**Java.lang Package**

The following are some of important classes present in java.lang package.

1. Object class

2. String class

3. StringBuffer class

4. StringBuilder class (1.5 v)

5. Wrapper Classes

6. Autoboxing and Autounboxing(1.5 v)

**Note :**

· For writing any java program the most commonly required classes and interfaces are encapsulated in the separate package which is nothing but java.lang package.

· It is not required to import java.lang package in our program because it is available by default to every java program.

**The following is the list of all methods present in java.lang Object class :**

1. public String toString();

2. public native int hashCode();

3. public boolean equals(Object o);

4. protected native Object clone()throws CloneNotSupportedException;

5. public final Class getClass();

6. protected void finalize()throws Throwable;

7. public final void wait() throws InterruptedException;

8. public final native void wait()throws InterruptedException;

9. public final void wait(long ms,int ns)throws InterruptedException;

10. public final native void notify();

11. public final native void notifyAll();

**toString() Method:**

1. We can use this method to get string representation of an object.

2. Whenever we are try to print any object reference internally toString() method will be executed.

3.If our class doesn't contain toString() method then Object class toString() method will be executed.

class Student

{

String name;

int rollno;

Student(String name, int rollno)

{

this.name=name;

this.rollno=rollno;

}

public static void main(String args[]){

Student s1=new Student("saicharan",101);

Student s2=new Student("ashok",102);

System.out.println(s1);

System.out.println(s1.toString());

System.out.println(s2);

}

}

Output:

Student@3e25a5

Student@3e25a5

Student@19821f

**hashCode() Method:**

1.For every object jvm will generate a unique number which is nothing but hashCode.

2. Jvm will using hashCode while saving objects into hashing related data structures like HashSet, HashMap, and Hashtable etc.

| **class Student**  **{**  **public int hashCode()**  **{**  **return 100;**  **}**  **}**  **It is *improper way* of overriding**  **hashCode() method because for every**  **object we are generating same**  **hashcode.** | **class Student**  **{**  **int rollno;**  **public int hashCode()**  **{**  **return rollno;**  **}**  **}**  **It is *proper way* of overriding hashcode()**  **method because for every object we are**  **generating a different hashcode.** |
| --- | --- |

**Equals() Method:**

1.We can use this method to check equivalence of two objects.

2. If our class doesn't contain .equals() method then object class .equals() method will be executed which is always meant for reference comparison[address comparison]. i.e., if two references pointing to the same object then only .equals() method returns true .

