**java.lang package – Part-05**

* **Java.lang.String: Case\_01:**

String s = new String(“durga”);

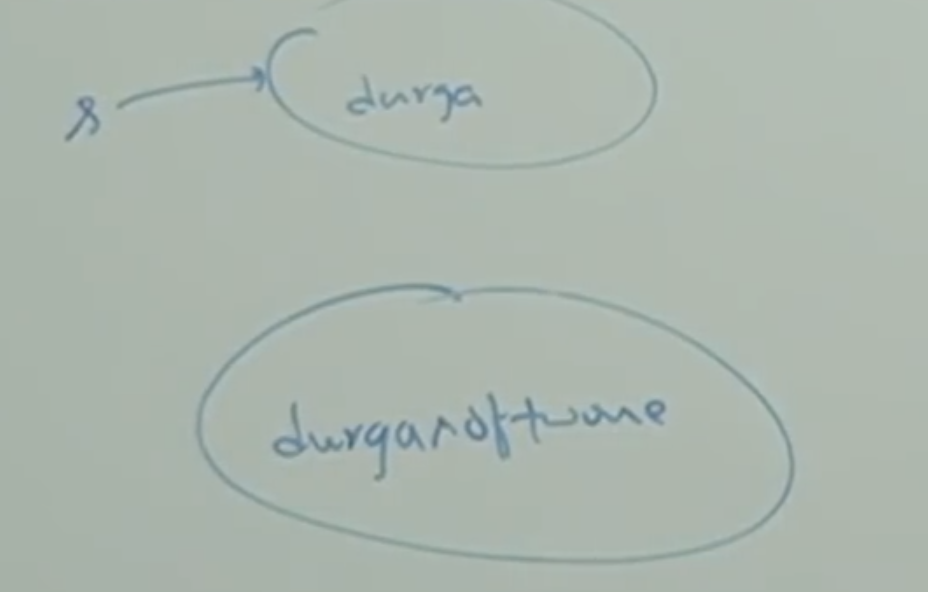
s.concat(“software”);

System.out.println(s);

Note:

Once we create a String object, we can’t perform any changes in the existing object.

If we are trying to perform any change with those changes a new object will be created. This non-changeable behavior is nothing but immutability of String.



Note:

The new object created with concat method doesn’t have any reference, so by default it is eligible for garbage collection. So, we will get the result as “durga”.

StringBuffer:

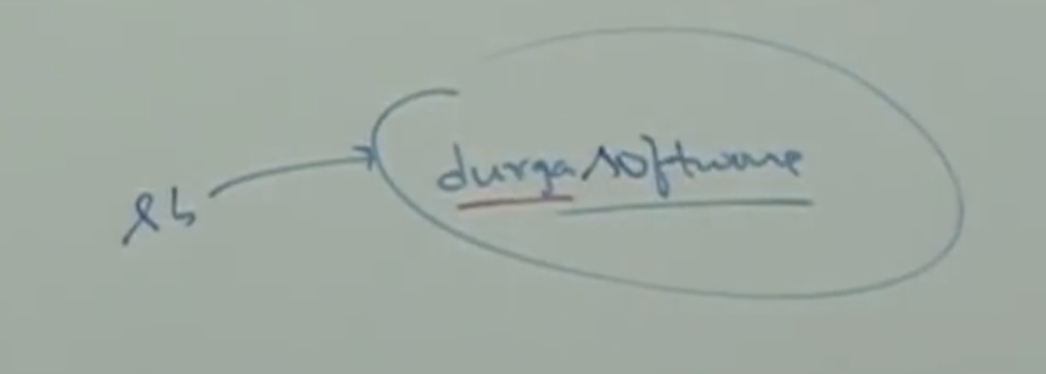
StringBuffer sb = new StringBuffer(“durga”);

sb.append(“software”);

System.out.println(sb);

Note:

Once we create StringBuffer object, we can perform any change in the existing object. This changeable behavior is nothing but mutability of StringBuffer object.



