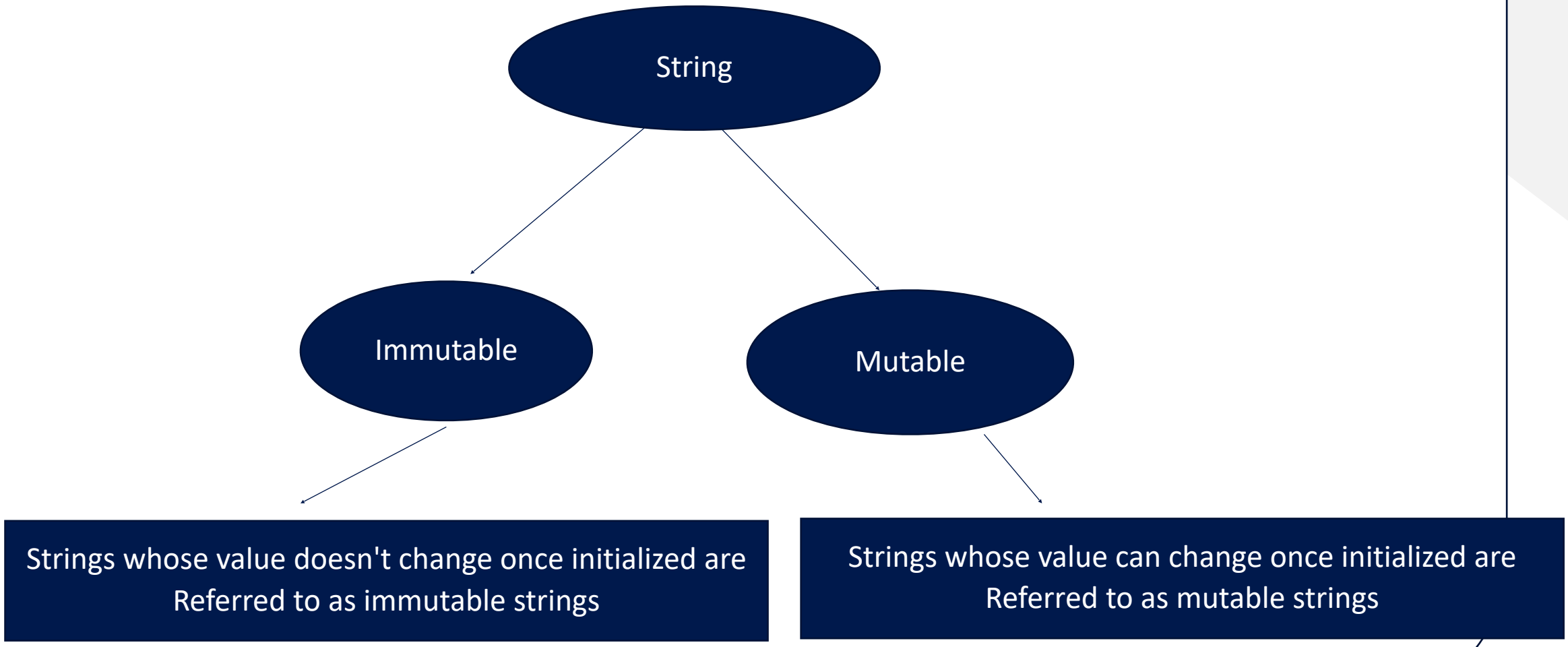
JAVA Certification

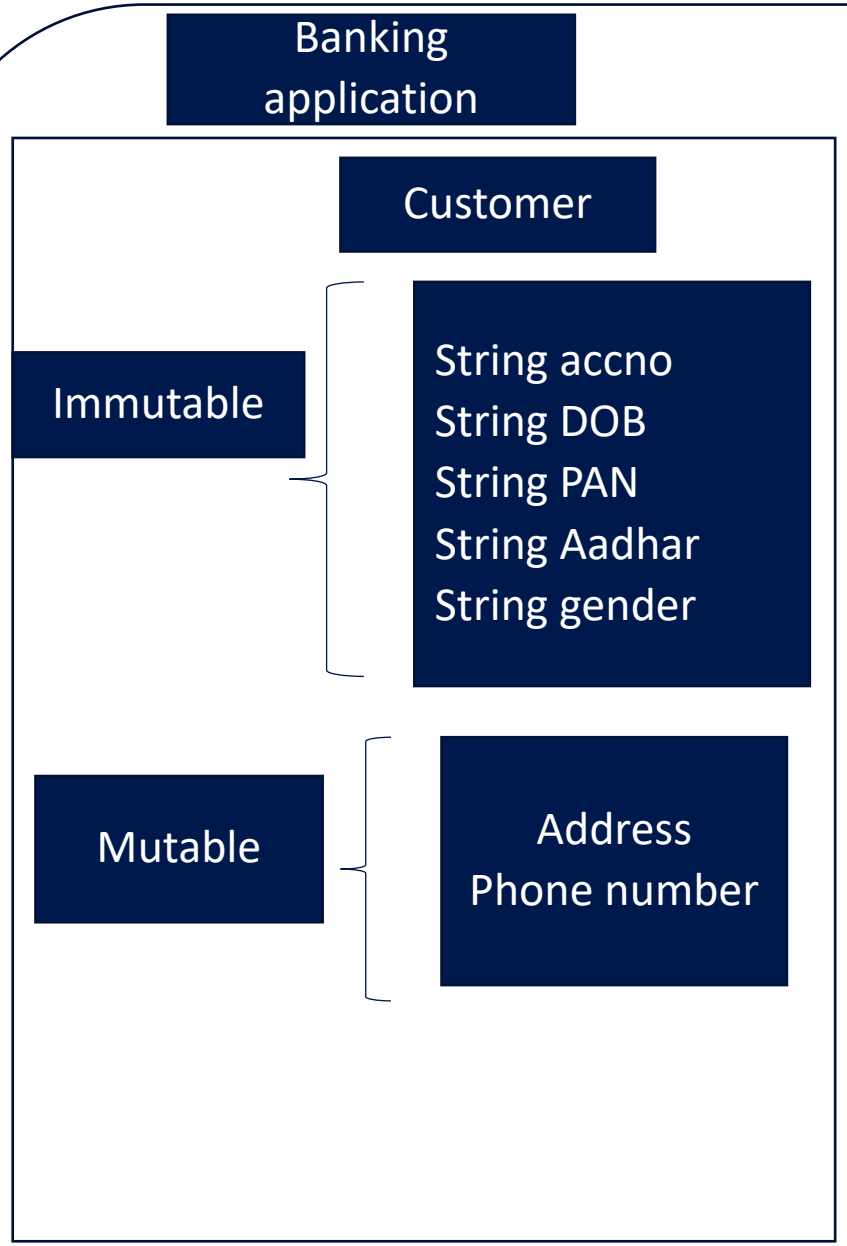
Strings:

String as a object that represent the sequence of characters.

```
public class sookshmas1
{
 public static void main(String args[])
{
String s1="sookshmas";
  System.out.println(s1);

char ch[]={'s','o','0','k'};
String s2=new String( ch );//converting char array to string
System.out.println(s2);

String s3=new String("java");//creating java string by new keyword
  System.out.println(s3);

 }
}
```

Java supports 2 types of strings





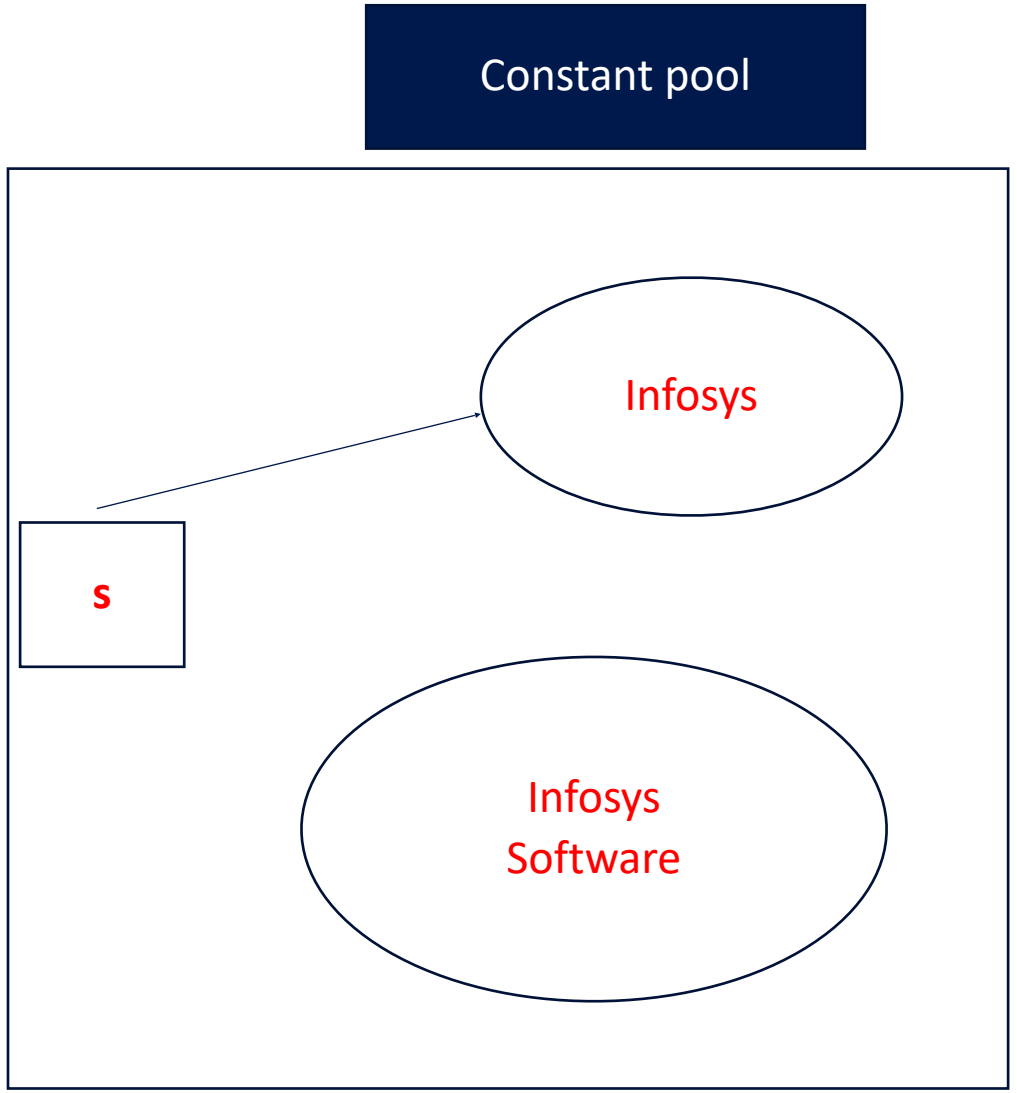
In design phase of an application we decide the properties of an object and at the same time also decide upon the values the property should be holding.

For all those properties whose value would not be changing through out the life time of the object would be considered as immutable string.

Those property whose value frequently changing would be mark as mutable.

Strings are immutable in java:

```
public class Immutable {

   public static void main(String args[]) {
      String s = "Infosys";

      s.concat(" Software");

      System.out.println(s);

   }
}
```



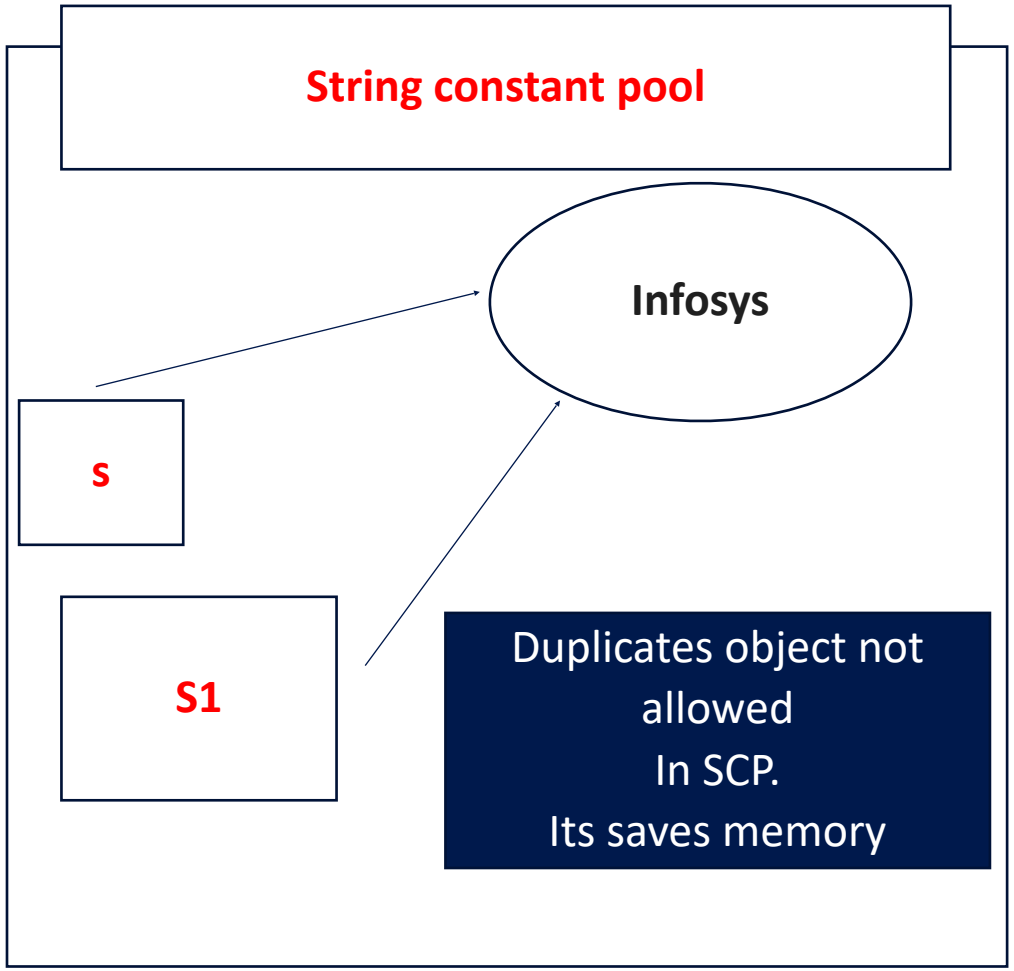
Comparison of strings

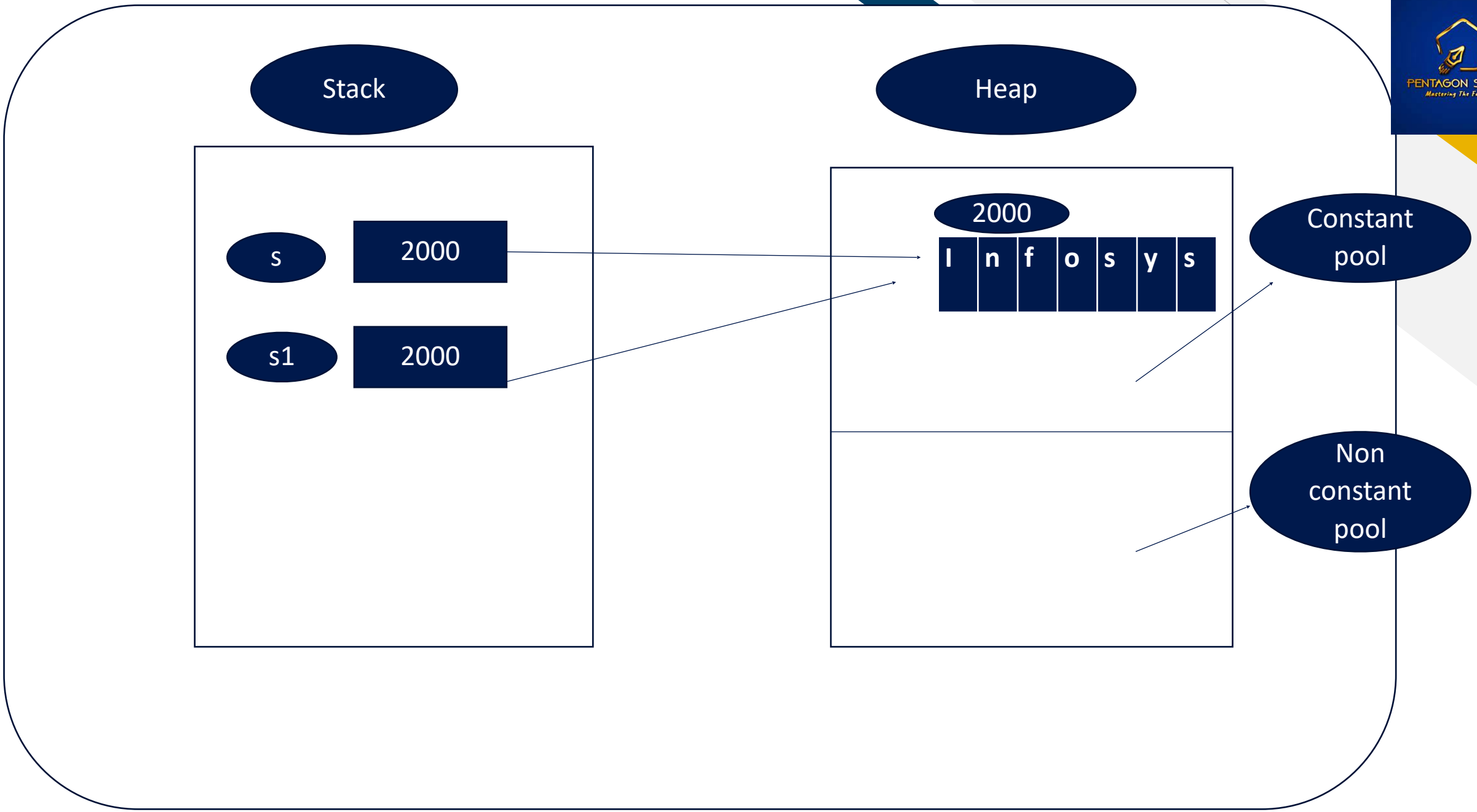
==
equals()
compareTo()

