=> String :-

=>String is the sequence of characters or say String is an array of characters.

For example : char[] c={'a', 'r', 'u', 'n'};

-String is a non-primitive data type.

-To create String or to perform String operations, java has provided some predefined classes :-

1. java.lang.String
2. java.lang.StringBuffer
3. java.lang.StringBuilder

=>Syntax :

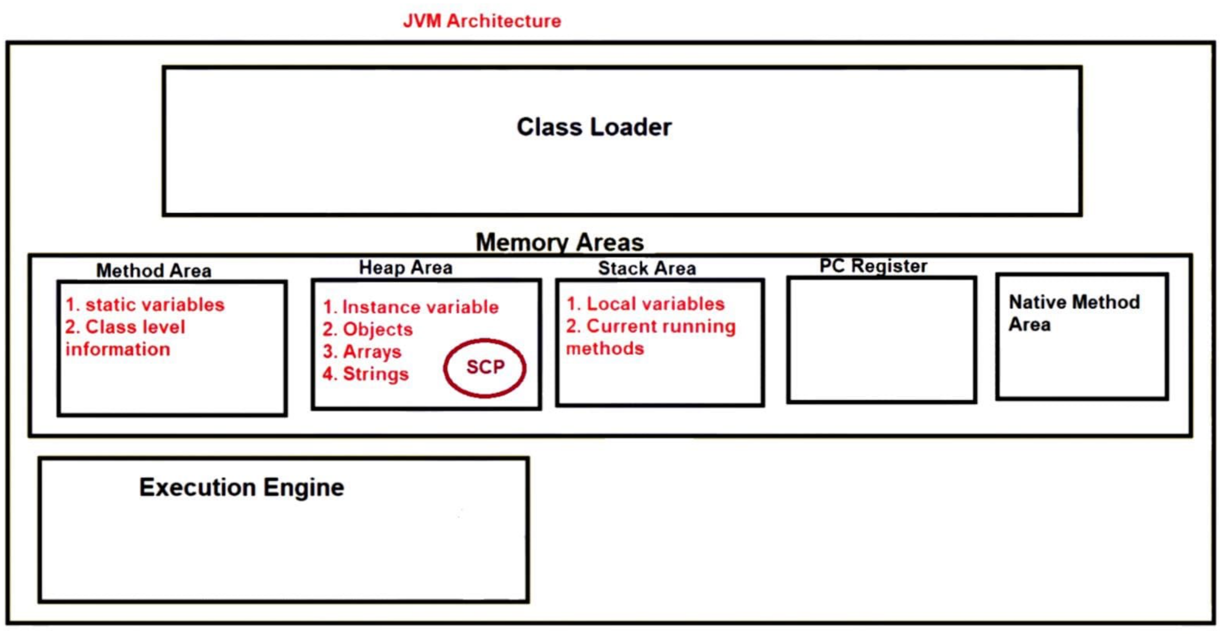
```
public final class String extends Object implements Serializable, Comparable, CharSequence
{
    //methods
}
```

-String is the class and we can create String class object. But we can create String class object by 2 ways :-

1. String str=new String("Arun");
2. String str="Arun";

=> Whenever we create String class object, objects created are "IMMUTABLE"

=>Whenever we create String objects, it allocates memory in special memory area i.e. "String Constant Pool" or "String Literal Pool"



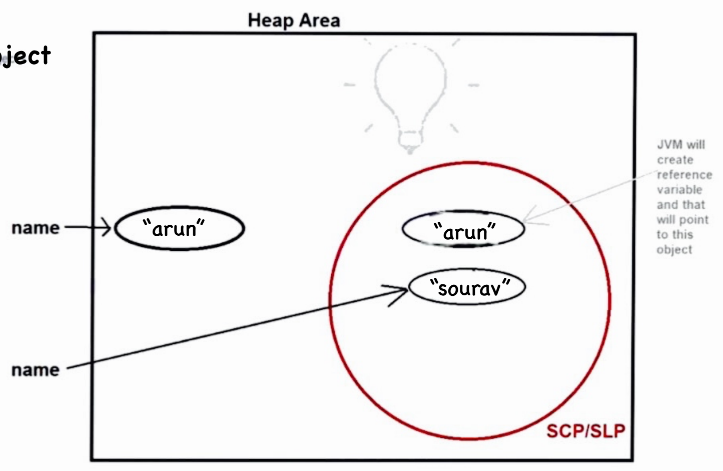
=>Garbage collection is not applicable for String Constant Pool