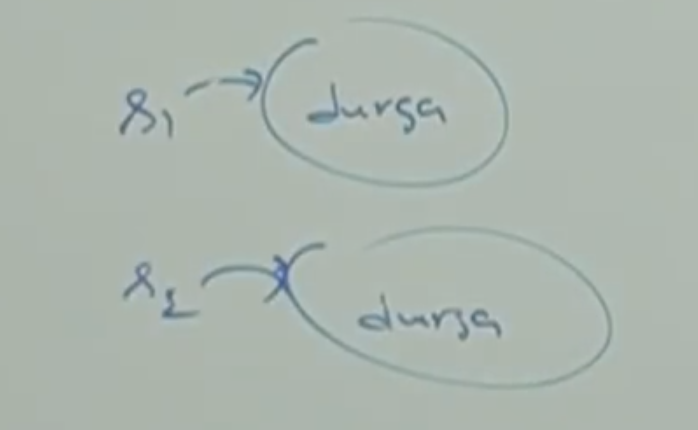
* **Case\_02:**

String s1 = new String(“durga”);

String s2 = new String(“durga”);

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

System.out.println(s1.equals(s2)); // true



Note:

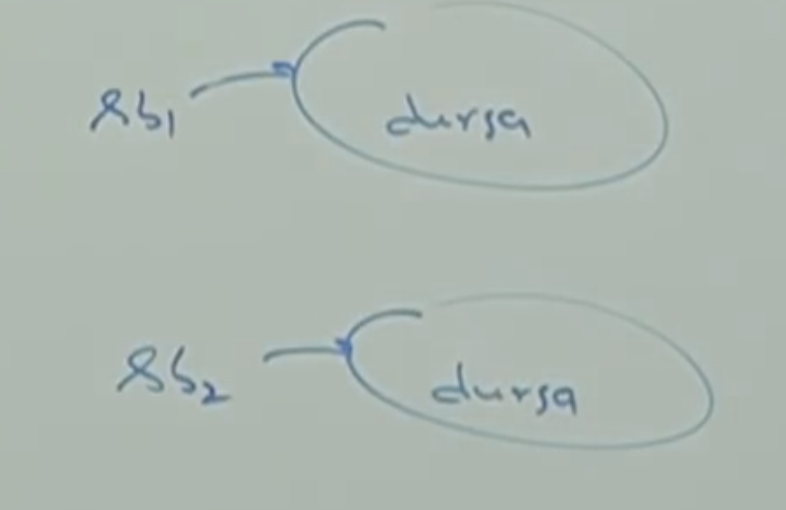
In String class .equals() method is overridden for content comparison. Hence, even though objects are difference, if content is same .equals() method returns true.

StringBuffer sb1 = new StringBuffer(“durga”);

StringBuffer sb2 = new StringBuffer(“durga”);

System.out.println(sb1==sb2); // false

System.out.println(sb1.equals(sb2));//false



Note:

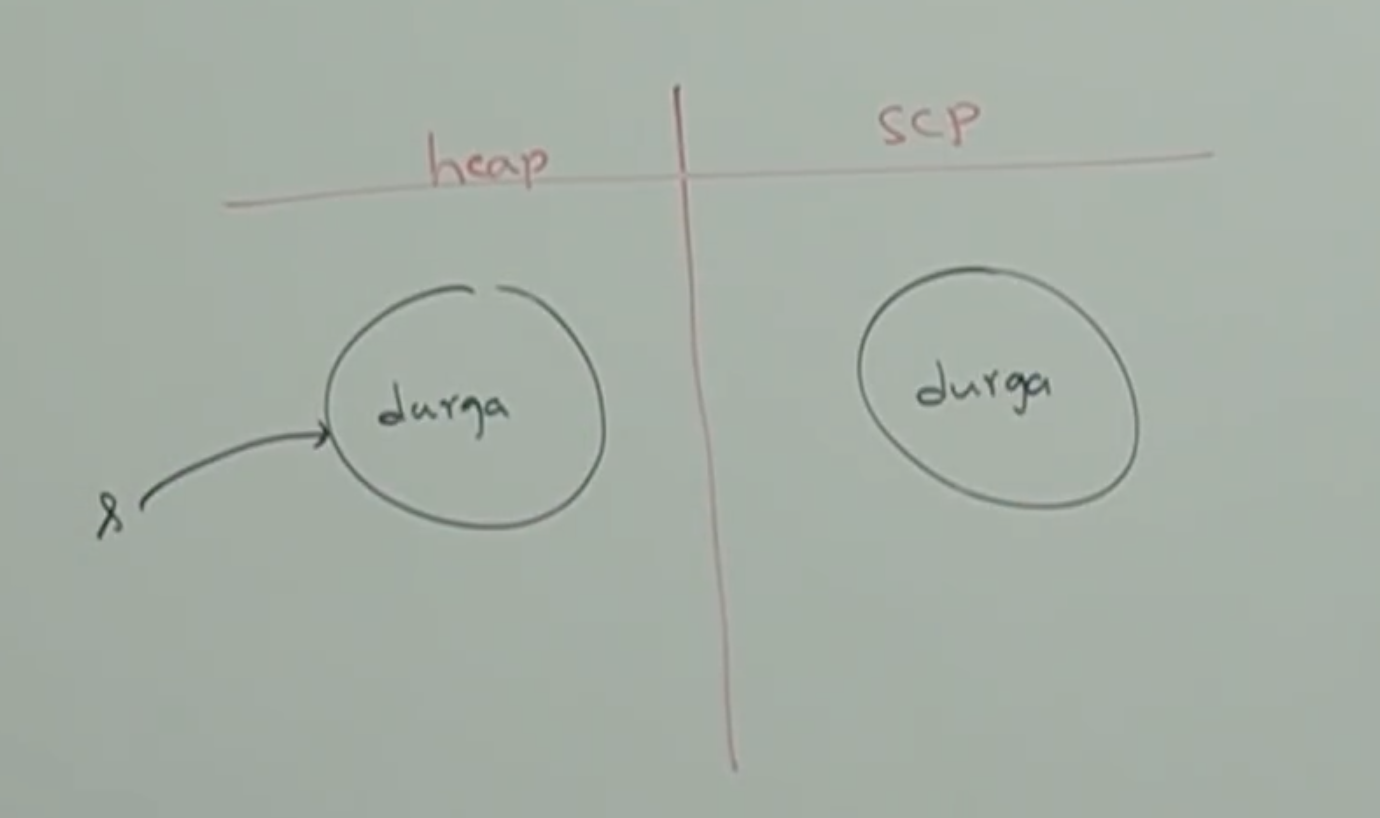
In StringBuffer class .equals() method is not overridden for content comparison. Hence, object class .equals() got executed, which is meant for reference comparison (address comparison). Due to this if objects are different .equals() method returns false even though content is same.