For every String constant one object will be created in SCP

A string constant pool is a separate place in the heap memory

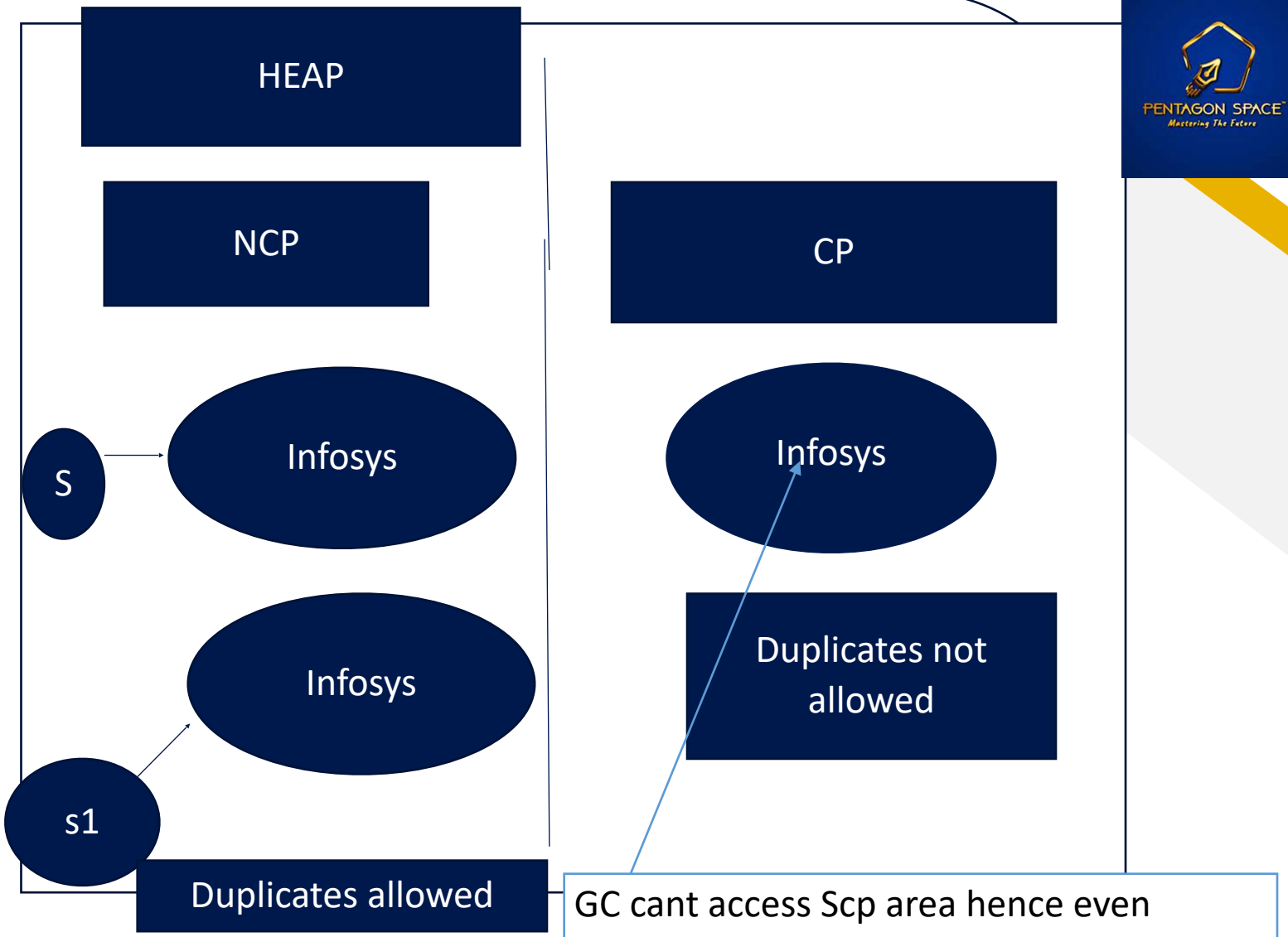
```
public class Immutable {

   public static void main(String args[]) {
      String s = "Infosys";

      String s1 = "Infosys";

      System.out.println(s + " " + s1);
   }
}
```



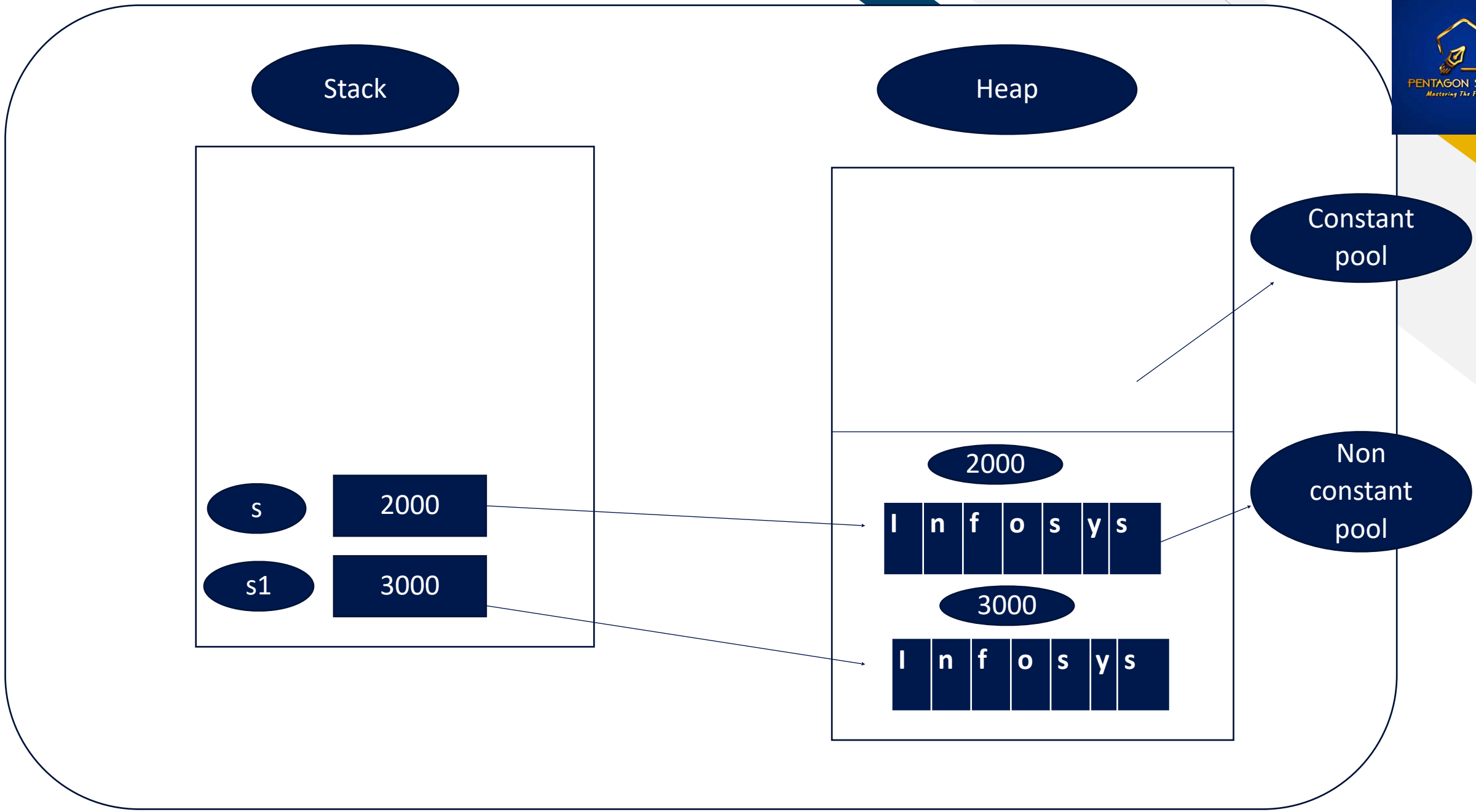

Stack
Heap
s
2000
s1
2000
2000
I n f o s y s
Constant pool
Non constant pool

```
public class Immutable {

    public static void main(String args[]) {
        String s = "Infosys";

        String s1 = "Infosys";

        if (s == s1) {
            System.out.println("ref are equal");
        } else {
            System.out.println(" ref are unequal");
        }
        if (s.equals(s1)) {
            System.out.println("equal");
        } else {
            System.out.println("unequal");

        }

    }
}
```

```
public class sookshmas2
{

   public static void main(String...k)
   {
    String s = new String("Infosys");

      String s1 = new String("Infosys");

      System.out.println(s + " " + s1);
}
}
```

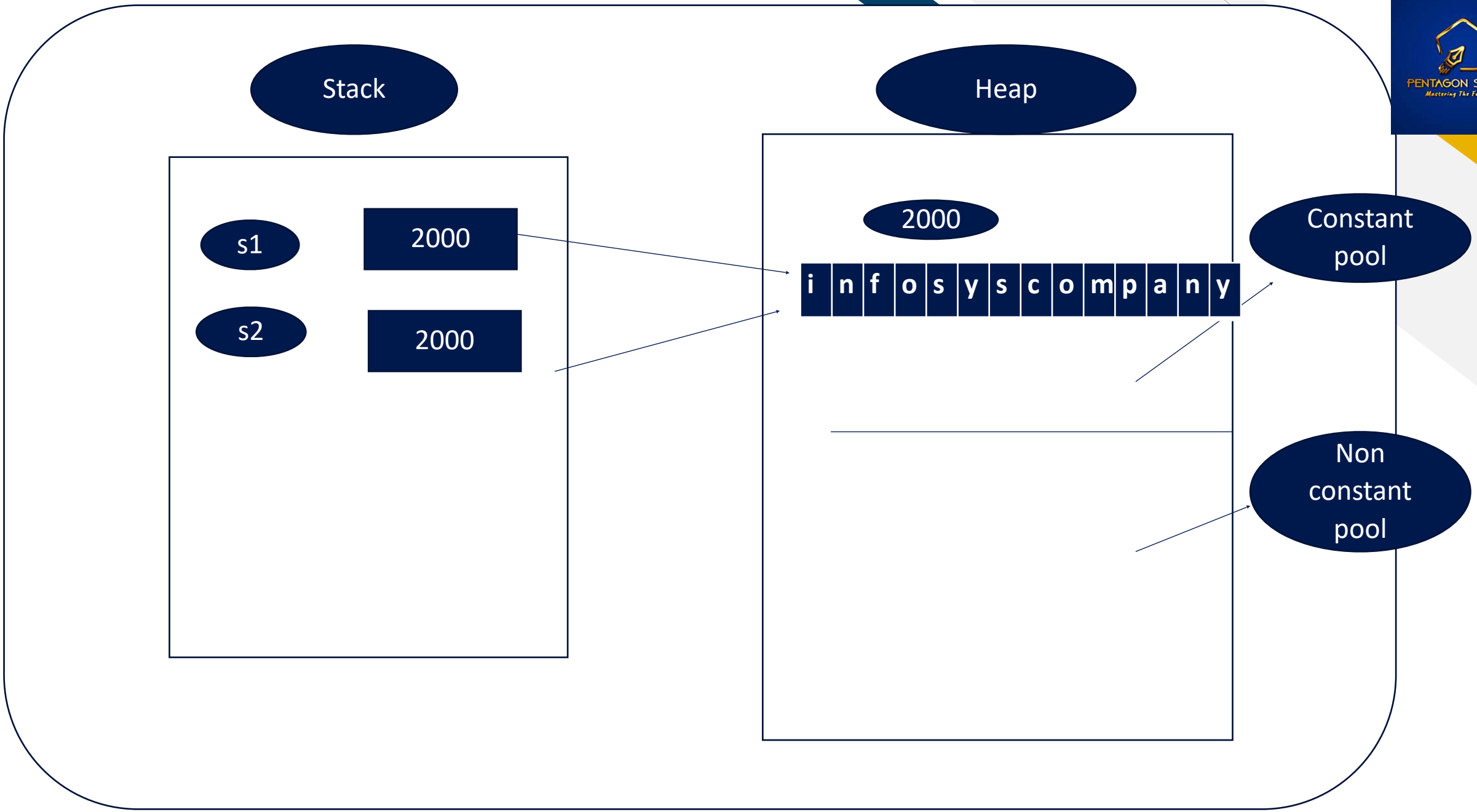


GC cant access Scp area hence even though objects doesn’t have any ref still that object is not eligible for GC
All scp object wil destroy at the time of jvm shutdown automatically