=> Difference between creating String objects by "new keyword" and by "String literal"

-If we create String object by using new keyword then an object is created in heap area. If we have provided any string literal in string constructor then 2 objects will be created and second object will be created in String Constant Pool

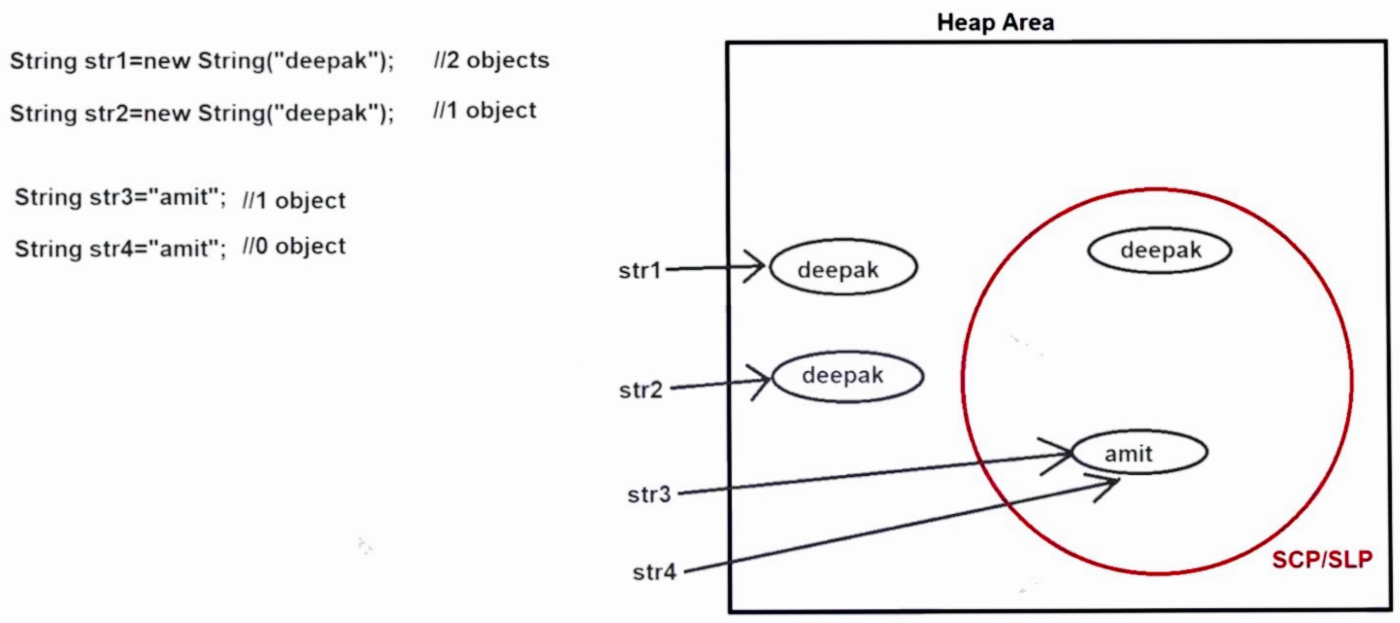
-If we create String object by using String literal then an object is created in String Constant Pool

```
String name = new String("arun"); //2 Object
String name = "sourav"
```


Heap Area
name
"arun"
name
"arun"
"sourav"
SCP/SLP
JVM will create reference variable and that will point to this object

=> Properties of "String Constant Pool" or "String Literal Pool"

1. SCP stores the String Literal Objects
2. Whenever String Literal Object is created in SCP, first it will check wether that literal object is already present in SCP or not, if it is not present then it will create new object otherwise it will not create new object and that reference variable will point to that same object



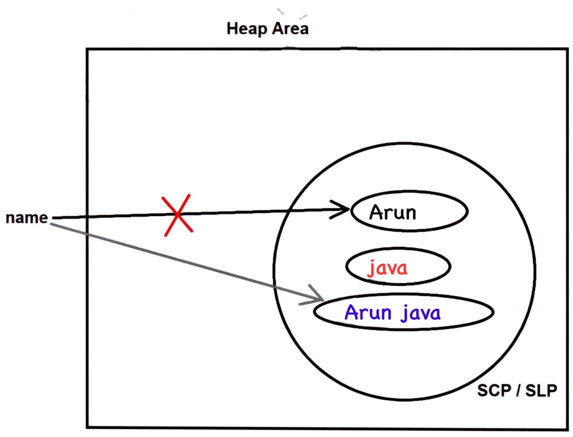
```
String str1=new String("deepak");     //2 objects

String str2=new String("deepak");     //1 object

String str3="amit";  //1 object

String str4="amit";  //0 object
```

=> Why String objects are immutable ?

