is available since JDK 1.5.

Constructors in StringBuilder class:

1.StringBuilder sb=new StringBuilder ();

So, the default capacity of StringBuilder is 16.

Once the StringBuilder is full of its capacity then new StringBuilder Objects will be created

With “34”.

new capacity=(currentcapacity+1)*2;

new capacity=(16+1)*2=34;

If stringBuilder reaches 34 maximum capacity then bigger StringObject is created.

(34+1)*2=70

So, now 70 capacity StringBuilder Object is Created.

2. **StringBuilder sb=new StringBuilder(int initialCapacity);**

suppose in some cases I know I want to add 1000 characters in that string.

So, first 16->34->70->142>286

So,here every time performance is going to down.

So, we can create bigger StringBuilder.

StringBuilder sb=new StringBuilder(1000);

So, based on our requirement we can create bigger String Object at beginning only.

Once stringBuilder reaches 1000 then the formula is applied i.e.,

Defaultcapacity=(oldcapacity+1)* 2

3. StringBuilder sb=new StringBuilder(String s);

For given string I want to create equivalent stringBuilder Object.

StringBuilder sb=new StringBuilder(“varshi”);

So,here the capacity is 16+length of string=16+6=22 is its capacity.

Example:

```
package com.strings;


public class StringBuilderConstructors {
public static void main (String [] args) {
//                creating empty stringbuilder with default capacity
StringBuilder sb=new StringBuilder();
```

```
System.out.println("empty StringBuilder Constructor capacity:"+sb.capacity());
//              creating StringBuilder with initial capacity
StringBuilder sb1=new StringBuilder (1000);
System.out.println("StringBuilder Constructor capacity:"+sb1.capacity());
//              for given String creating equivalent string object
StringBuilder sb2=new StringBuilder("varshi");
System.out.println("stringbuilder capacity for equivalent string:"+sb2.capacity());
}
}
```

Output:

```
empty StringBuilder Constructor capacity:16
StringBuilder Constructor capacity:1000
stringbuilder capacity for equivalent string:22
```

Methods of StringBuffer class:

1.public int length();

Here length method returns no:of characters in that StringBuilder.

Length always start from the index 1.

2.public int capacity();

Here it will returns the capacity of that particular StringBuilder.

Length is different from capacity

Length means no:of characters in that particular string.

Capacity means no:of characters it can able to store

3.public char charAt(int index)

StringBuilder sb=new StringBuilder("varshi");

System.out.println(sb.charAt(3));

Here it will return the character present at 3rd index.

System.out.println(sb.charAt(30));

Here there is no 30th character in string

So,here stringIndexOutOfBound Exception will raise(whether it may be string or StringBuffer same exception will raise)

4.public void setCharAt(int index,char newChar);

So,here whatever character placed in particular index replace it with new Character.

StringBuilder sb=new StringBuilder("java");

Sb.setCharAt(0,"y");

Then output will be yava.

5.public StringBuilder append(String s)

This append method is overloaded method it contains

public StringBuilder append(byte b)

public StringBuilder append(int i)

public StringBuilder append(long l)

public StringBuilder append(float f)

So,there are multiple append methods are available for StringBuffer

StringBuilder sb=new StringBuilder();

Sb.append("PI value is");

Sb.append(3.14); //double

Sb.append("it is exactly");

Sb.append(true);

System.out.println(Sb);

Here upto know we are adding string at lastposition

If I want to add string at specified index we go for index() method.

6.public StringBuilder insert(int index,String s)

public StringBuilder insert(int index,double d)

public StringBuilder insert(int index,boolean b)

public StringBuilder insert(int index,character ch)

So,in insert method also multiple overloaded methods are there

StringBuilder sb=new StringBuilder("abcdefgh");

Sb.insert(2,"xyz");

System.out.println(sb);//abxyzcdefgh

So,in second position xyz is added

7.public StringBuilder delete(int begin,int end)

It deletes characters from begin index to end-1 index.

StringBuilder sb=new StringBuilder("abcdefgh");

Sb.delete(2,5);

System.out.println(sb);

If suppose I want to delete character which is located at specified index means at 4th position.

8.public StringBuilder deleteCharAt(int index)

StringBuilder sb=new StringBuilder("abcdefgh");

Sb.deleteCharAt(3);

System.out.println(Sb); //abcefgh

9.If I want to reverse the content we go for

Public StringBuilder reverse ()

In strings there is no reverse method but in case of StringBuilder,it is there

Whenever we create reverse ()->then the no: of characters will be reversed.

