String Manipulation --> Class

==========

String is used to represent group of characters or character array enclosed within the double quote.

example 1

```
package com.spartan;
public class Test {
	//String manipulation
	public static void main(String[] args) {
		String str = "omkar";
		System.out.println(str);
		String str1 = new String("omkar");
		System.out.println(str1);
		//conversion of char array to string format.

		char[]ch= {'o','m','k','a','r'};
		String str2= new String(ch);
		System.out.println(str2);//omkar

		char[]ch1= {'o','m','k','a','r'};
		String str3= new String(ch1,2,2);
		//give me character from second index and next 2 represents give me two characters only
		System.out.println(str3);
		//conversion of byte array to string format
		byte[] b = {65,66,67,68,69,70};
	String str4 = new String(b);
		System.out.println(str4);//it will print unicode values
		byte[] b1 = {65,66,67,68,69,70};
	String str5 = new String(b1,3,2);
```

```
                System.out.println(str5);
        } }
```

It is possible to create string object in two way.

1. without using new keyword or new operator

```
  String str ="omkar";
```

2. by using new operator

```
  String str = new String("omkar");
```

```
//creating string object without using new operator
    String str1 ="omkar";
    String str2 ="vikas";
    String str3 ="omkar";
```

the objects are created in SCP (string constant pool area)

str1,str3 -----> omkar

str2 -----> vikas

str1==str2 : false

str1==str3 : true

str2==str3 : false

== reference comparison operator

= assignment operator

In string constant pool memory just before object creation it is always checking previous object.

that means for reference str1 and str3 only one objects created in SCP.

a. if the previous object is not available it will create new object.

b. if the previous object is available with same content(data) then reference variable is pointing to existing object.

== : operator always meant for reference comparison.

SCP does not allowed duplicate object

```
//creating a string object by using new operator
        String str1 = new String("omkar");
            String str2 = new String("vikas");
            String str3 = new String("omkar");
```

The objects are created in heap memory

when we create object in heap area instead of checking previous object it directly create new object.

there fore 3 new objects are created in heap area.

str1 ----> omkar

str2 ----> vikas

str3 ----> omkar

str1==str2 : false

str1==str3 : false

str2==str3 : false

Program-:

```
package com.tcs;
public class StringManipilation {
        //String is group of character enclosed within double quote
        //java.lang package
        //by default final --->we cant inherit
        //StringBuffer--->final
        public static void main(String[] args) {
```

```
        //Strng class object creation without using new key word i.e String literal

        String s1= "java world";
        String s2= "selenium world";
        String s3= "java world";
        System.out.println(s1==s2);//false
        System.out.println(s1==s3);//true
        System.out.println(s2==s3);//false
        //string class object creation by using new keyword
        String n1 = new String("omkar");
        String n2 = new String("sagar");
        String n3 = new String("omkar");

        System.out.println(n1==n2);//false
        System.out.println(n1==n3);//false
        System.out.println(n2==n3);//false

        //string buffer also create new object every time.

        StringBuffer sb1 = new StringBuffer("omkar");
        StringBuffer sb2 = new StringBuffer("sagar");
        StringBuffer sb3 = new StringBuffer("omkar");
        System.out.println(sb1==sb2);   //false
        System.out.println(sb1==sb2);   //false
        System.out.println(sb1==sb3);   //false
    }
}
```

Example : == to vs .equals() method

==========

```
package com.spartan;
public class Text {
	Text (String name){
	}
	public static void main(String[] args) {
		//object class equals method : doing Reference comparison

		Text t1 =new Text("omkar");
		Text t2 =new Text("omkar");

		System.out.println(t1.equals(t2));//false
		System.out.println(t1==t2);	//false

		//String class overriding equals method : doing content(data)comparison

		String str1= "omkar";
		String str2= "omkar";

		System.out.println(str1.equals(str2));//true
		System.out.println(str1==str2);	//true

		String s1 = new String("omkar");
		String s2 = new String("omkar");

		System.out.println(s1.equals(s2));//true
		System.out.println(s1==s2);	//false
```

```
                //string buffer class can not overriding equals method of object class : it uses object class equals method
                StringBuffer sb1 = new StringBuffer("omkar");
                StringBuffer sb2 = new StringBuffer("omkar");
                System.out.println(sb1.equals(sb2));//false
                System.out.println(sb1==sb2);//false //false
        }
}
```

equals() method present in object class doing reference comparison and it returns Boolean value.

if two reference variable are pointing to same object it returns true otherwise false

string is a child class of object class and it is overriding equals method & doing Content (data) comparison.

if two objects Data is same then returns true otherwise false.

stringbuffer class is child class of object and it is not overriding equals method hence it is using object class equals method

& doing reference comparison.

note-: equals method class to Class behaviour is changed

but == operator always meant for reference comparison.

example : compareTo Vs .equals method

equals() method returns boolean value

String class equals() method doing Content comparison.

omkar omkar = true

omkar vikas = false

compareTo() methods returns----> integer value .

a=97 A =65

```
package com.spartan;
public class Test1 {
	//compareTo() method vs equals()
	public static void main(String[] args) {
		String str1 = "ramu";
		String str2 = "naresh";
		String str3 = "ramu";

		//first we will go for equals method

		System.out.println(str1.equals(str2));//false
		System.out.println(str1.equals(str3));//true
		System.out.println(str2.equals(str3));//false

		System.out.println("omkar".equals("OMKAR"));//false
		System.out.println("omkar".equalsIgnoreCase("OMKAR"));//true

		//now we will go for comapareTo();
		System.out.println(str1.compareTo(str2));//+ve  integer value  114-110
		System.out.println(str1.compareTo(str3));//0
		System.out.println(str2.compareTo(str3));//-ve value  -4

		System.out.println("omkar".compareTo("OMKAR"));
		System.out.println("omkar".compareToIgnoreCase("OMKAR"));

	}}
```

omkar omkar = 0 --> if strings are same then output is zero

omkar aadi = +ve -- if the first string first character/letter Unicode value is bigger than the second string first character Unicode value

then output is positive

adi omkar = -ve if the first string first letter Unicode value is lesser than the second string first letter Unicode value then output is negative.