-String objects are immutable which means that if we create any String object, then we cannot change its value
-String objects are immutable because if we change any string object value, then it will create new object and will not affect other reference variables

NOTE : Strings are not immutable, String objects are immutable

```
public class Main {
    public static void main(String[] args)
    {
        String name="Arun";
        //name=name+" java";             // Arun java
        //name=name.concat(" java");    //Arun java
        name.concat(" java");           // Arun
        System.out.println(name);
    }
}
```



=> What String class is "final" ?

-> String class is final because we cannot inherit String class and thus we were not able to change the String class properties

```
public class Test1
{
    public static void main(String[] args)
    {
        String str1=new String("Arun");
        System.out.println(str1);  //Arun

        System.out.println("=====================");

        byte[] b={97, 98,99,100};
        String str2=new String(b);
        System.out.println(str2);   //abcd

        System.out.println("=====================");

        char[] c={'a','b'};
        String str3=new String(c);
        System.out.println(str3);  //ab
    }
}
```

=> String class Methods :-

Validating User Input

- length() - It provides the int value i.e. no of characters in the String
- isEmpty() - Checks wether string is empty or not, if empty it will return true else false
- trim() - It is used to remove the front or back side spaces

Comparing two strings :-

- equals() - it compares the 2 strings and retruns the boolean value
- equalsIgnoreCase() - it compares the 2 strings but ignores the uppercase and lowercase
- compareTo() - It compares the 2 strings lexicographically(means convert in ASCII value and then compare)
- compareToIgnoreCase()

```java
public class Test2
{
    public static void main(String[] args)
    {
        String str1=new String("Arun");

        String str2="Arun";
        System.out.println(str2.length());

        System.out.println("======================");

        System.out.println(str2.isEmpty());

        System.out.println("======================");

        String str="          Arun                    Kumar           Sharma            ";
        System.out.println(str.trim());
    }
}
```

Contination of two strings :-

- '+' operator
- concat() - It concats the 2 strings

Get sub-string from String :-

- subString() - It will return the sub string according to the provided starting and ending index positions
- subSequence() - It will return the CharSequence according to the provided starting and ending index positions

Replacing or Removing characters :-

- replace() - It will replace the string according to provided another string
- replaceFirst() - It will replace the first string
- replaceAll() - it will replace all the strings

Searching characters in string :-

- indexOf() - It will retrun the index position of provided character
- lastIndexOf() - It will retrun the index position of provided character from last position
- contains() - It will return boolean value, true if string present else false
- charAt() - It will return the index position of provided character
- endsWith() - It will return boolean value if string is matched from ending
- startsWith()- It will return boolean value if string is matched from starting

Case conversion methods :-

- toLowerCase() - It will convert the string into lowercase
- toUpperCase() - It will convert the string into uppercase

Type conversion methods :-

- valueOf() - It will convert other data type into string (it is static method)
- toCharArray() - It will convert the string into character array

Other method :