* **Case\_03:**

String s = new String(“durga”);

Note:

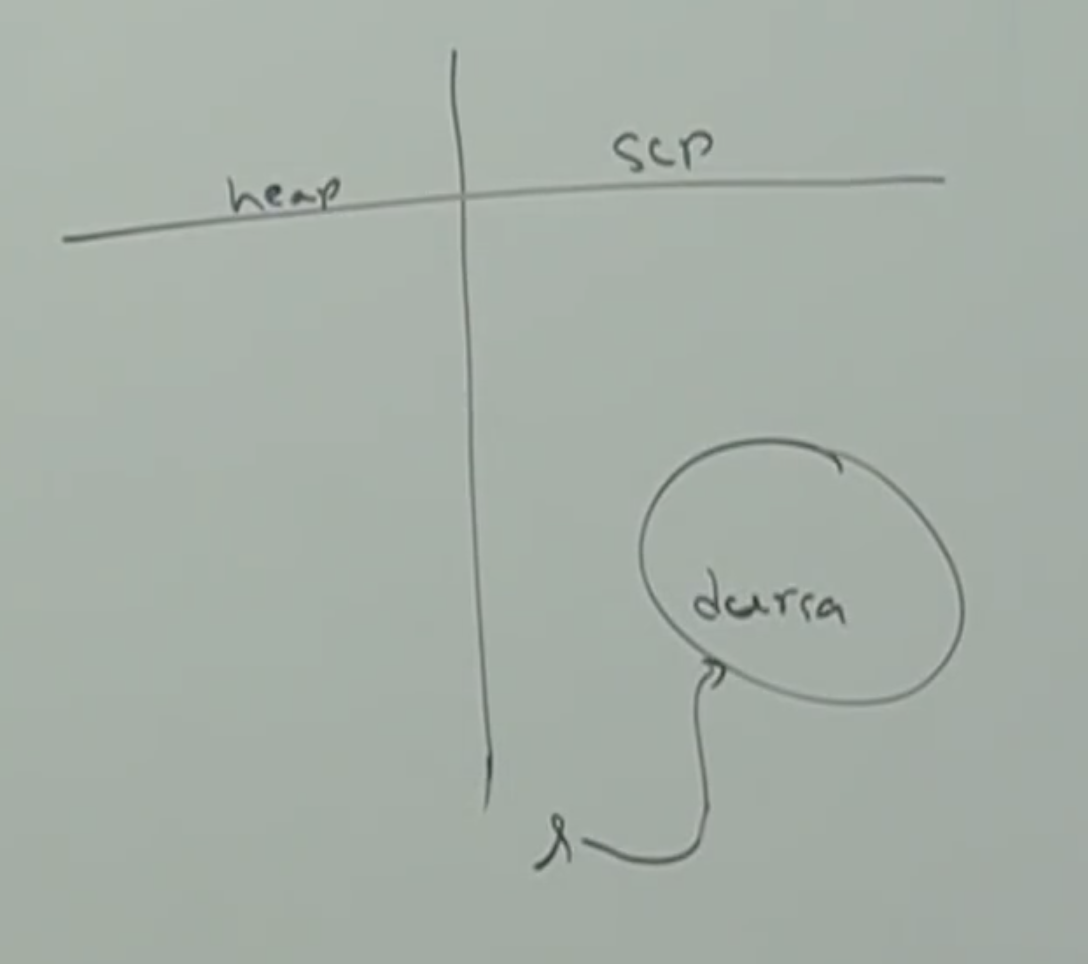
In this case two objects will be created one in the heap area and the other is in SCP (String Constant Pool) and “s” is always pointing to heap.