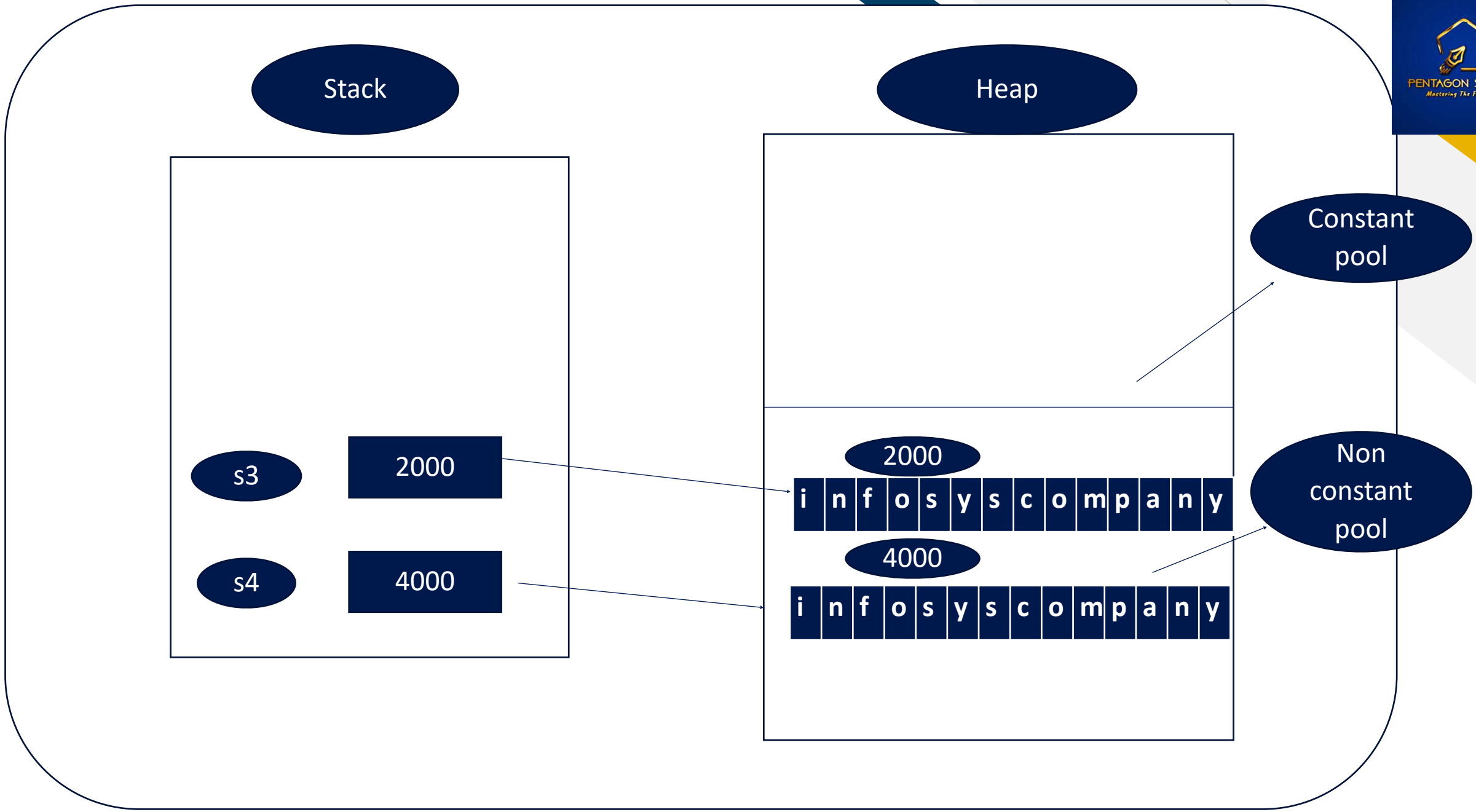

Stack
Heap
Constant pool
Non constant pool
s
2000
s1
3000
2000
I n f o s y s
3000
I n f o s y s

```
public class Immutable {

    public static void main(String args[]) {
        String s = new String("Infosys");

        String s1 = new String("Infosys");

            if (s == s1) {
           System.out.println("ref are equal");
        } else {
           System.out.println(" ref are unequal");
        }
        if (s.equals(s1)) {
           System.out.println("equal");
        } else {
           System.out.println("unequal");

        }

    }
}
```

```
public class Immutable {

    public static void main(String args[]) {

        String s1, s2;

        s1 = "infosys" + " company";
        System.out.println(s1);
        s2 = "infosys" + " company";
        System.out.println(s2);
        if (s1 == s2) {
            System.out.println(" s1 and s2 are pointing to the same object");
        } else {
            System.out.println(" s1 and s2 are not pointing to the same object");
        }

    }
}
```


Stack
Heap
s1
2000
s2
2000
2000
i n f o s y s c o m p a n y
Constant pool
Non constant pool

```
public class Immutable {

   public static void main(String args[]) {

      String s1, s2;

      s1 = "infosys";

      s2 = " company";

      String s3, s4;

      s3 = s1 + s2;
      s4 = s1 + s2;
      if (s3 == s4) {
         System.out.println(" s3 and s4 are pointing to the same object");
      } else {
         System.out.println(" s3 and s4 are not pointing to the same object");
      }

   }
}
```

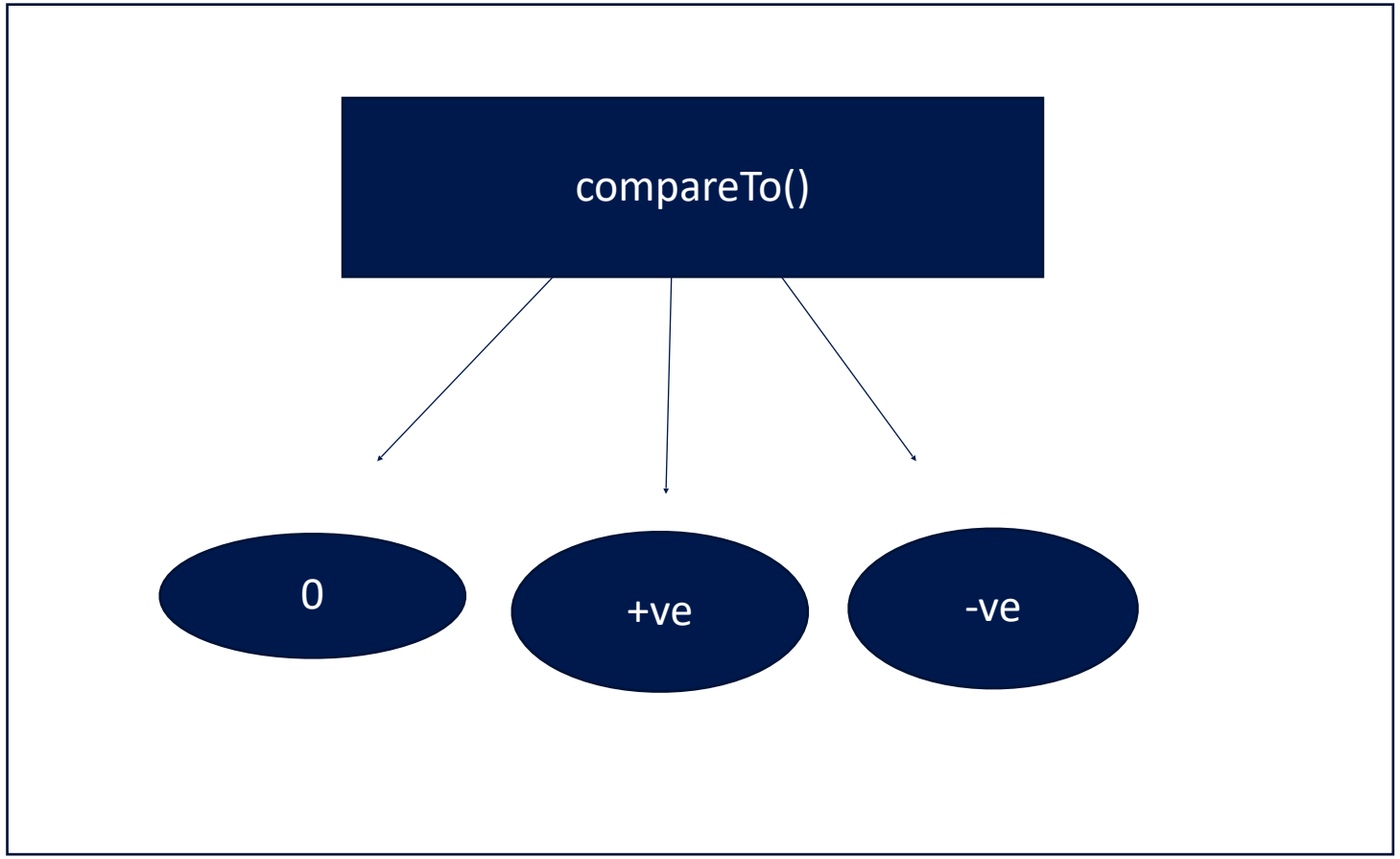

Stack
Heap
Constant pool
Non constant pool
s3
2000
s4
4000
2000
i n f o s y s c o m p a n y
4000
i n f o s y s c o m p a n y

Note:

1.value+value----→ Constant pool

2.Variable +variable ---→ non constant pool

3.value+variable---→non constant pool

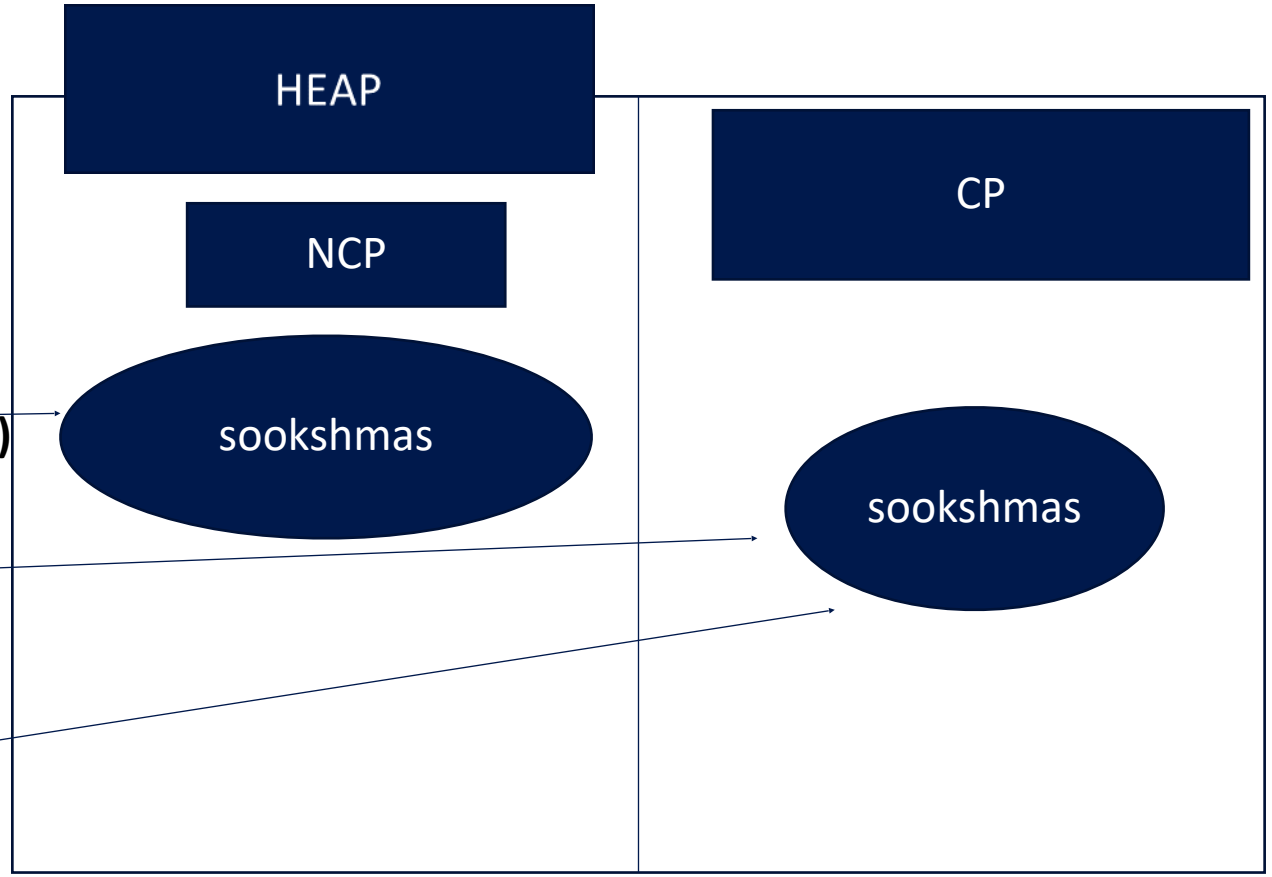

compareTo()
0
+ve
-ve

```
public class Demo {

   public static void main(String args[]) {

      String s1="karthik";
      String s2="karthik";

      if(s1.compareTo(s2)<0){
         System.out.println(" string s1 is lesser than string s2");
      }

      else if(s1.compareTo(s2)>0){
         System.out.println(" string s1 is greater than string s2");
      }
      else
      {
         System.out.println(" strings are equal");
      }

   }
}
```

```
public class Demo {

    public static void main(String args[]) {

        String s1 = "ZBCD";
        String s2 = "ZBCD";
        String s3 = "Karthik";
        System.out.println(s1.compareTo(s2));
        //0
        System.out.println(s1.compareTo(s3));
     //(because s1>s3)

        System.out.println(s3.compareTo(s1));
        //(because s3 < s1 )

    }
}
```

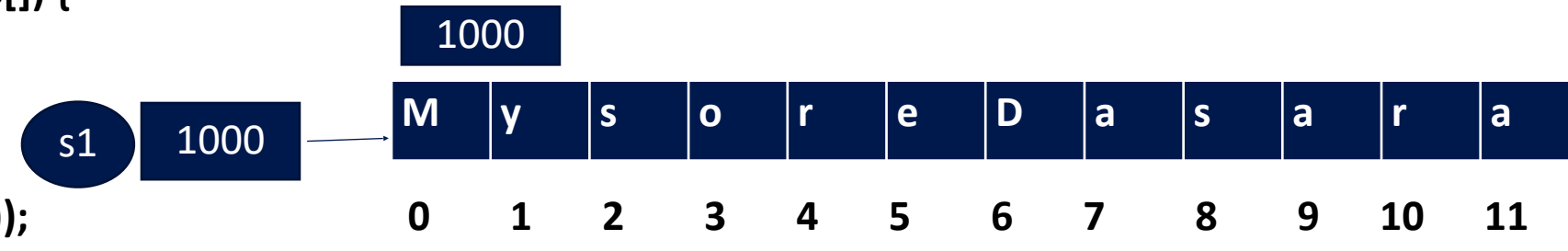
Intern()

```
public class sookshmas7
{
    public static void main(String...k)
   {

      String s1=new String("sookshmas");

      String s2=s1.intern();

      System.out.println(s2==s1);

      String s3="sookshmas";

      System.out.println(s2==s3);
}

}
```



Intern(): it is used to create copy of the string which is in the non constant pool region to constant pool region.

```
public class Demo {

    public static void main(String args[]) {

        String s1 = "MysoreDasara";
        System.out.println(s1.length());
        System.out.println(s1.charAt(2));
        System.out.println(s1.indexOf("s"));
        System.out.println(s1.lastIndexOf("s"));
        System.out.println(s1.toUpperCase());
        System.out.println(s1.toLowerCase());
        System.out.println(s1.substring(7));
        System.out.println(s1.substring(7, 11));

    }
}
```



```
public class Demo {

   public static void main(String args[]) {

      String s1 = "rama";
      String s2 = "Rama";

      if (s1.equalsIgnoreCase(s2)) {
         System.out.println(" s1 and s2 are equal");
      } else {
         System.out.println(" s1 and s2 are not equal");
      }

   }
}
```

```
public class Demo {

   public static void main(String args[]) {

      StringTokenizer s = new StringTokenizer("mysore  dasara", " ");
      while (s.hasMoreTokens()) {
         System.out.println(s.nextToken());
      }

   }
}
```

C strings	Java Strings
1.In c strings are considered as collection of character. 2. C strings end with null character. 3.C strings by default are mutable. 4. C strings are not secure.	1.In java strings are considered as objects. 2.Java strings doesn't end with null character. 3. Java Strings by default immutable 4. Java Strings are secure.

Mutable string

all those strings whose value can be changed even after initialization is referred to as mutable string

In java we can create mutable string using the following classes

1.StringBuilder
2.StringBuffer

String:

1. If the content will change frequently then it is not recommended.
2. Because every changes new objected will be created.

StringBuffer:

1.All the changes will be performed in existing object. So no need to create a new object.