- split() - It will split the string according to provided regex

```
public class Main {
    public static void main(String[] args)
    {
        String str1="Arun@gmail.com";
        String str2="arun123";

        System.out.println(str1.equals(str2));
        if(str1.equalsIgnoreCase("arun@gmail.com") && str2.equals("arun123"))
        {
            System.out.println("login successfully");
        }
        else
        {
            System.out.println("failed");
        }

        //-------------------------------------------------------------
        String str3="a";
        String str4="A";
        System.out.println(str3.compareTo(str3));      //32
        System.out.println(str4.compareToIgnoreCase(str3));    //0


        char c1='a';    //97
        char c2='A';      //65
        System.out.println((int)c1);

        String str5="arun";
        String str6="java";
        System.out.println(str5+str6);                //arunjava
        System.out.println(str5.concat(str6));     //arunjava

        //-------------------------------------------------------------

        String str7="arun java";
        System.out.println(str7.substring(3, 8));      //3 included, 8 excluded
        System.out.println(str7.subSequence(3, 8));  ////3 included, 8 excluded

        //----------------------------------------------------------

        String str8="arun java";
        System.out.println(str8.replace("arun", "Deepak"));
        System.out.println(str8.replace("e", "x"));
//-------------------------------------------------------------------

        String str9="arun is teaching java";
        //System.out.println(str9.lastIndexOf("is"));
        //System.out.println(str9.contains("un"));
        //System.out.println(str9.charAt(7));
        System.out.println(str9.startsWith("aru"));   //true
        System.out.println(str9.endsWith("java"));   //true

        //-------------------------------------------------------------

        String str10="Arun sharma";
        System.out.println(str10.toLowerCase());

        //---------------------------------------------------------------
        int rollno=1001;
        System.out.println(String.valueOf(rollno).length());   //convert int to String

        String str11="arun";
        char[] c=str11.toCharArray();
        System.out.println(c);

        System.out.println(str11.replaceFirst("r", "z"));

        //---------------------------------------------------------------
        String str12="this is my first demo";
        String[] str=str12.split("is");
        for(String s:str)
        {
            System.out.println(s);
        }

    }
}
```

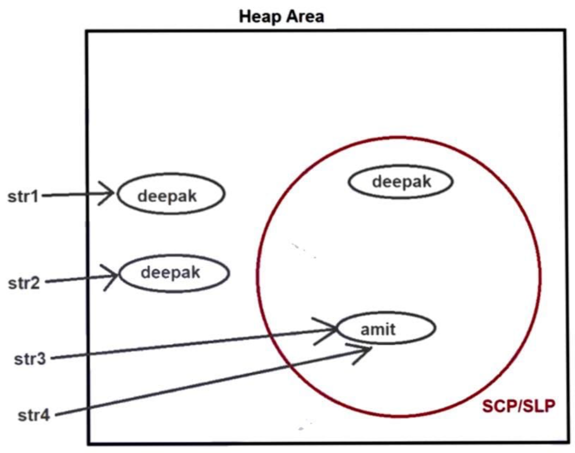
=> What is difference between equals() and == ?

1. equals() method is the "Object" class method. Object class equals() method compare the reference of 2 objects (address comparison)
2. String class overrides the equals() method of Object class

=> == operator is used for reference comparison or address comparison

=> equals() method is used for content comparison

```
String str1 = new String("Deepak") ;
String str2 = new String("Deepak") ;
System.out.println(str1==str2);         //false
System.out.println(str1.equals(str2));  //true

 System.out.println("============");

 String str3 = "Amit" ;
 String str4 = "Amit" ;
 System.out.println(str3==str4);         //true
 System.out.println(str3.equals(str4));  //true
```



=> StringBuffer class :-

-In some cases using String objects is not helpful because String objects are immutable but we want to update the same string object. If we want to update the string object again and again then java has provided one class i.e. StringBuffer class

NOTE : In case of StringBuffer string mutable object is created

-StringBuffer is the class thus it has some constructors and methods

=> StringBuilder class :-

-StringBuilder is same as StringBuffer but one difference is that all the methods of StringBuilder are non-synchronized but all the methods of StringBuffer are synchronized

=> What is difference between StringBuffer & StringBuilder :-

1. StringBuffer methods are synchronized
StringBuilder methods are non-synchronized
2. StringBuffer will take more execution time
StringBuilder will take less execution time
3. In case of StringBuffer application performance is slow
In case of StringBuilder application performance is fast
4. StringBuffer is threadsafe
StringBuilder is not threadsafe
5. StringBuffer came in JDK 1.0 version
StringBuilder came in JDK 1.5 version

```
String name="arun";
name.concat("java");
System.out.println(name);    //arun

StringBuffer sb=new StringBuffer("arun");
sb.append("java");
System.out.println(sb);   //arunjava

//System.out.println(sb.reverse());
System.out.println(sb.replace(4,7," Kumar Sharma"));
System.out.println(sb.indexOf("Kumar"));

StringBuilder sbb=new StringBuilder("arun");
sbb.insert(3, "abc");
System.out.println(sbb);   //aruabcn
```

=> StringTokenizer :-

-StringTokenizer is the class which is used to divide the strings into tokens

Methods :-

1. hasMoreTokens()
2. nextToken()
3. countTokens()
4. hasMoreElements()
5. nextElement()