String s = “durga”;

Note:

In this case only one will be created in SCP and “s” is always pointing to that object.



Note:

1. Object creation in SCP is optional, first it will check is there any object already present in SCP with required content. If object already present then existing object will be reused. If object not already available then only a new object will be created. But this rule is applicable only for SCP, but not for the heap.
2. Garbage collector is not allowed to access SCP area. Hence, even though object doesn’t contain reference variable it is not eligible for GC, if it is present SCP area.
3. All SCP objects will be destroyed automatically at the time of JVM shutdown. In web application, this will happen at the time of server restart.

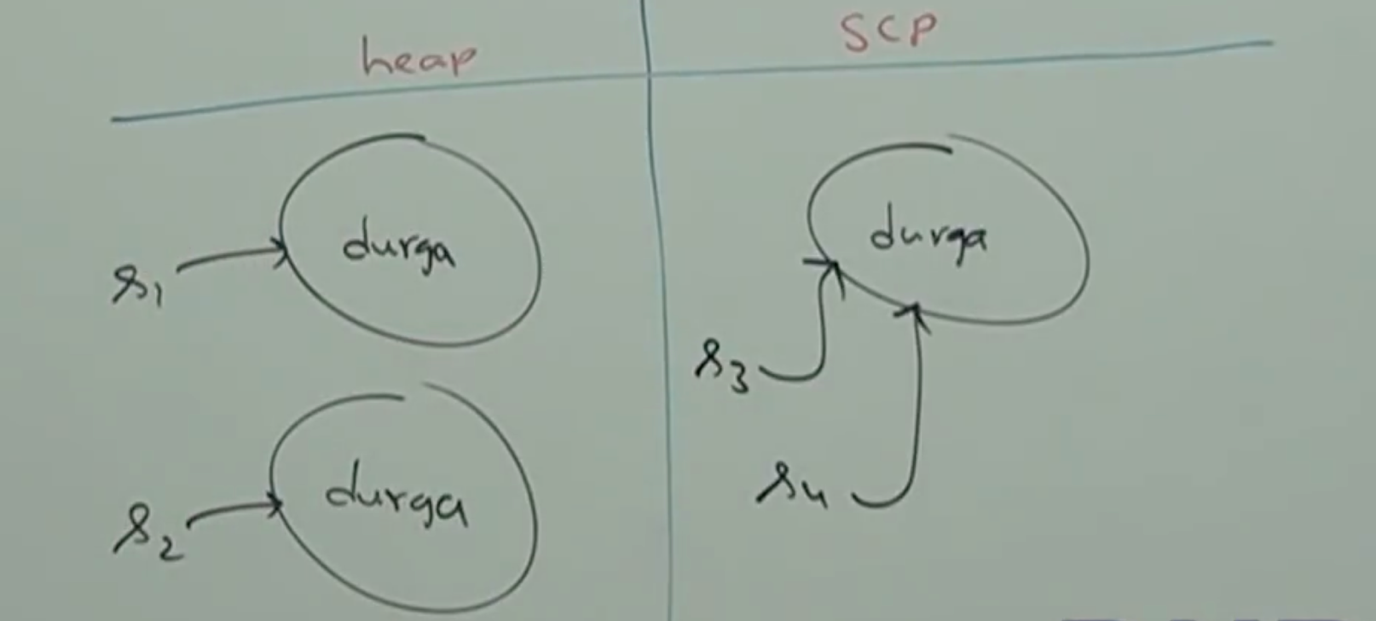
**Example\_02:**

String s1 = new String(“durga”);

String s2 = new String(“durga”);

String s3 = “durga”;

String s4 = “durga”;



**Note:**

Whenever we are using new operator compulsory a new object will be created in the heap area. Hence, there may be a chance of existing two objects with same content in the heap area, but not in SCP. That is, duplicate objects are possible in the heap area, but not in SCP.