Methods:

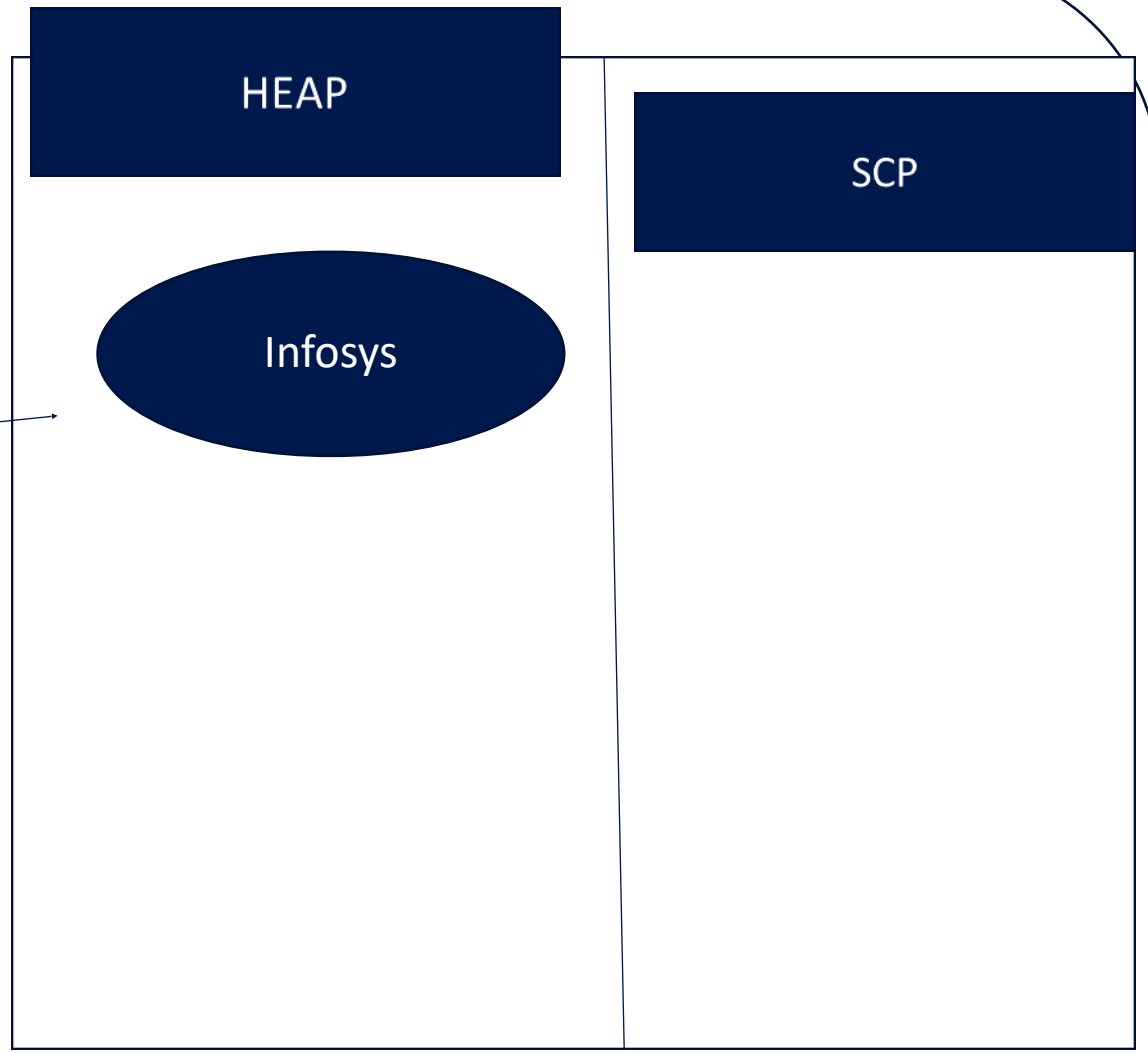
public StringBuffer append(String data)

```
public class sookshmas7
{
    public static void main(String...k)
   {
StringBuffer s1 = new StringBuffer("Infosys");
      StringBuffer s2 = s1.append(" Software ");
      StringBuffer s3 = s2.append(" Bangalore");
      System.out.println(s1);
      System.out.println(s2);
      System.out.println(s3);
```

Final result for s1,s2,s3

```
}

}
```

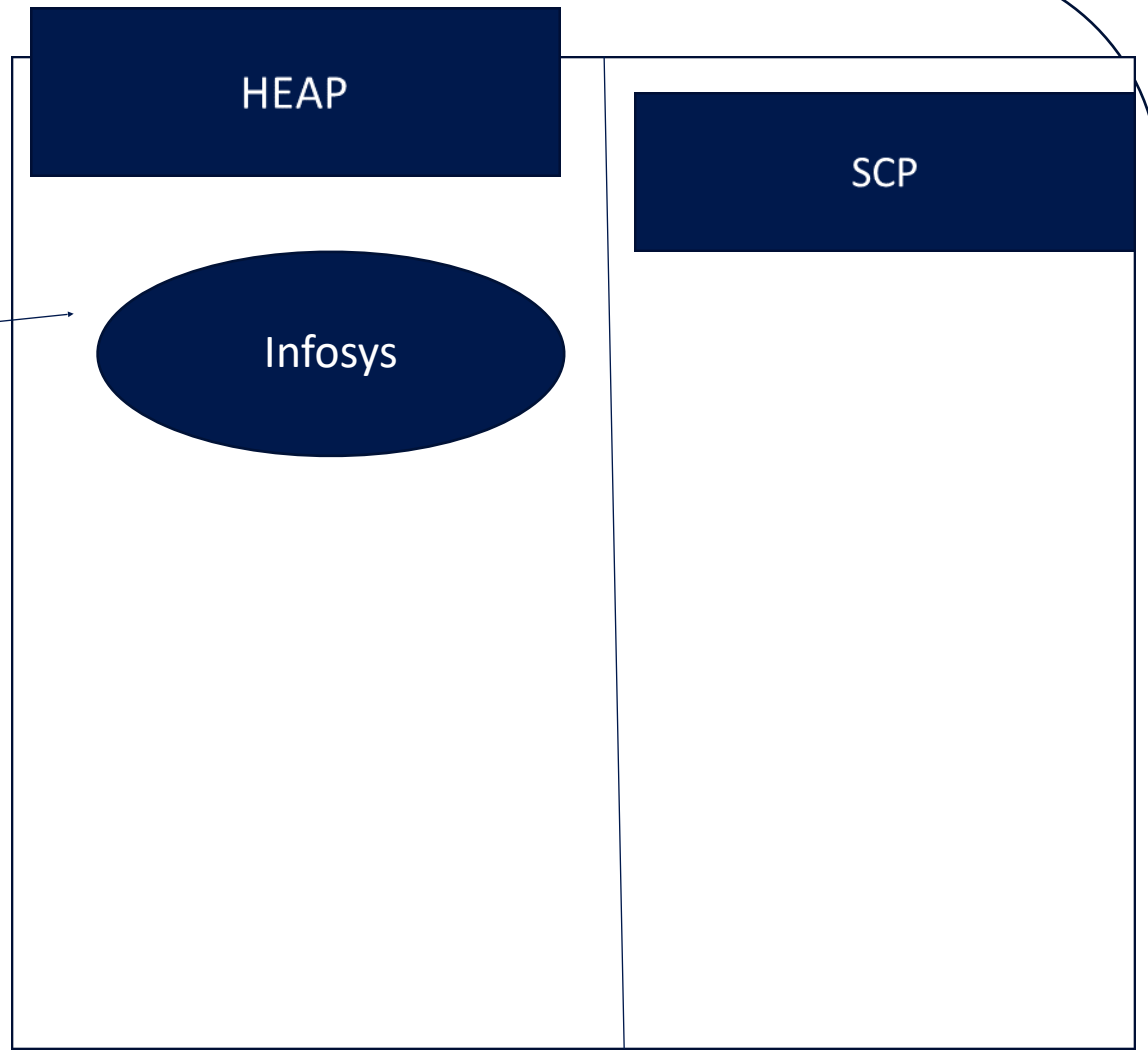
Infosys Software Bangalore



```
public class sookshmas7
{
     public static void main(String...k)
   {
StringBuilder s1=new StringBuilder(“ Infosys ");

StringBuilder s2=s1.append(“ Software ");

StringBuilder s3=s2.append(“ Bangalore");
System.out.println(s1);
System.out.println(s2);
System.out.println(s3);

}

}
```

Final result for s1,s2,s3

Infosys Software Bangalore



```
public class Demo {

    public static void main(String args[]) {

       StringBuilder s=new StringBuilder();
        System.out.println(s.capacity());
        System.out.println(s.length());
        s.append("sachin");
        System.out.println(s.capacity());
        System.out.println(s.length());
        s.append(" is a great batsman");
        System.out.println(s.capacity());
        System.out.println(s.length());
        s.append("He is from india");
        System.out.println(s.capacity());
        System.out.println(s.length());

    }

}
```

The initial capacity of a mutable string would be 16. Once the capacity is fully occupied memory would be expanded using the following formula

ExistingCapacity*2+2

16*2+2=34

```
public class Demo {

    public static void main(String args[]) {

        StringBuilder sb = new StringBuilder(5);
        System.out.println(sb.capacity());
        sb.append("sachin");
        System.out.println(sb.capacity());

    }

}
```

```
public class Demo {

    public static void main(String args[]) {

        StringBuffer sb = new
StringBuffer(5);
        System.out.println(sb.capacity());
        sb.append("sachin");
        System.out.println(sb.capacity());

    }

}
```

3.public StringBuffer(String str)
It create StringBuffer object with the specified data .

Newcapacity = InitialCapacity + data_Length;

```
public class Demo {

   public static void main(String args[]) {

       StringBuffer sb=new StringBuffer("xyz");
      System.out.println(sb);
      System.out.println(sb.capacity());


   }

}
```

```
public class sookshmas7
{
     public static void main(String...k)
   {
  StringBuffer sb = new StringBuffer();
       sb.ensureCapacity(100);
       System.out.println(sb.capacity());
       sb.append("Infosys");
       sb.trimToSize();
       System.out.println(sb.capacity());
}

}
```

To release extra allocated memory

StringBuffer:

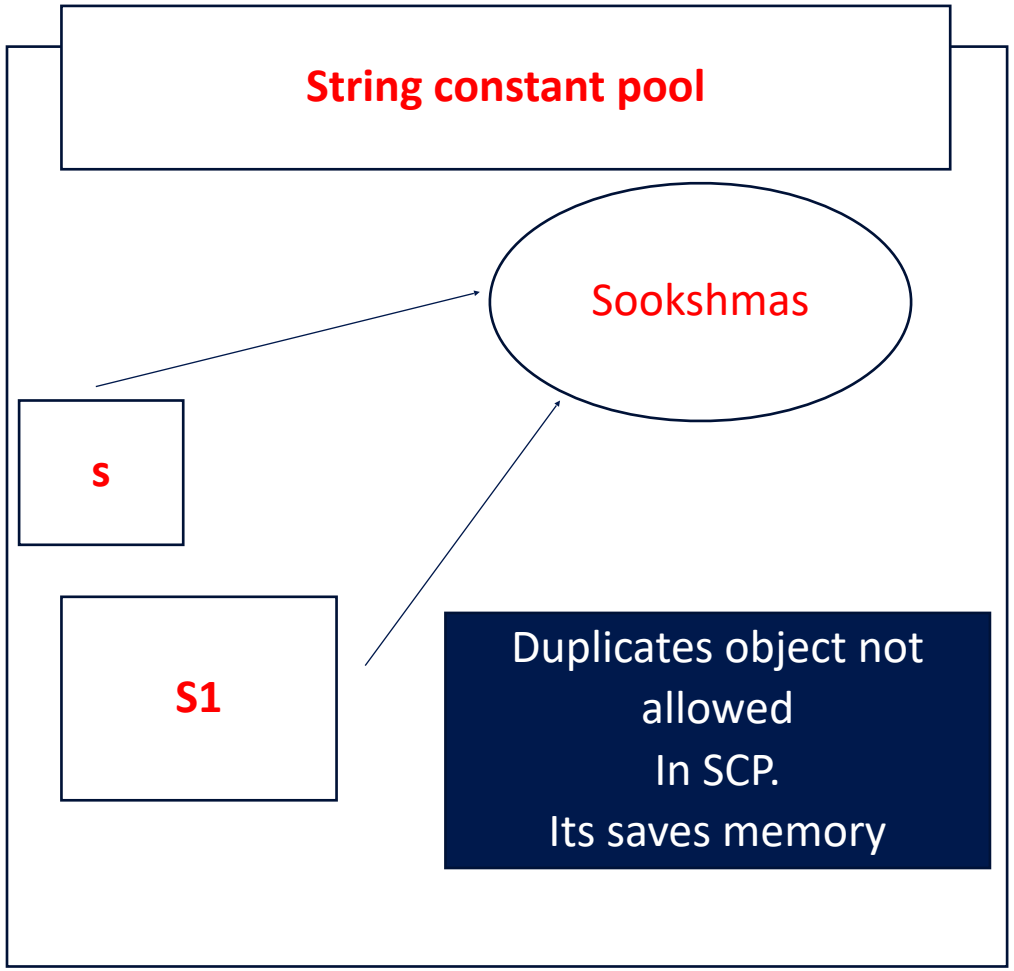
1. Every method is Synchronized.
2. StringBuffer object is threadsafe
3. Low performance
4. Introduced 1.0 version

StringBuilder:

1. No method is Synchronized.
2. Not thread safe.
3. High performance
4. Introduced in 1.5 v

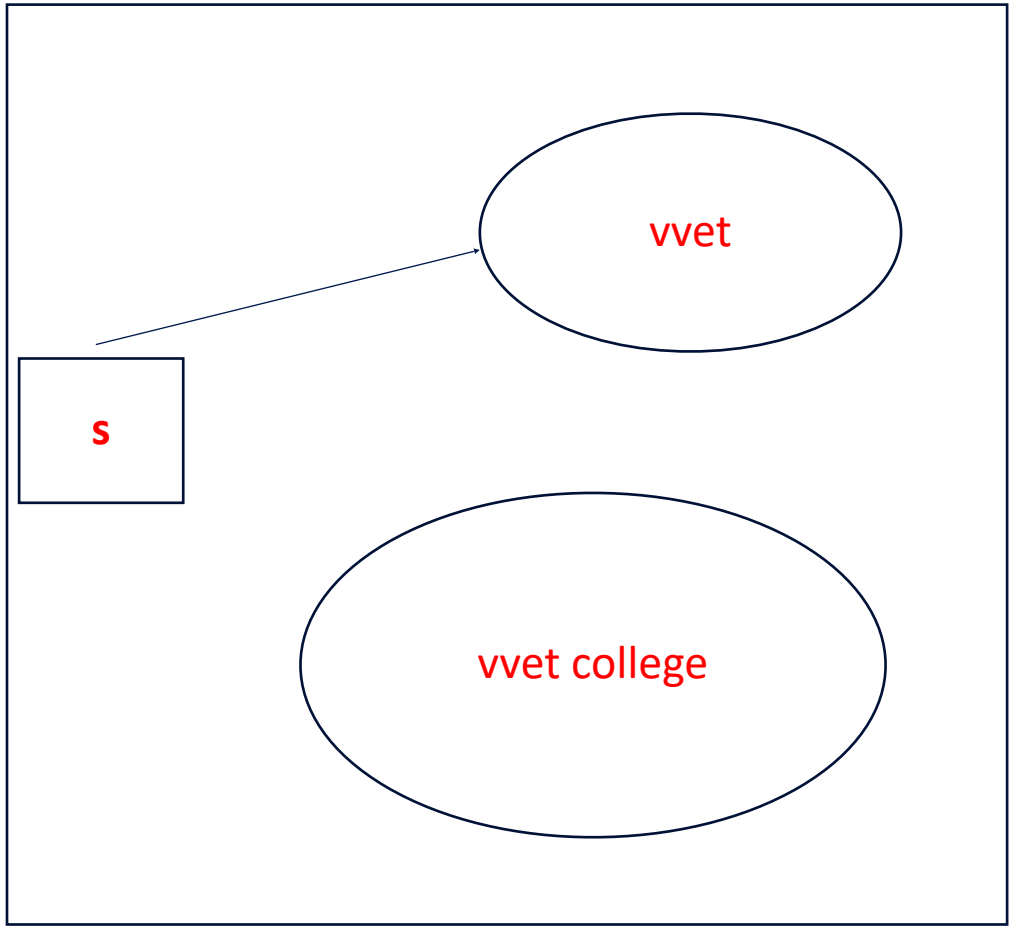
A string constant pool is a separate place in the heap memory

```
public class immutable {


 public static void main(String args[])
{
  String s="Sookshmas";

  String s1=“Sookshmas”;

  System.out.println(s +” “+s1);
 }
}
```



