Case3:

```
  String s =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.

vs

```
String s ="sachin";
```

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

Note:
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#1.

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

refer diagram to understand the memory map.

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.

```
case 5:: String s1= new String("sachin");
                s1.concat("tendulkar");
                s1+="IND";
            String s2=s1.concat("MI");
            System.out.println(s1);
            System.out.println(s2);
```

How many objects are eligible for GC?

```
total      :: 8 objects
GC Eligible:: 2 objects
```

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

Q>
public class Test {
    public static void main(String[] args) {
        final String fName = "James";
        String lName = "Gosling";
        String name1 = fName + lName;
        String name2 = fName + "Gosling";
        String name3 = "James" + "Gosling";
        System.out.println(name1 == name2);
        System.out.println(name2 == name3);
    }
}
What will be the result of compiling and executing Test class?
A. true
   true
B. true
   false
C. false
   false
D. false
   true


Q> System.out.print(""=="");//true
   System.out.print(" ");//
   System.out.print("A"=="A");//true
   System.out.print("a==A");//a==A
```

```
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
SUNMS 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.




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

      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(s1 == s3);//false
         System.out.println(s2 == s3);//false
       System.out.println(s3 == s4);//true

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
```

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


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


Q>
public class DemoApp{
      public static void main(String... args){
            if(args[0].equals("hello") ? false : true)
                  System.out.println("success");
            else
                  System.out.println("failure");
      }
}
What is the output if the program is executed in the following style?
 java DemoApp hello
      |
      DemoApp.main(new String[]{"hello"})

A. success
B. failure
C. CE
D. ArrayIndexOutOfBoundsException
E. StringIndexOutOfBoundsException

Answer: B

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)

Important methods of String
===========================
Note: Even though String data is stored internally as "Arrays", as a programmer we
can't access the data at the index level directly.
      we need to use methods.
  1.public char charAt(int index)
              eg:: String s="sachin";
                 System.out.print(s.charAt(0));//s
System.out.print(s.charAt(-1));//StringArrayIndexOutOfBoundsException

System.out.print(s.charAt(10));//StringArrayIndexOutOfBoundsException

  2.public String concat(String str)
              eg:: String s="sachin";
                     System.out.println(s.concat("tendulkar"));
                s+="IND";
                s=s+"MI";
                System.out.print(s);

  3.public boolean equals(Object o)
             It is used for Content Comparison,In String class equals() method is
Overriden to check the content of the object

  4.public String subString(int begin)
        It gives the String from the begin index to end of the String.
             String s="Ineeuron";
           System.out.print(s.substring(2));//searching from 2 to end of the string

  5.public String subString(int begin,int end)
        It gives the String from the begin index to end-1 of the String.
             String s="Ineeuron";
           System.out.print(s.substring(2,6));//searching from 2 to 5 will happen


  6.public int length()
             It returns the no of characters present in the String.
             String s="pwskills";
             System.out.print(s.length());//8
             System.out.print(s.length);//Compile time error

  8.public String replace(char old,char new)
             String s="ababab";
           System.out.print(s.replace('a','b')); //bbbbbb

  9.public String toLowerCase()
  10.public String toUpperCase()


Predict the output
===================
Q>
String s1="sachin";
String s2=s1.toUpperCase();
String s3=s1.toLowerCase();
System.out.print(s1==s2);
System.out.print(s1==s3);

Q>
String s1="sachin";
String s2=s1.toString();
System.out.print(s1==s2);


Q>
String s1=new String("sachin");
String s2=s1.toString();
String s3=s1.toUpperCase();
String s4=s1.toLowerCase();
String s5=s1.toUpperCase();
String s6=s1.toLowerCase();
System.out.print(s1==s6);
System.out.print(s3==s5);
```

Q>

```
String string = "string".replace('i', '0'); str0ng
System.out.println(string.substring(2, 5));
```

Q>

```
String s1 = new String("JAVA");
String s2 = new String("JAVA");
System.out.println(s1 == s2);
System.out.println(s1.equals(s2));
System.out.println(s1 == s2.intern());
System.out.println(s1.intern() == s2.intern());
System.out.println(s1.intern() == s2);
```