**Example\_3:**

String s1 = new String(“durga”);

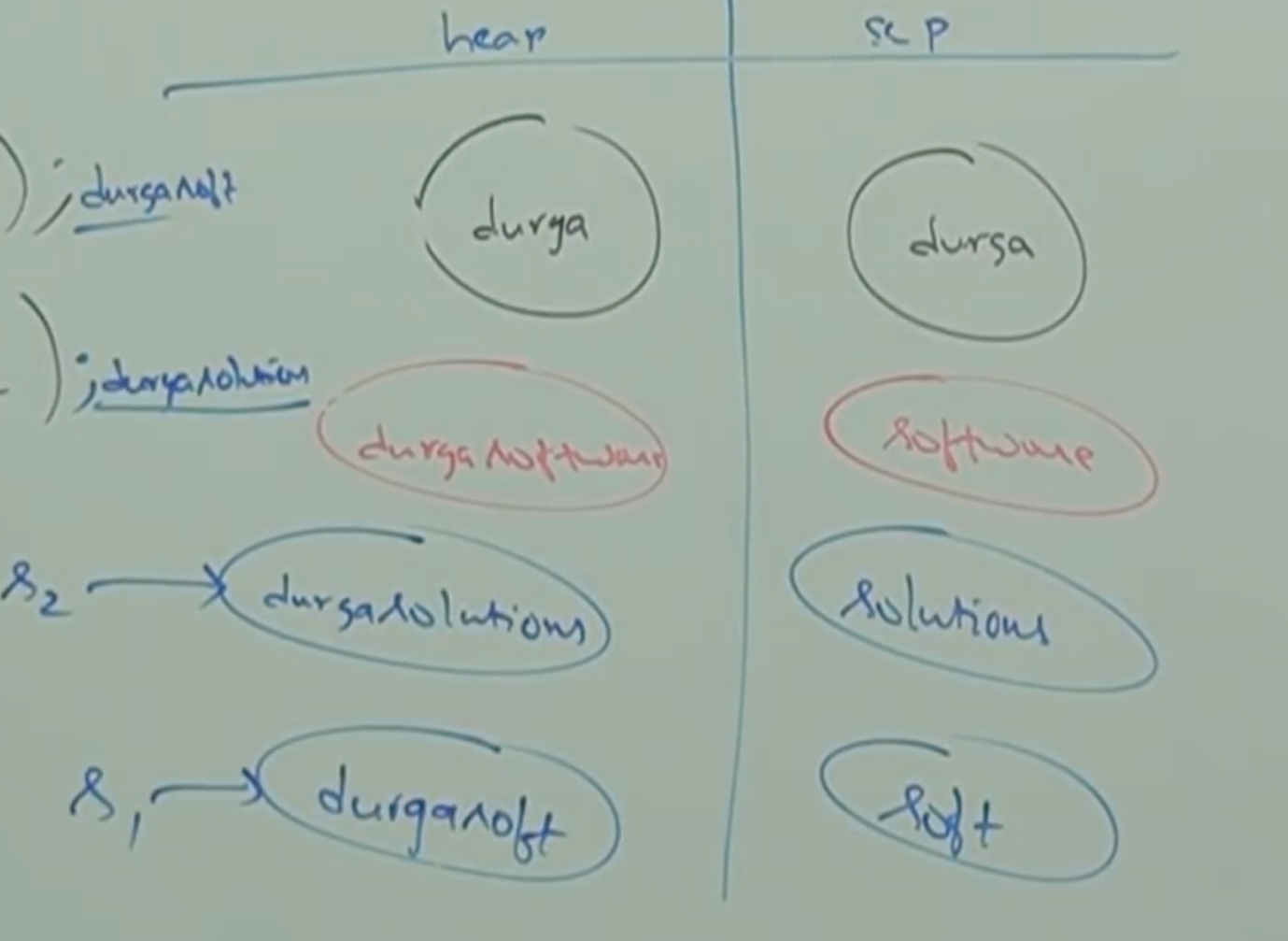
s1.concat(“software”);

String s2 = s1.concat(“solutions”);

s1 = s1.concat(“soft”);

System.out.println(s1); // durgasoft

System.out.println(s2); //durgasolutions



Note:

1. For every String constant one object will be placed in SCP area.
2. Because of some runtime operation if an object is required to create, that object will be placed only in the heap area but not in SCP area.

**Example\_04:**

String s1 = new String(“spring”);

s1.concat(“summer”);

String s2= s1.concat(“winter”);

s1=s1.concat(“fall”);

System.out.println(s1); // springfall

System.out.println(s2);// springwinter