Strings are immutable in java:

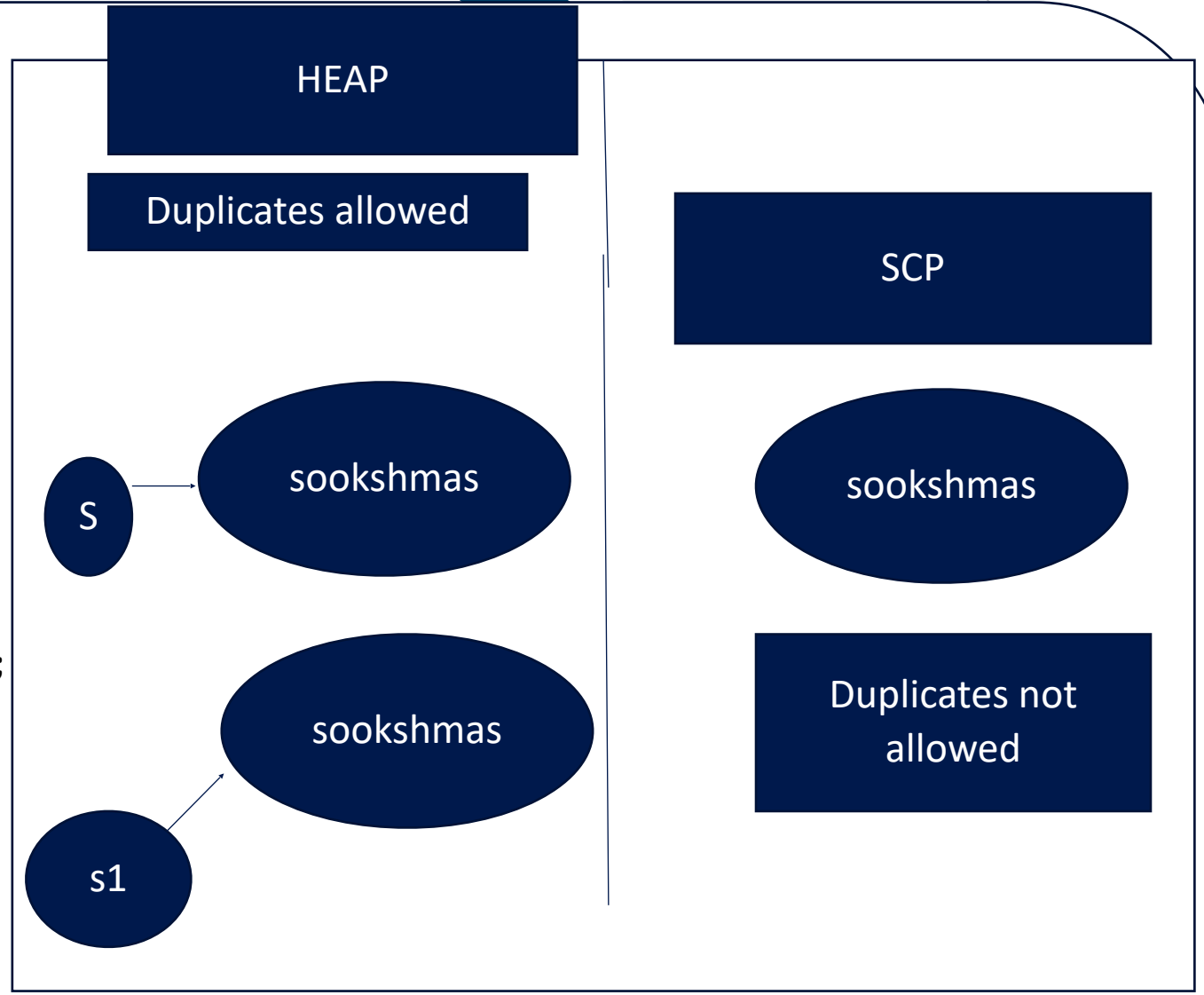
```
public class immutable {

 public static void main(String args[])
{


   String s1=new String(" vvet");

   s1.concat("college");

    System.out.println(s1);
}
}
```



1.For every String constant one object will be created in SCP

2.For runtime operation if any object require to create that object will be created in heap area.

```
public class sookshmas2
{

   public static void main(String...k)
   {
      String s=new String("sookshmas");

     String  s1=new String("sookshmas");

      System.out.println(s +" "+ s1);
   }
}
```



String:

1. If the content will change frequently then it is not recommended.
2. Because every changes new objected will be created.

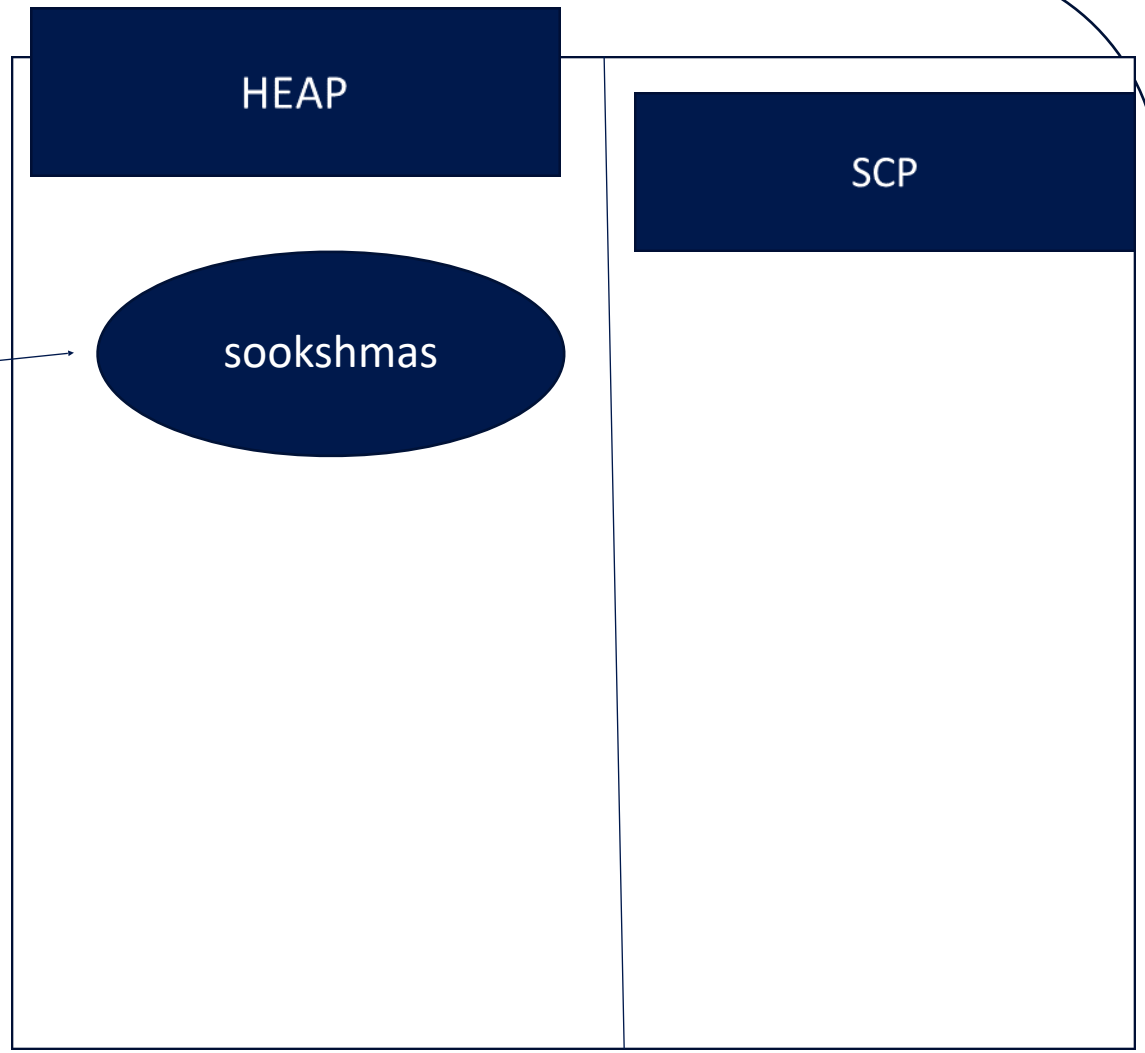
StringBuffer:

1.All the changes will be performed in existing object. So no need to create a new object.

Methods:

public StringBuffer append(String data)

```
public class sookshmas7
{
    public static void main(String...k)
   {
   StringBuffer s1=new StringBuffer("sookshmas ");

StringBuffer s2=s1.append("Software ");

StringBuffer s3=s2.append("bangalore");
System.out.println(s1);
System.out.println(s2);
System.out.println(s3);



}

}
```



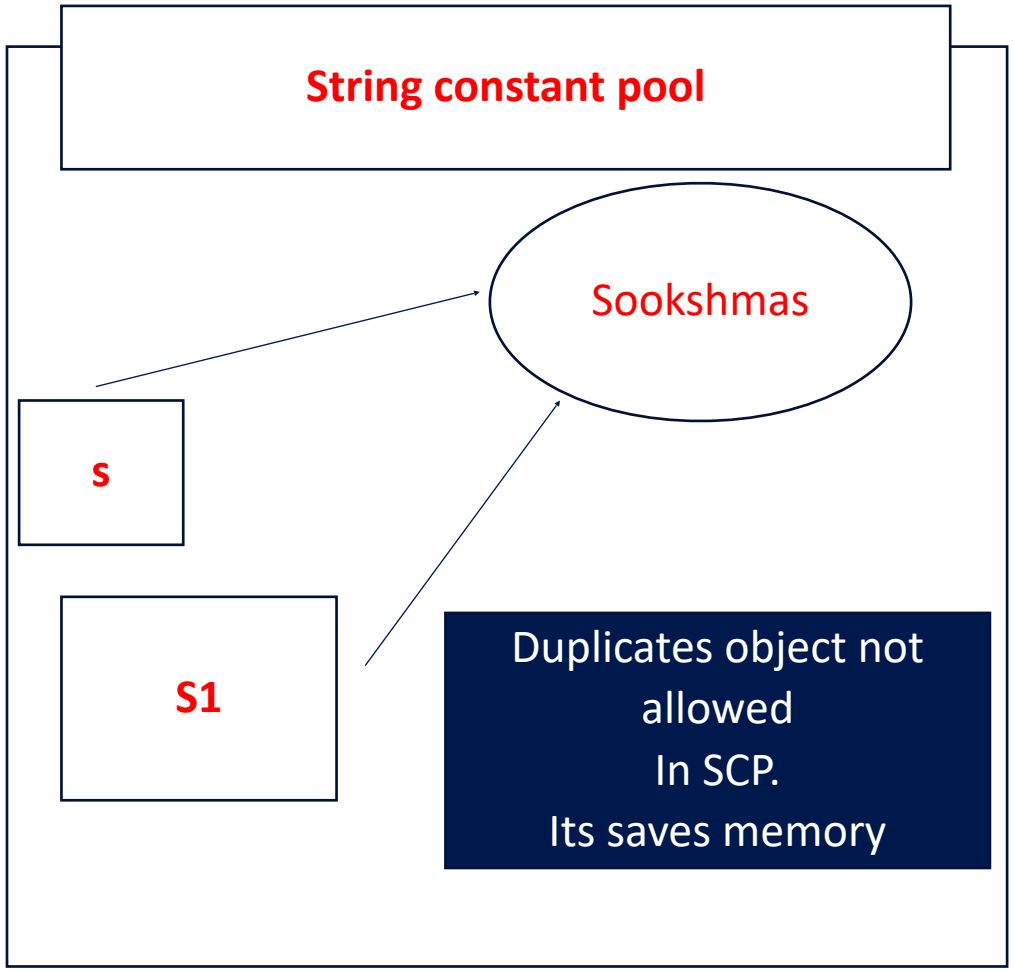
1. == Reference comparison.
2. equals() for content comparison.

```
public class immutable
 {

 public static void main(String args[])
{
   String s="Sookshmas";

 String s1="Sookshmas";

 if(s==s1)
 {
    System.out.println("ref are equal");
 }

 else
 {
    System.out.println(" ref are unequal");
 }
```

```
if( s. equals(s1))
 {
   System.out.println("equal");
 }
 else
 {
    System.out.println("unequal");

 }

   }
}
```

A string constant pool is a separate place in the heap memory

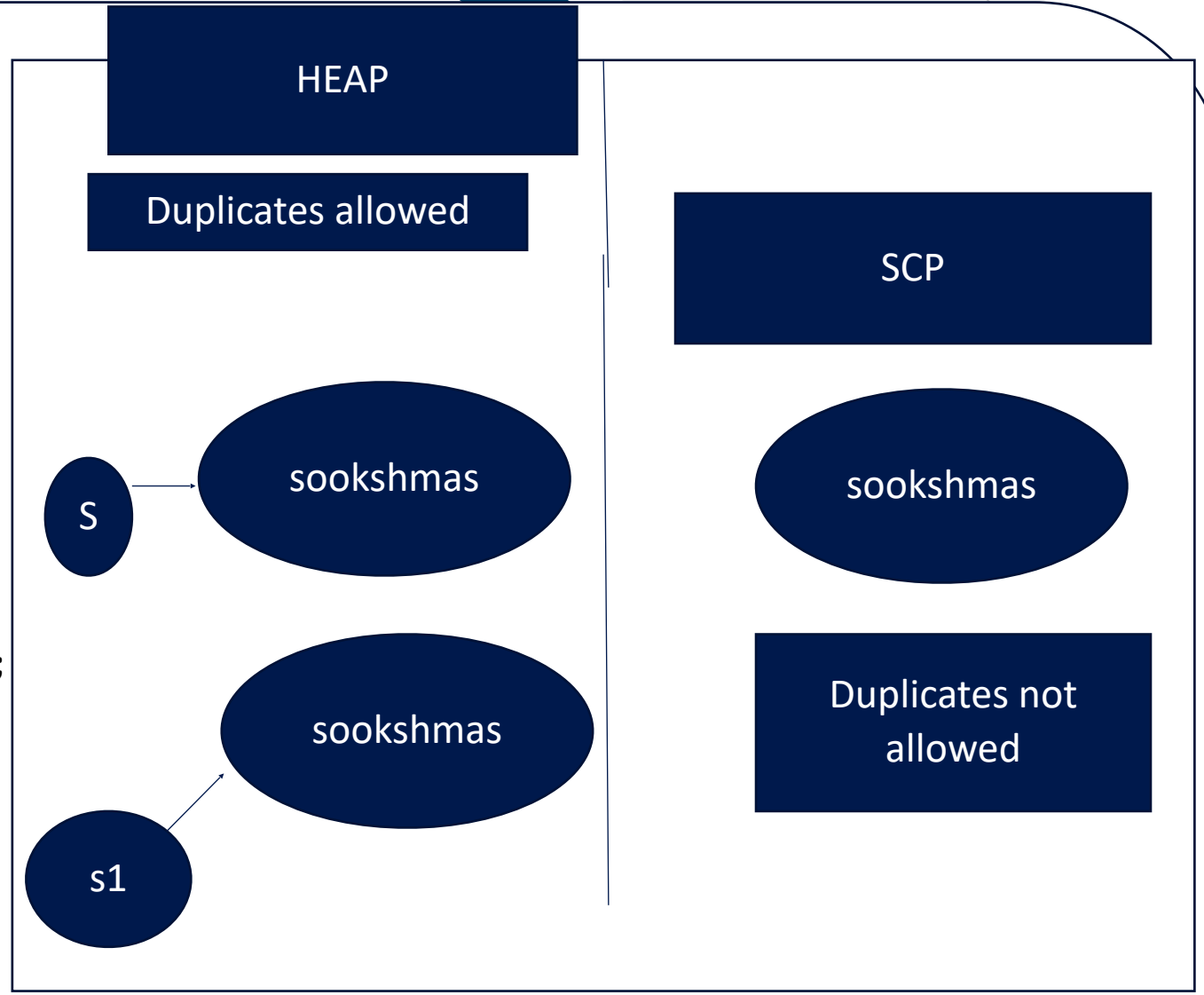
```
public class immutable {


 public static void main(String args[])
{
  String s="Sookshmas";

  String s1=“Sookshmas”;

  System.out.println(s +” “+s1);
 }
}
```