**class Student**

**{**

**String name;**

**int rollno;**

**Student(String name,int rollno)**

**{**

**this.name=name;**

**this.rollno=rollno;**

**}**

**public static void main(String[] args){**

**Student s1=new Student("vijayabhaskar",101);**

**Student s2=new Student("bhaskar",102);**

**Student s3=new Student("vijayabhaskar",101);**

**Student s4=s1;**

**System.out.println(s1.equals(s2));**

**System.out.println(s1.equals(s3));**

**System.out.println(s1.equals(s4));**

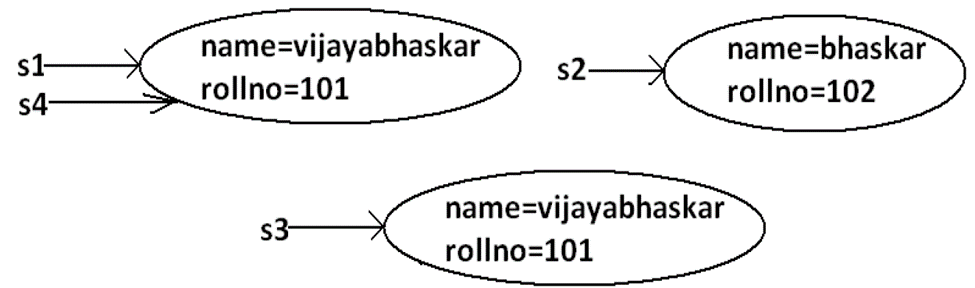
**}}**

**Output:**

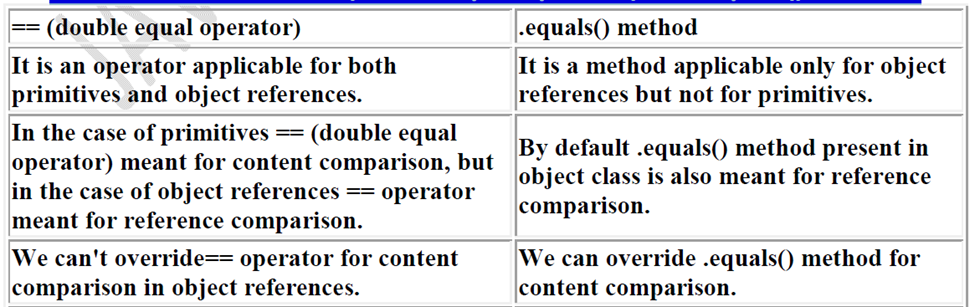
**False**

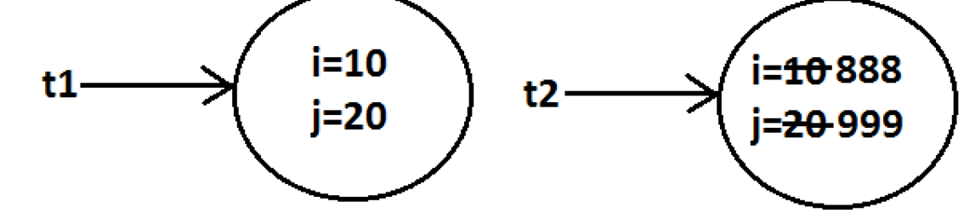
**False**

**True**



**Difference between == operator and .equals() method:**



****

**getClass() Method:**

This method returns runtime class definition of an object.

**Example :**

**class Test implements Cloneable {**

**public static void main(String[] args)throws**

**CloneNotSupportedException {**

**Object o=new String("ashok");**

**System.out.println("Runtime object type of o is**

**:"+o.getClass().getName());**

**}**

**}**

**Output:**

**Runtime object type of o is: java.lang. String**

**Ex : To print Connecton interface implemented vendor specific class name .**

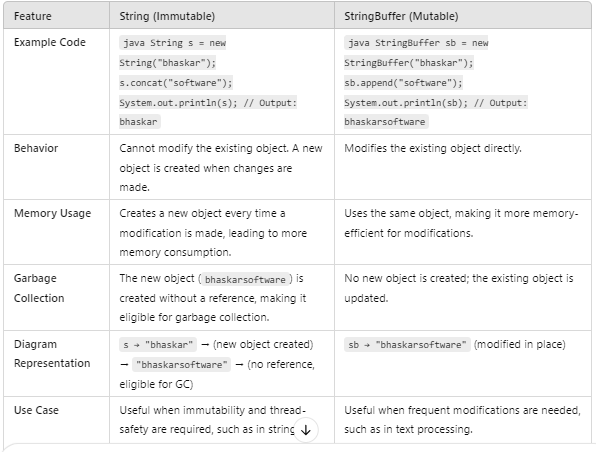
**System.out.println(con.getClass( ).getName( ) );**

**Finalize() Method:**

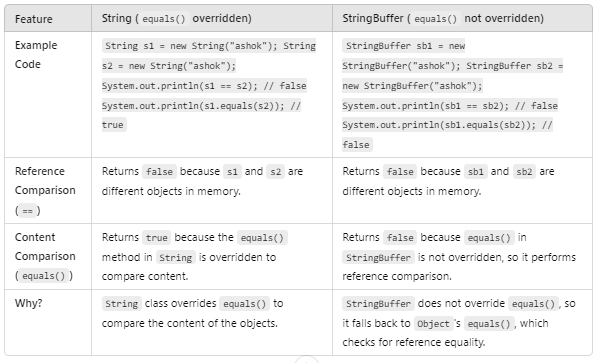
Just before destroying an object GC calls finalize( ) method to perform CleanUp activities .