possible to concat the string data in two way

==========

approach 1-: using concat

approach 2-: using + operator

```
package com.spartan;

public class A {
	//concatination of string
	public static void main(String[] args) {
		String str1= "Om";
		String str2= "sairam";
		//Approch 1 : Using conact()
		String result=str1.concat(str2);
		System.out.println(result);//omsairam
		//Approch 1 : Using '+' operator
		String r = str1 + str2;
		System.out.println(r);
	}
}
```

String Class Methods

==========

1. trim()

2. length()

3. charat()//reverse string

4. indexOf()

5. lastIndexOf()

6. endsWith()

7. startsWith()

8. split()

9. toUpperCase()

10.toLowerCase()

11.contains()

12.replace()

13.substring(2)

14.substring(2,5)// start index included and end index exluded.

```
package com.str;
public class StrinMethods {
	public static void main(String[] args) {
		String s1="  omkar  ";
		System.out.println(s1);
		System.out.println(s1.trim());
		//trim method is used to remove space before string and after string
		System.out.println(s1.length());

		//length method is used to get size of string.
		//method chaining
		System.out.println(s1.trim().length());
		String s2="omkarrsss";

		System.out.println(s2.charAt(2));//k
```

```
		//charAt() method is  used to know character present in perticular index.
		//System.out.println(s2.charAt(11));//java.lang.StringIndexOutOfBoundsException

		System.out.println(s2.indexOf('r'));

		//indexof method is used to get index of given character
		System.out.println(s2.lastIndexOf('r'));

		//indexof method is used to get index of given character but character searching starts from last
		String s3="hi sir";
		System.out.println(s3.startsWith("hi"));//boolean value
		//startsWith method is used to know weather given string is starts with character or words that we r passing as an argument.
		System.out.println(s3.endsWith("u"));
		//startsWith method is used to know weather given string is end with character or words that we r passing as an argument.
		System.out.println(s3.toUpperCase());
		//data is printed in  upercase

		System.out.println(s3.toLowerCase());
		//data is printed in  lowercase

		String s4="hi omkar sir";
		System.out.println(s4.contains("how"));
		//it returns boolean value
		//contains method is used to check that given argument is present in our string or not

		String s5="hi vikas vikas sir how r u";
```

```
		System.out.println(s5.replace('v', 'o'));
		System.out.println(s5.replace("vikas", "omkar"));
		///used to replace the character or word with another character or word
		System.out.println(s5.replace(" ", ""));
		//replacing space with empty string.

		String s6 = "omkarit";
		System.out.println(s6.substring(2));
		System.out.println(s6.substring(2, 5));
		//substring method is used to get the required string from original string.
		//it include the bigin-index and exlude the end-index
		String [] s = s3.split(" ");//used to split the string.
		//store the output of split method in array format and print array by using foe each
loop.
		for(String ss:s) {
			System.out.println(ss);
		}	}
}
```

mutable Vs Immutable

==========

```
package com.str;
public class stringVsStringBuffer {
	public static void main(String[] args) {
		//case 1
		//string is immutable : once we create the String  ,modifications are not allowed.
		String str="omkar";
		str.concat("vikas");
```

```
System.out.println(str);//omkar


//string Buffer is mutable : it allows the modification
StringBuffer sb =new StringBuffer("omkar");
sb.append("vikas");//append () use to append the two string.
System.out.println(sb);//omkarvikas

//case 2-concat method is used to combine the two string ,the resulted value is stored in new string.

String str1="omkar";
String res =str1.concat("vikas");

System.out.println(res);//omkarvikas

//here we are not doing modification in existing string object.
//in this case new string object is created having reference "res" and the value store in object is omkar vikas.

//str1----->pointing to----> omkar
//res---->pointing to------->omkarvikas (new object created)
//case 3 --> Reassigning the reference variable.
//here we are doing cocatination of two string and reassigned the same reference variable.
//so object having value "omkar" is destroyed and new object is created having same reference that is "str2".

String str2="omkar";
str2 =str2.concat("vikas");
System.out.println(str2);
```

```
		//str2----> pinting to---> omkar (this object is destroyed)
		//str2-----> pointing to ---> omkarvikas
		//reassigning the same reference variable to other object
		//note-: initially str2 is pointing to omkar later it is ressigned to omkarvikas, so "omkar" object is garbage collected.
		//here we are not doing any modification.
	}}
```

StringBuffer class methods

```
package com.str;
public class StringBufferMethods {
	public static void main(String[] args) {
		StringBuffer sb1 = new  StringBuffer("omkarit");
		System.out.println(sb1.delete(2, 4));//omrit, delete the required character.
		//character on start index is included but character on end index is excluded
		System.out.println(sb1.deleteCharAt(3));
		StringBuffer sb2 = new  StringBuffer("omkarit");
		System.out.println(sb2.reverse());//used to reversed string
		StringBuffer sb3 = new  StringBuffer("omkarit");
		System.out.println(sb3.append("vikasit"));
		StringBuffer sb4 = new  StringBuffer("omkar");
		System.out.println(sb4.insert(0, "hi"));
		StringBuffer sb5 = new  StringBuffer("hi omkar it");
		System.out.println(sb5.replace(2,7,"vikas"));
		//you can replace the fields with string
    }
}
```