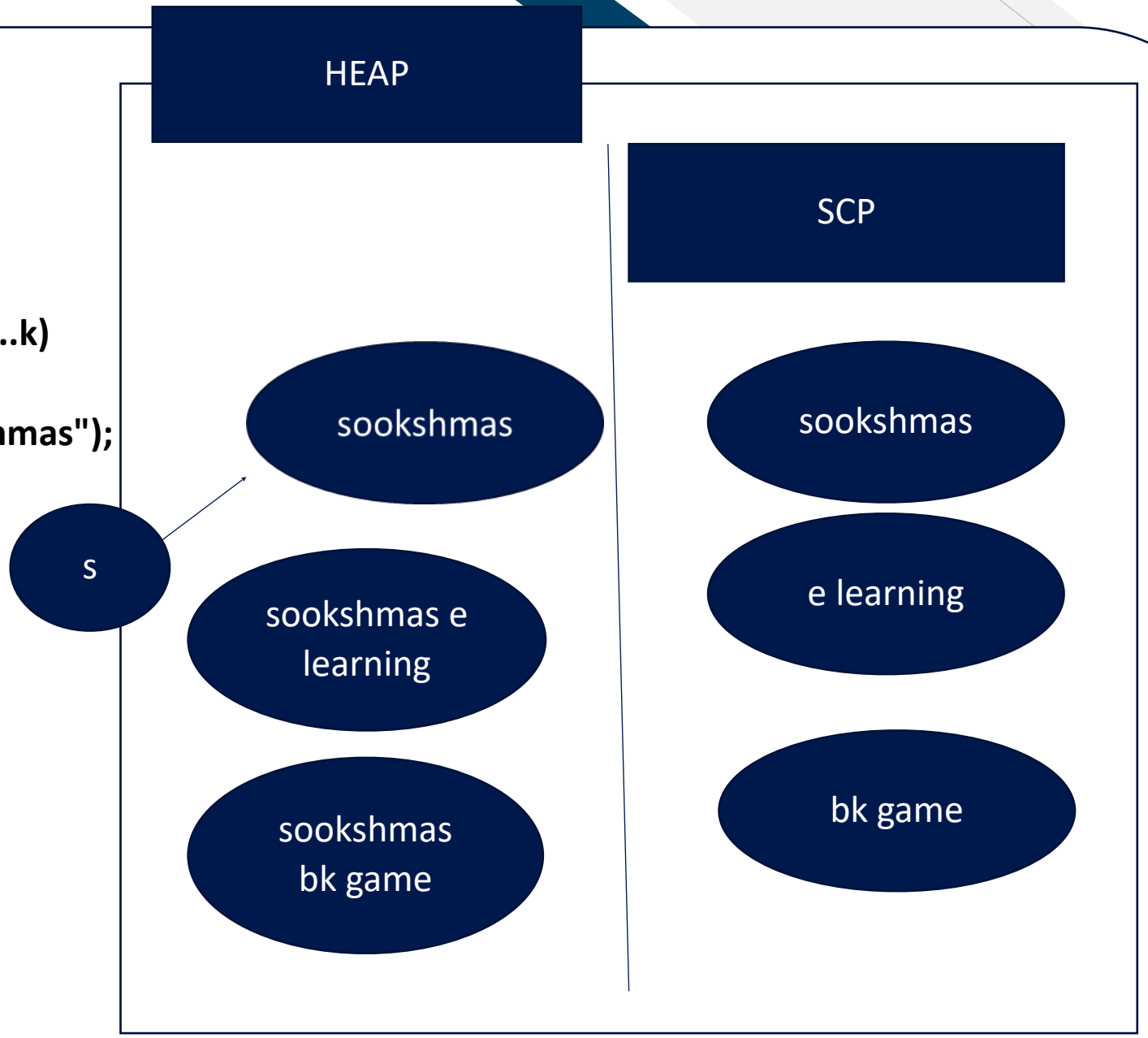
```
public class sookshmas2
{

   public static void main(String...k)
   {
      String s=new String("sookshmas");

     String  s1=new String("sookshmas");


      if(s==s1)
 {
    System.out.println("ref are equal");
 }

 else
 {
    System.out.println(" ref are unequal");
 }
```

```
if(s.equals(s1))
 {
   System.out.println("equal");
 }
 else
 {
    System.out.println("unequal");

 }
  }
}
```

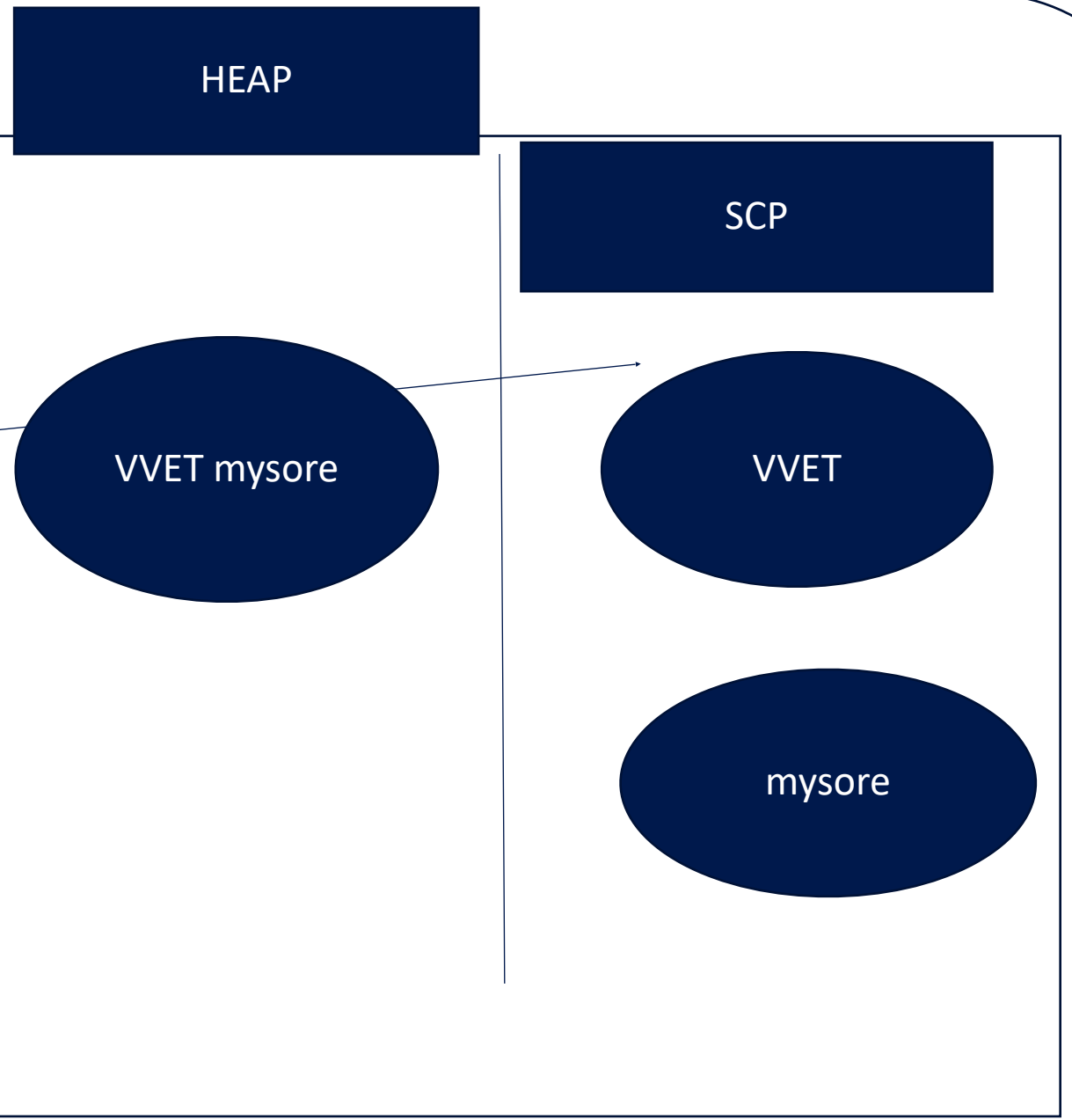
```
public class sookshmas2
{

   public static void main(String...k)
   {
      String s=new String("sookshmas");

     String  s1=new String("sookshmas");

      System.out.println(s +" "+ s1);
   }
}
```



```
public class Sookshmas3
{
   public static void main(String...k)
   {
      String s=new String("sookshmas");

      s.concat("e learning");

      s.concat("bk game");

      System.out.println(s);
   }

}
```

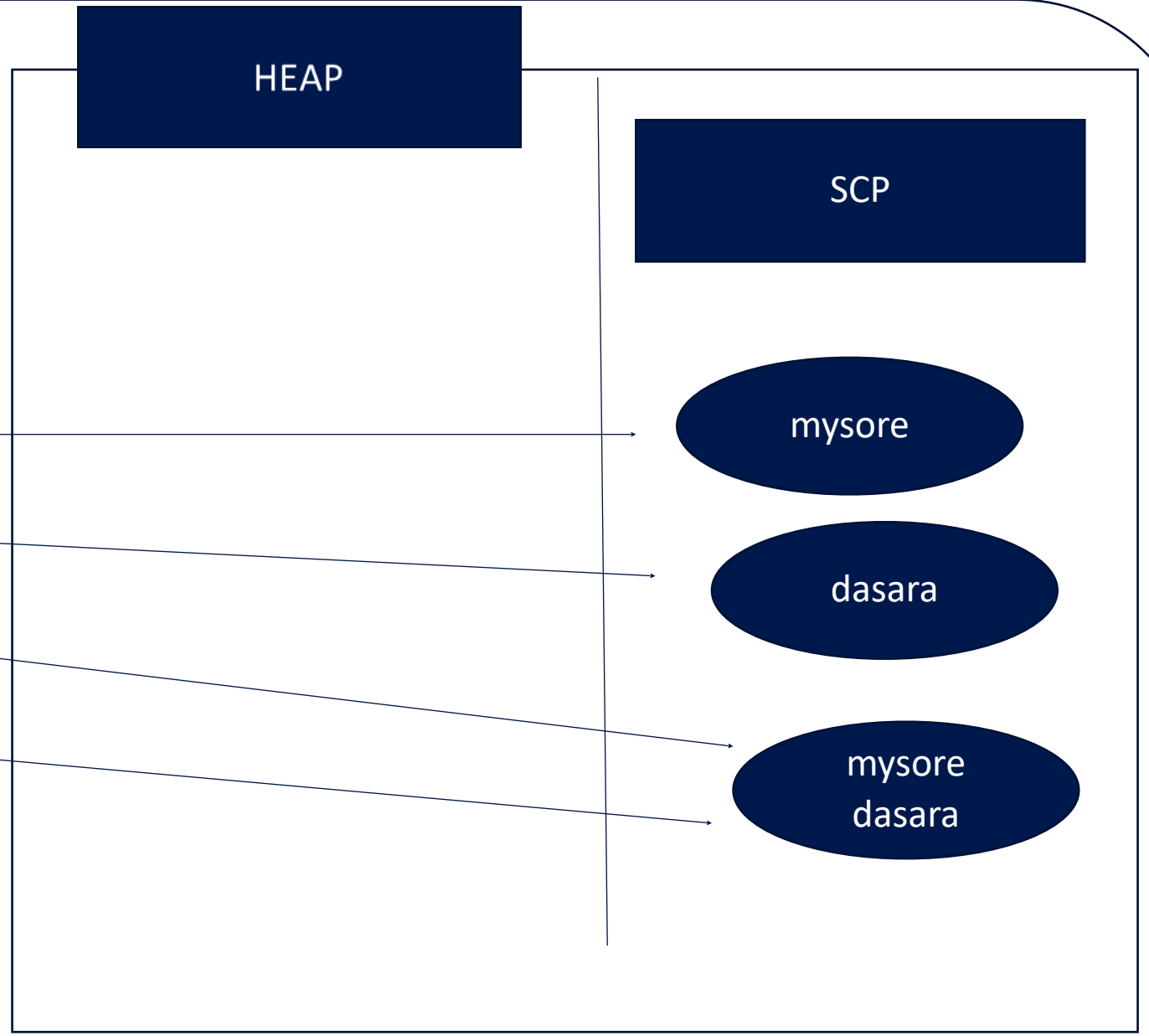


```
public class sookshmas4
{
  public static void main(String...k)
  {
     String college="VVET ";

     String address= college +"mysore";


  }
}
```



```
public class test
{
 public static void main(String args[])
{

        String college="VVET ";

    String address= college +"mysore";
     String address1= college +"mysore";

     if(address==address1)
     {
        System.out.println("EQ");
     }
     else
     {
        System.out.println("NOT EQ");
     }

  }
}
```

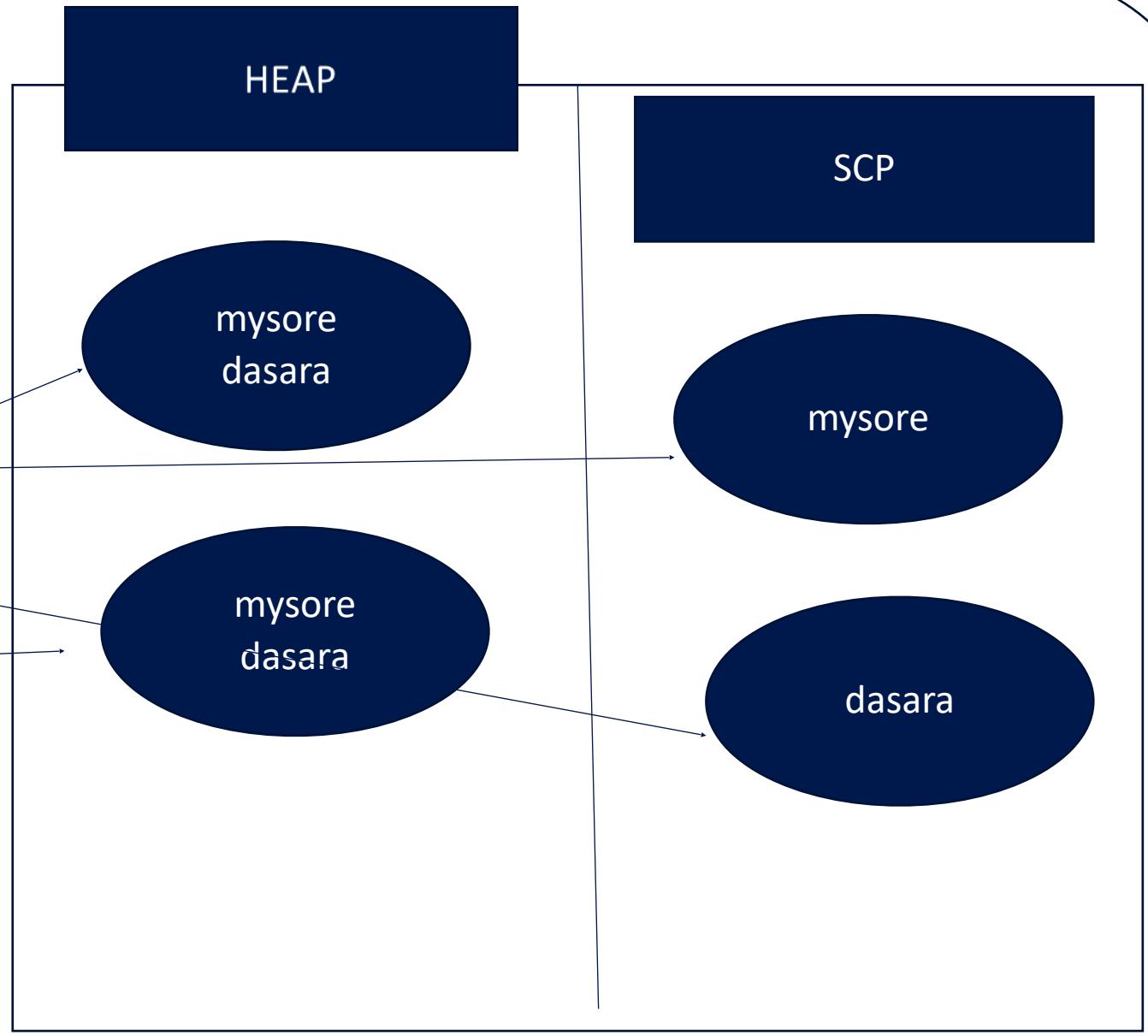
```
public class sookshmas5
{
   public static void main(String...k)
   {
      String s1="mysore ";
      String s2="dasara";

     String s3="mysore"+"dasara";

     String s4="mysore"+"dasara";
   }

}
```



```
public class sookshmas5
{
   public static void main(String...k)
   {
      String s1="mysore";
      String s2="dasara";

     String s3="mysore"+"dasara";

     String s4="mysore"+"dasara";

     if(s3==s4)
 {
    System.out.println("ref are equal");
 }

 else
 {
    System.out.println(" ref are unequal");
 }
```

```
if(s3.equals(s4))
 {
   System.out.println("equal");
 }
 else
 {
    System.out.println("unequal");

 }
  }

}
```

```
public class sookshmas6
{

    public static void main(String...k)
    {
        String s1="mysore ";
        String s2="dasara";

        String s3=s1+s2;
        String s4=s1+s2;


    }
}
```

