List of Concepts Involved:

- String introduction
- Types of String
- Immutable Strings and Memory Map

Topic 1:String Introduction

String it refers to an Object in java present in package called java.lang.String.
String refers to collection of characters.

Example

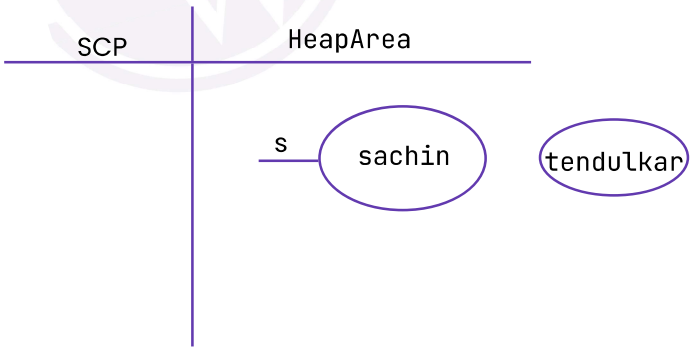
```
String s= "sachin";
System.out.println(s);//sachin

String s =new String("sachin");
System.out.println(s);//sachin
```

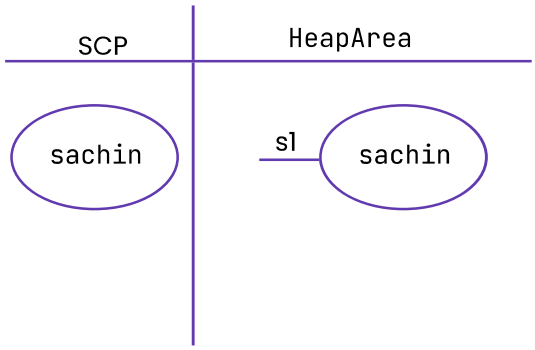
In java String object is by default immutable,meaning once the object is created we cannot change the value of the object, if we try to change then those changes will be
reflected on the new object not on the existing object.

Example

```
String s= "sachin";
s.concat("tendulkar");//(new object got created with modification so immutable)
System.out.println(s);// sachin
```

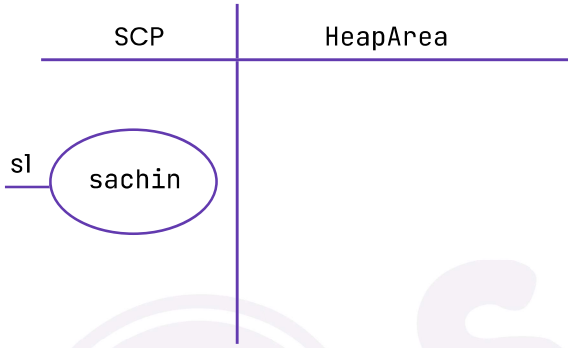


Note:
String s1 =new String("sachin");
In this case 2 objects will be created one in the heap and the other one in the String Constant Pool, the reference will always point to Heap.




```
String s1 ="sachin";
```

In this case only one object will be created in the SCP and it will be referred by our reference.



Note

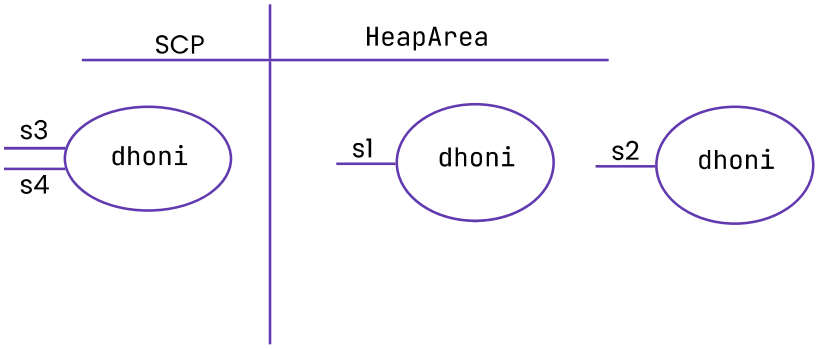
- Object creation in SCP is always optional,1st JVM will check if any object already created with required content or not.
- If it is already available then it will reuse the existing object instead of creating the new Object.
- If it is not available only then a new object will be created, so we say in SCP there is no chance of existing 2 objects with the same content.
- In SCP duplicates are not permitted.
- Garbage Collector cannot access SCP Area, Even though Object does not have any reference still object is not eligible for GC.
- All SCP objects will be destroyed only at the time of JVM ShutDown.

Example

```
String s1=new String("dhoni");
String s2=new String("dhoni");
String s3="dhoni";
String s4="dhoni";
```

Output

Two objects are created in the heap with data as "dhoni" with reference as S1,S2.
One object is created in SCP with the reference as S3,S4.



Note:

Importance of SCP

- In our program if any String object is required to use repeatedly then it is not recommended to create multiple objects with same content iit reduces performance of the system and affects memory utilisation.
- We can create only one copy and we can reuse the same object for every requirement. This approach improves performance and memory utilisation. We can achieve this by using "scp".
- In SCP several references pointing to the same object the main disadvantage in this approach is by using one reference if we are performing any change the remaining references will be impacted. To overcome this problem sun people implemented immutability concept for String objects.
- According to this once we create a String object we can't perform any changes in the existing object if we are trying to perform any changes with those changes a new String object will be created hence immutability is the main disadvantage of scp.

Types of String

In java Strings are classified into 2 types

1. Mutable String
2. Immutable String

Mutable String

Once if we create a String, on that String if we try to perform any operation and if those changes get reflected in the same object then such strings are called “Mutable String”.

Example: StringBuffer, StringBuilder

Immutable String

Once if we create a String, on that String if we try to perform any operation then those changes won’t be reflected in the same object, rather a new object will be created.Such type of String is called as “Immutable String”.

Example: Strin

String class Constructor

- String s =new String()
  Creates an Empty String Object

- String s =new String(String literals)
  Creates an Object with String literals on Heap

- String s =new String(StringBuffer sb)
  Creates an equivalent String object for StringBuffer

- String s =new String(char[] ch)
  Creates an equivalent String object for character array

- String s =new String(byte[] b)
  Creates an equivalent String object for byte array

Example

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

Example

```
byte[] b={100,101,102};
String s=new String(b);
System.out.println(s)//def
```