StringBuilder sb=new StringBuilder("varshi");

Sb.reverse();

System.out.println(Sb);

package com.strings;

```
public class StringBuilderMethods {

public static void main(String[] args) {

StringBuilder sb=new StringBuilder("varshi java");

System.out.println("To find length of stringBuilder we use this length
method:"+sb.length());

System.out.println("To find capacity of StringBuilder we use capcity
method:"+sb.capacity());

System.out.println("To find char at specified position we use this method
charAtmethod:"+sb.charAt(5));

sb.setCharAt(0, 'a');

System.out.println("after replacing character by using setChar method the result is:"+sb);

sb.setCharAt(0, 'v');

sb.append(" and jamesGosling java");

System.out.println("after append the value the string is:"+sb);

sb.insert(7, "Core");

System.out.println("after inserting value by using insert method the string is:"+sb);

sb.delete(13, 35);

System.out.println("after deleting string from specified index to specified index by using
delete method the string is:"+sb);

sb.deleteCharAt(6);

System.out.println("delete char at specified index:"+sb);

sb.reverse();

System.out.println("after reversing a string:"+sb);

}

}
```

Output:

To find length of stringBuilder we use this length method:11

To find capacity of StringBuilder we use capcity method:27

To find char at specified position we use this method charAtmethod:i

after replacing character by using setChar method the result is:aarshi java

after append the value the string is:varshi java and jamesGosling java

after inserting value by using insert method the string is:varshi Corejava and jamesGosling java

after deleting string from specified index to specified index by using delete method the string is:varshi Corejava

delete char at specified index:varshiCorejava

after reversing a string:avajeroCihsrav

10.public void setLength(int length)

StringBuilder sb=new StringBuilder("jamesGosling");

Sb.setLength(8);

Here whenever I am setting the length to 8 then only 8 characters are allowed and remaining by default will be gone.

11.public void ensureCapacity(int capacity)

StringBuilder sb=new StringBuilder ();

So,here what happens stringBuffer with capacity 16 is created

After creating I know that I want to create 1000 characters we are going to add

So,if you want to insert the capacity dynamically, we go for ensureCapacity()

Sb.ensureCapacity(1000);

12. Public StringBuilder trimToSize();

I thought that 1000 characters are going to add in the StringBuffer .so, I created bigger String Object.

StringBuilder sb=new StringBuilder (1000);

Sb.append("abc");

System.out.println(sb.capacity);

Sb.trimToSize();

So,here upto whenever we apply trimToSize() method then only 3 characters only it is going to Consider and remaining 97 character capacity is deallocated.

```
package com.strings;


public class StringBuilderOtherMethods {

public static void main (String [] args) {

StringBuilder sb=new StringBuilder("jamesGosling");

sb.setLength(8);

System.out.println("after setting length by using setLength method the string is:"+sb);

sb.ensureCapacity(1000);

System.out.println("after dynamically entering capacity by using ensureCapacity method is:"+sb.capacity());

sb.trimToSize();

System.out.println("after triming the size:"+sb.capacity());

}

}
```

Output:

after setting length by using setLength method the string is:jamesGos

after dynamically entering capacity by using ensureCapacity method is:1000

after triming the size:8

Interview Questions:

1) How many ways to create a String object & StringBuffer object?

2) What is the difference between a. String str="ratan"; b. String str = new String("ratan");

3) equals() method present in which class?

4) What is purpose of String class equals() method.

5) What is the difference between equals() and == operator?

6) What is the difference between by immutability & immutability?

7) Can you please tell me some of the immutable classes and mutable classes?

8) String & StringBuffer & StringBuilder & StringTokenizer presented package names?

9) What is the purpose of String class equals() & StringBuffer class equals()?

10) What is the purpose of StringTokenizer nd this class functionality replaced method name?

11) How to reverse String class content?

12) What is the purpose of trim?

13) Is it possible to create StringBuffer object by passing String object as a argument?

14) What is the difference between concat() method & append()?

15) What is the purpose of concat() and toString()?

16) What is the difference between StringBuffer and StringBuilder?

17) What is the difference between String and StringBuffer?

18) What is the difference between compareTo() vs equals()?

19) What is the purpose of contains() method?

20) What is the difference between length vs length ()?

21) What is the default capacity of StringBuffer?

22) What do you mean by factory method?

23) Concat() method is a factory method or not?

24) What is the difference between heap memory and String constant pool memory?

25) String is a final class or not?

26) StringBuilder and String Tokenizer introduced in which versions?

27) What do you mean by legacy class & can you please give me one example of legacy class?

28) How to apply StringBuffer class methods on String class Object content?

29) When we use String & StringBuffer & String

30) What do you mean by cloning and use of cloning?

31) Who many types of cloning in java?

32) What do you mean by cloneable interface present in which package and what is the purpose?

33) What do you mean by marker interface and Cloneable is a marker interface or not?

34) How to create duplicate object in java (by using which method)