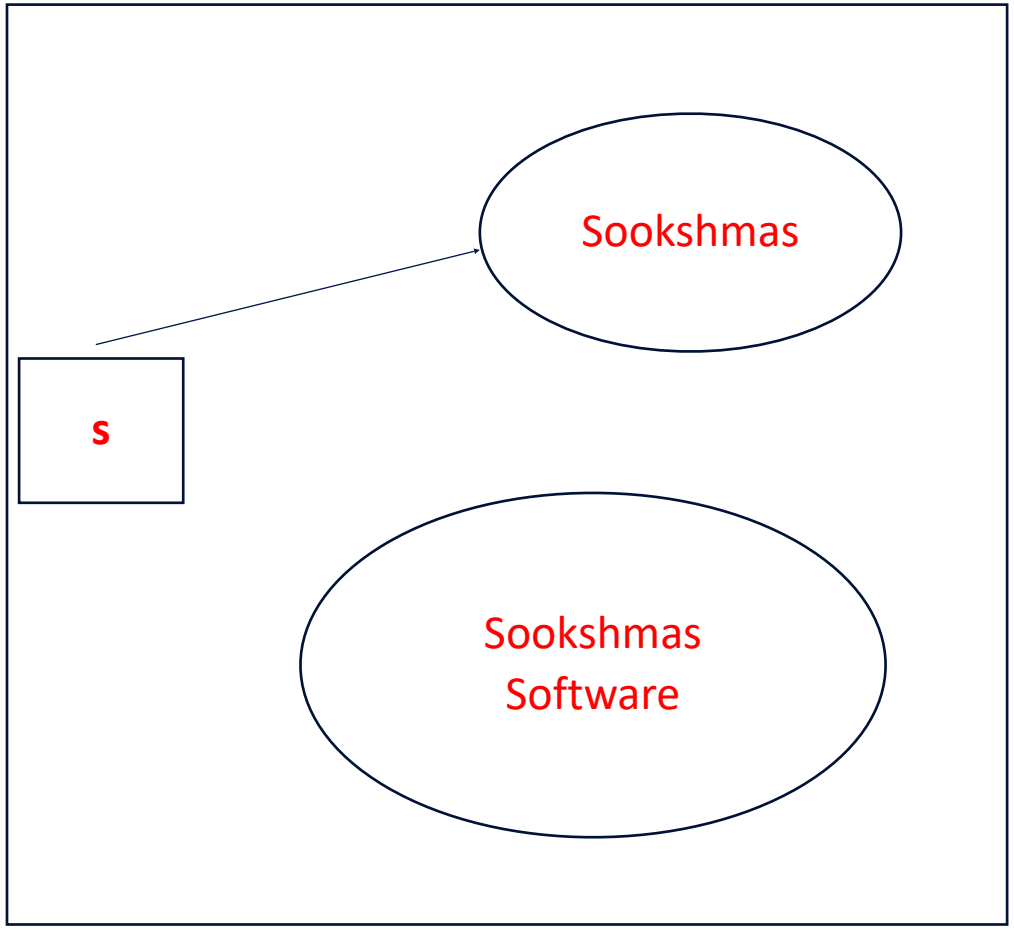


```
public class sookshmas6
{

  public static void main(String...k)
  {
     String s1="mysore";
     String s2="dasara";

     String s3=s1+s2;
     String s4=s1+s2;
     if(s3==s4)
{
   System.out.println("ref are equal");
}

else
{
   System.out.println(" ref are unequal");
}
```

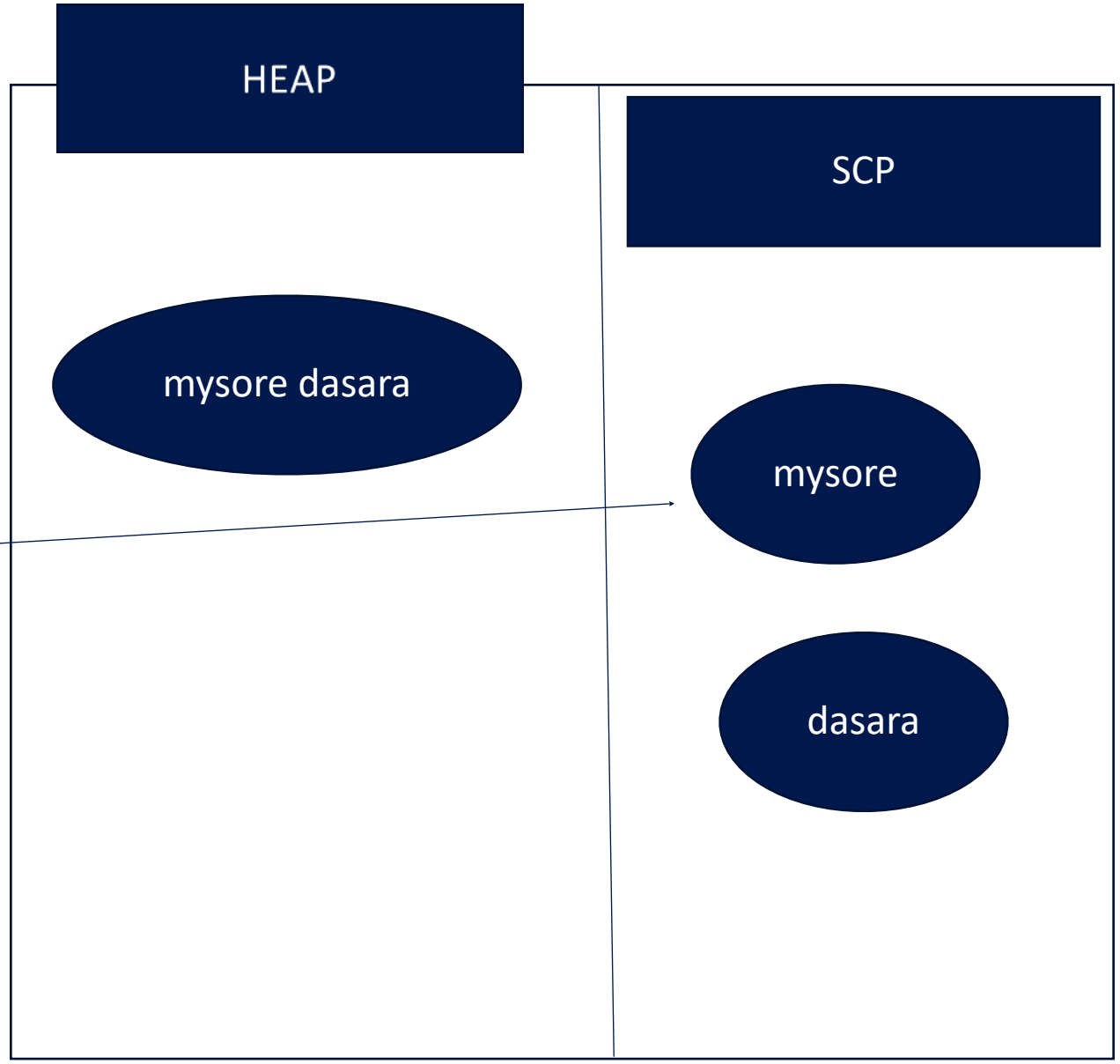
```
if(s3.equals(s4))
 {
   System.out.println("equal");
 }
 else
 {
    System.out.println("unequal");

 }


   }
}
```

Strings are immutable in java:

```
public class immutable {

 public static void main(String args[])
{
 String s="Sookshmas";

   s.concat(" Software");

   System.out.println(s);

}
}
```



```
public class sookshmas7
{

   public static void main(String...k)
   {
      String s1="mysore ";
      s1.concat("dasara");
   }
}
```

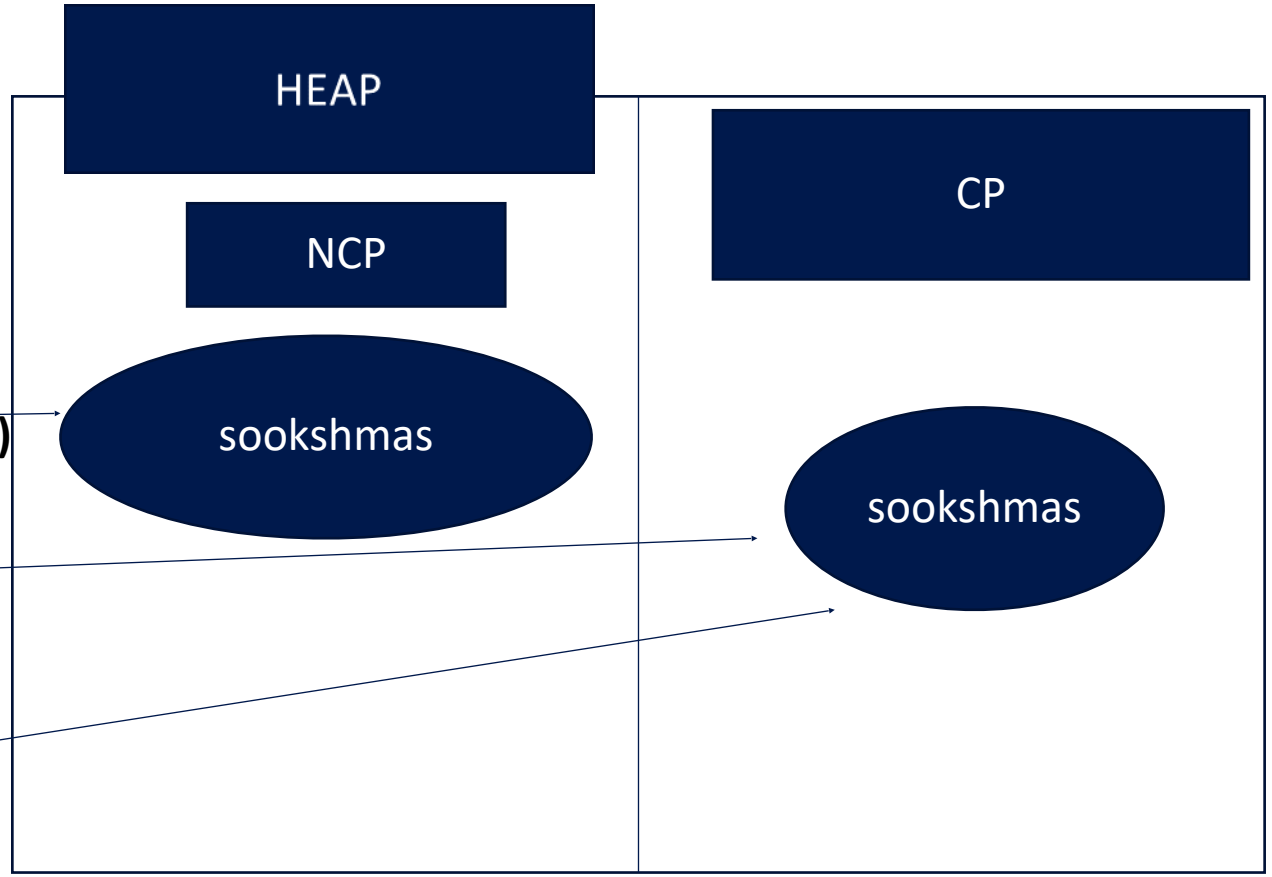


```
public class test
{
 public static void main(String args[])
{

      String s1="mysore ";
  String    s2= s1.concat("dasara");
      String s3=s1.concat("dasara");

      System.out.println(s2+" "+s3);
```

```
if(s2==s3)
      {
         System.out.println("equal");
      }

      else
      {
         System.out.println("not EQ");
      }

      if(s2.equals(s3))
      {
         System.out.println("eq");
      }
      else
      {
         System.out.println("not eq");
      }


}
}
```

Intern()

```
public class sookshmas7
{
    public static void main(String...k)
   {

      String s1=new String("sookshmas");

      String s2=s1.intern();

      System.out.println(s2==s1);

      String s3="sookshmas";

      System.out.println(s2==s3);
}

}
```



Intern(): it is used to create copy of the string which is in the non constant pool region to constant pool region.

String:

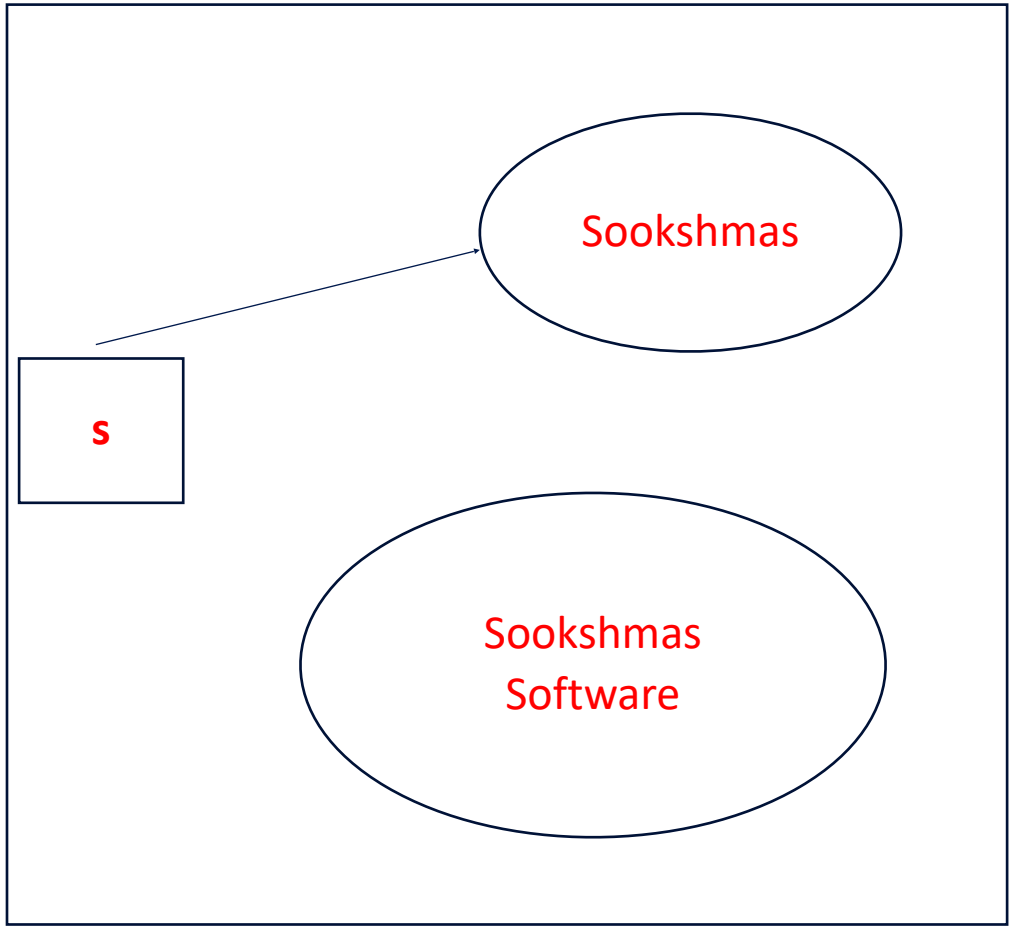
1. If the content will change frequently then it is not recommended.
2. Because every changes new objected will be created.

