Note:: Object creation in SCP is always optional,1st jvm will check is 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 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.

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

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

case 4:: String s = new String("sachin");
s.concat("tendulkar");
s=s.concat("IND");
s="sachintendulkar";

Output:: Direct literals are always placed in SCP,Because of runtime operation if object is required to create compulsorily that object
should be placed on the Heap,but not on SCP.

Q>
```
String s1= new String("sachin");
s1.concat("tendulkar");
s1+="IND";
String s2=s1.concat("MI");
System.out.println(s1);
System.out.println(s2);
```
Find out totally how many objects are created and how many are eligible for garbage collection?
output: 8 and gc eligible is 2

Q>
```
String s1=new String("you cannot change me!")
String s2=new String("you cannot change me!");
System.out.println(s1==s2);

String s3="you cannot change me!";
System.out.println(s1==s3);
String s4="you cannot change me!";
System.out.println(s3==s4);

String s5="you cannot " + "change me!";
System.out.println(s3==s5);

String s6="you cannot ";
String s7=s6+"change me!";
System.out.println(s3==s7);
```

```
final String s8="you cannot ";
String s9=s8+"change me!";
System.out.println(s3==s9);
System.out.println(s6==s8);
```

output
false
false
true
true
false
true
true

case7 :: Interning=> Using Heap object reference, if we want to get Corresponding SCP Object, then we need to use intern() method.

eg1::

```
        String s1 =new String("sachin");// One in heap(s1) and the other one in SCP
        String s2=s1.intern();//using s1 access object in SCP which has no reference
        System.out.println(s1==s2);//false
        String s3="sachin";
        System.out.println(s2==s3);//true
```

eg2::
Using heap object reference, if we want to get the corresponding SCP object and if the Object does not exists,
then intern() will create a new object in SCP and it returns.

eg2::

```
        String s1=new String("sachin");// One in heap(s1) and the other one in SCP
        String s2=s1.concat("IND");// One in SCP(IND) and the other one in heap(s2)
        String s3=s2.intern();
        String s4="sachinIND";
    System.out.println(s3 == s4);//true
```

Note:
Importance of SCP
==========
1. In our program if any String object is required to use repeatedly then it is not recommended to create multiple object with same content
it reduces performance of the system and effects memory utilization.
2. We can create only one copy and we can reuse the same object for every requirement. This approach improves performance and memory
utilization we can achieve this by using "scp".
3. In SCP several references pointing to 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.
4. According to this once we creates 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.

Note: if the class name and method name is same in a class then that method is called "Constructor".

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
    eg: String str = new String("sachin");
  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

eg:
```
char[] ch = {'j','a','v','a'};
String s1 =new String(ch);
System.out.println(s1);//java

System.out.println();

byte[] b = {65,66,67,68};
String s2= new String(b);
System.out.println(s2);//"ABCD"

StringBuffer sb =new StringBuffer("sachin");
System.out.println("StringBuffer data is :: "+sb);

String s1 =new String(sb);
System.out.println("String data is : "+s1);
```

Important methods of String
==========
1.public char charAt(int index)
2.public String concat(String str)
3.public boolean equals(Object o)
4.public boolean equalsIgnoreCase(String s)
5.public String subString(int begin)
6.public String subString(int begin,int end)
7.public int length()
8.public String replace(char old,char new)
9.public String toLowerCase()
10.public String toUpperCase()
11.public String trim()
12.public int indexOf(char ch)
13.public int lastIndexOf(char ch)

Note:
```
package java.lang;
class String
{
      //method name
      public int length(){

      }
}
```

Integer Array Class(proxy class)
==========
class [I
{
	//property name
	int length;
}

eg:
String s ="sachin";
System.out.println(s.length());//6

int[] arr= {10,20,30};
System.out.println(arr.length);//3