StringBuffer:

1.All the changes will be performed in existing object. So no need to create a new object.

Strings are immutable in java:

```
public class immutable {

 public static void main(String args[])
{
 String s="Sookshmas";

  s.concat(" Software");

  System.out.println(s);

  String s1=new String(" vvet");

  s1.concat("college");

   System.out.println(s1);
}
}
```



```
public class niit6

{

    public static void main(String args[])
{
   String s1="Sookshmas";
   String s2="sookshmas";
  if(s1==s2)
  {
      System.out.println("true");
  }
  else
  {
     System.out.println("false");
  }
   System.out.println(s1.equals(s2));
   System.out.println(s1.equalsIgnoreCase(s2));
 }

}
```

```
//compareTo() method

public class niit8
{
        public static void main(String args[])
 {

    String s1="ZBCD";
  String s2="ZBCD";
  String s3="Karthik";
  System.out.println(s1.compareTo(s2));
//0
  System.out.println(s1.compareTo(s3));
//(because s1>s3)

 System.out.println(s3.compareTo(s1));
        //(because s3 < s1 )  */
 }

}
```

```
// + (string concatenation) operator
//             By concat() method

public class niit9
{

  public static void main(String args[])
 {


  String s1=10+30+" banglore "+100+40+" ashok";
                System.out.println(s1);


 }


}
```

```
public class niit10
{
   public static void main(String args[])
 {

   String s1="sookshmas";
  System.out.println(s1.substring(6));
  System.out.println(s1.substring(0,6));

            String s4="sookshmas";
             System.out.println(s4.toUpperCase());
      System.out.println(s4.toLowerCase());
             System.out.println(s4);//(no change in original)
}
}
```

```
public class niit10
{
   public static void main(String args[])
 {

String s5=" sookshmas bang ";

System.out.println(s5.length());
System.out.println(s5.trim());
System.out.println(s5.trim().length());
System.out.println(s5.charAt(1));
System.out.println(s5.charAt(3));

}
}
```

```
//The string valueOf() method coverts given type such as
//int, long, float, double,
//boolean, char and char array into string.

public class niit13
{
   public static void main(String args[])
   {
      int a=10;
      char d='K';
      float d1=67.78f;
     boolean b=true;
     char[] a1={'k','a','r','t','h','i','k'};

     String s=String.valueOf(a);
     System.out.println(s);
     System.out.println(String.valueOf(d));
     System.out.println(String.valueOf(d1));
     System.out.println(String.valueOf(b));
     System.out.println(String.valueOf(a1));

   }
   }
```

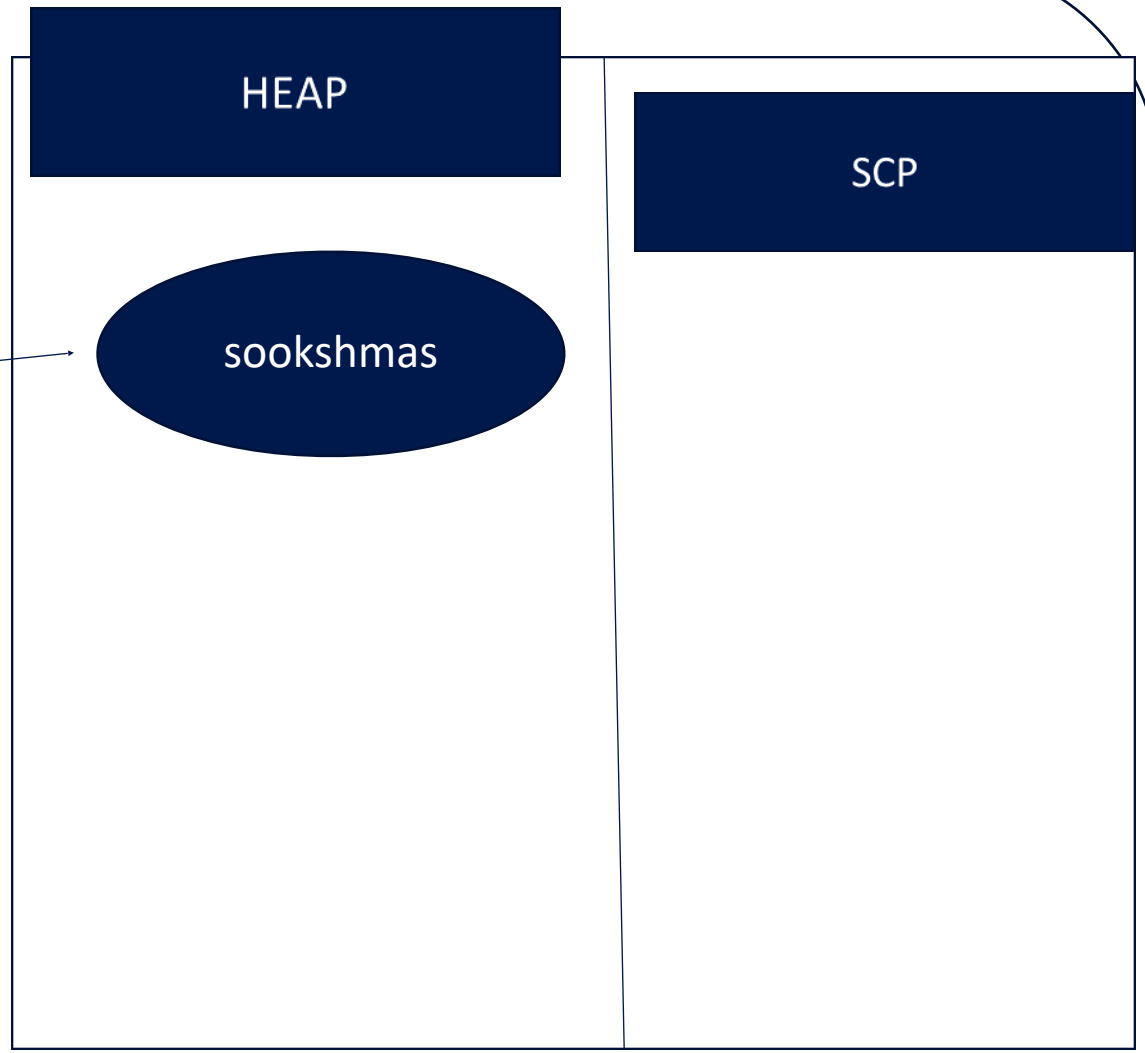
```
String s1="sookshmas vijayanagara sookshmas
jayanagar";

String replaceString=s1.replace("sookshmas","mysore");

System.out.println(s1.replace("sookshmas","BANG"));

      System.out.println(replaceString);
```

```
//public static String copyValueOf(char[] data,
        //int offset, int count)

    char[] Str1 = {'n','i','i','t','v','i','j','a','y','a','n'};
    String Str2 = "";
    Str2 = Str2.copyValueOf( Str1, 2, 6 );
    System.out.println("Returned String: " + Str2);
    System.out.println(".....\n");



String Str = new String("mysore is a good place!!");
    boolean retVal;

    retVal = Str.endsWith( "place!!" );
    System.out.println("Returned Value = " + retVal );

    retVal = Str.endsWith( "ait" );
    System.out.println("Returned Value = " + retVal );
```

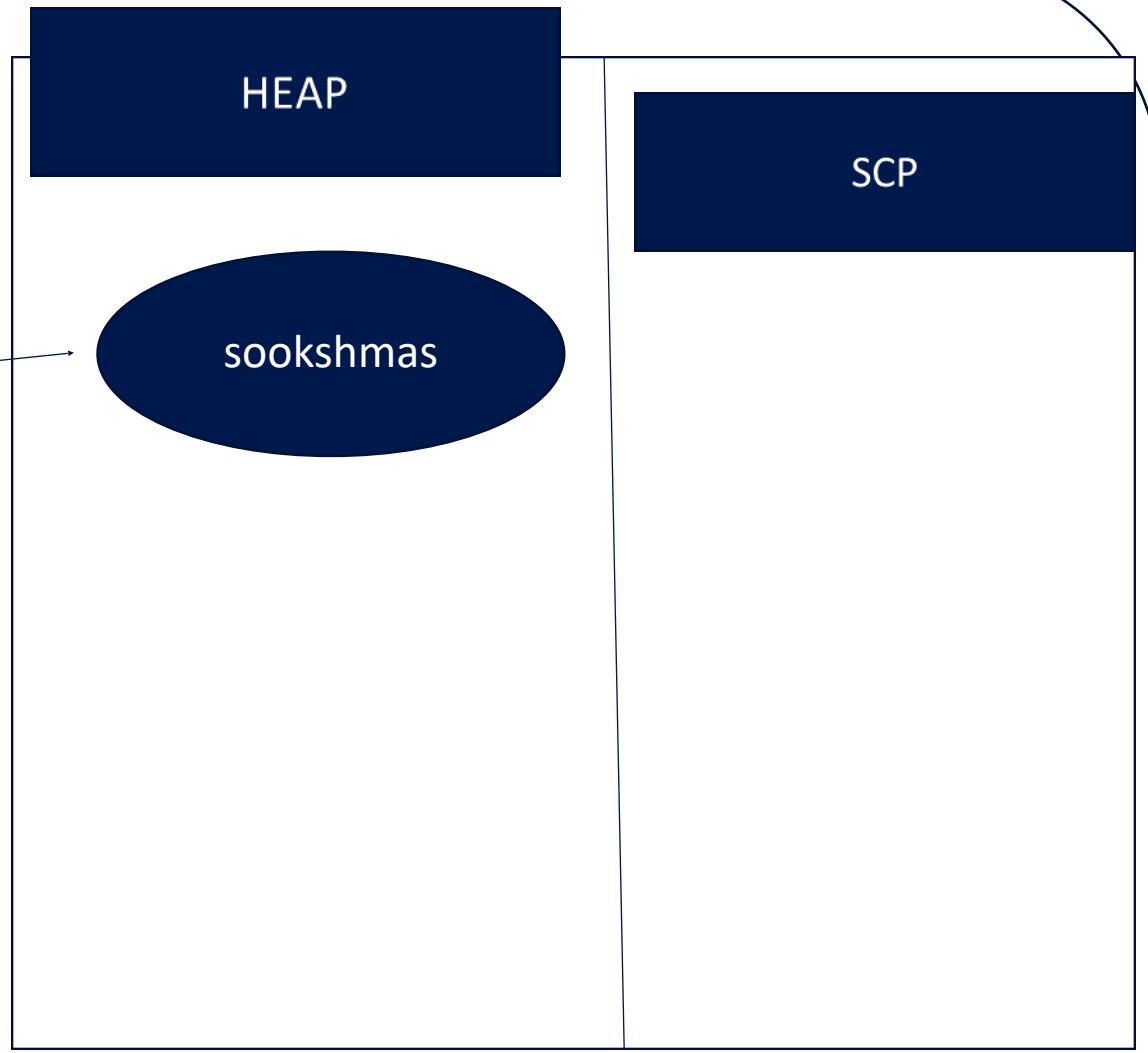
Methods:

public StringBuffer append(String data)

```
public class sookshmas7
{
     public static void main(String...k)
    {
    StringBuffer s1=new StringBuffer("sookshmas ");

StringBuffer s2=s1.append("Software ");

StringBuffer s3=s2.append("bangalore");
System.out.println(s1);
System.out.println(s2);
System.out.println(s3);



}

}
```



```
public class sookshmas7
{
     public static void main(String...k)
   {
StringBuilder s1=new StringBuilder("sookshmas ");

StringBuilder s2=s1.append("Software ");

StringBuilder s3=s2.append("bangalore");
System.out.println(s1);
System.out.println(s2);
System.out.println(s3);



}

}
```



StringBuffer: (mutable)

Costructors:
1.public StringBuffer()

Its create an empty StringBuffer object with 16 elements as initial capacity.

```
public class sookshmas7
{
      public static void main(String...k)
   {

     StringBuffer sb=new StringBuffer();
System.out.println( sb.capacity() );

      }

}
```

2.public StringBuffer(int capacity)

Its create an StringBuffer object with the specified capacity value.

```
public class sookshmas7
{
     public static void main(String...k)
   {
   StringBuffer sb =new StringBuffer(10);
System.out.println(sb.capacity());


}

}
```

3.public StringBuffer(String str)
It create StringBuffer object with the specified data .

Newcapacity = InitialCapacity + data_Length;

```
public class Demo {

   public static void main(String args[]) {

       StringBuffer sb=new StringBuffer("xyz");
      System.out.println(sb);
      System.out.println(sb.capacity());


   }

}
```

```
public class sookshmas7
{
     public static void main(String...k)
   {
   StringBuffer s=new StringBuffer();
      System.out.println( s.capacity()); //16

      s . append("sookshmas e learining pvt ltd");

      System.out.println( s.capacity()); //34


}

}
```

New capacity=(old capacity * 2)+2

```
public class sookshmas7
{
     public static void main(String...k)
   {
   StringBuffer s=new StringBuffer();
      System.out.println( s.capacity());  //16

      s.ensureCapacity(100);

      System.out.println(s.capacity()); //100


}

}
```

public StringBuffer reverse().
This method will reverse the content of StringBuffer object.

```
public class sookshmas7
{
     public static void main(String...k)
   {
  StringBuffer s=new StringBuffer("sookshmas Software" );

System.out.println(s);
System.out.println( s.reverse());




}

}
```

4.public StringBuffer delete(int start_index, int end_index)

```
public class sookshmas7
{
     public static void main(String...k)
   {
  StringBuffer sb=new StringBuffer("sookshmas Software");

System.out.println(sb);
System.out.println(sb.delete(2,4)); //soshmas Software




}

}
```

```
public StringBuffer insert(int index, String str)

public class sookshmas7
{
     public static void main(String...k)
   {
  StringBuffer sb=new StringBuffer("sookshmas Solutions");

System.out.println(sb);
System.out.println(sb.insert(6,"Software"));

}

}
```

```
public class sookshmas7
{
     public static void main(String...k)
   {
  StringBuffer sb = new StringBuffer();
       sb.ensureCapacity(100);
       System.out.println(sb.capacity());
       sb.append("Infosys");
       sb.trimToSize();
       System.out.println(sb.capacity());
}

}
```

To release extra allocated memory

StringBuffer:

1. Every method is Synchronized.
2. StringBuffer object is threadsafe
3. Low performance
4. Introduced 1.0 version

StringBuilder:

1. No method is Synchronized.
2. Not thread safe.
3. High performance
4. Introduced in 1.5 v

```
public class sookshmas7
{
     public static void main(String...k)
   {
 final StringBuffer sb=new StringBuffer("sookshmas");

 sb=new StringBuffer("bangalore"); // cte

       System.out.println(sb);


final StringBuffer sb1=new StringBuffer("sookshmas ");

 sb1.append("bangalore");

       System.out.println(sb1);

}

}
```