**Java.lang.String class:**

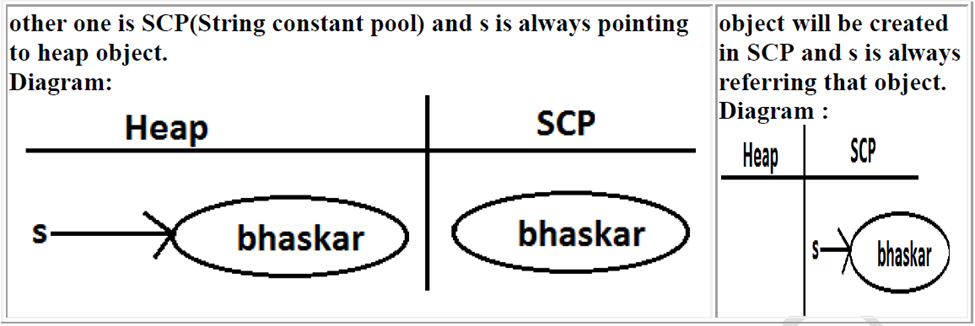
**Case 1:**

****

**Case 2:**

****

**Case 3:**

****

**Note :**

1. 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 existing object instead of creating new object. If it is not already there then only a new object will be created. Hence there is no chance of existing 2 objects with same content on SCP that is duplicate objects are not allowed in SCP.

2. Garbage collector can't access SCP area hence even though object doesn't have any reference still that object is not eligible for GC if it is present in SCP.

3. All SCP objects will be destroyed at the time of JVM shutdown automatically.

**Example 1:**

**String s1=new String("bhaskar");**

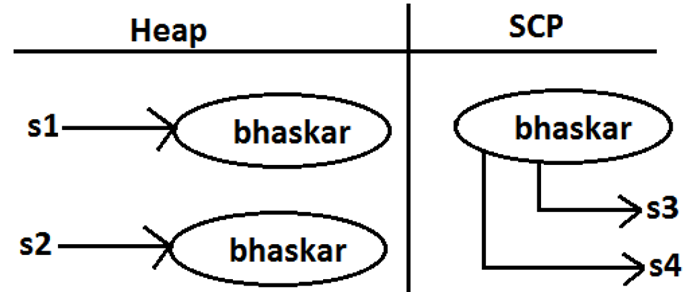
**String s2=new String("bhaskar");**

**String s3="bhaskar";**

**String s4="bhaskar";**

**Note :**

When ever we are using new operator compulsory a new object will be created on the Heap . There may be a chance of existing two objects with same content on the heap but there is no chance of existing two objects with same content on SCP . i.e., duplicate objects possible in the heap but not in SCP .



**String Constant Pool:**

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.

**String Buffer:**

1.If the content will change frequently then never recommended to go for String object because for every change a new object will be created internally.

2. To handle this type of requirement we should go for StringBuffer concept.

3. The main advantage of StringBuffer over String is, all required changes will be performed in the existing object only instead of creating new object.(won't create new object)

Object

**Note :**

Every method present in StringBuffer is synchronized hence at a time only one thread is allowed to operate on StringBuffer object , it increases waiting time of the threads and creates performance problems , to overcome this problem we should go for StringBuilder.

**String Builder:**

1. Every method present in StringBuffer is declared as synchronized hence at a time only one thread is allowed to operate on the StringBuffer object due to this, waiting time of the threads will be increased and effects performance of the system.

2. To overcome this problem sun people introduced StringBuilder concept in 1.5v.

**String vs StringBuffer vs StringBuilder**

1. If the content is fixed and won't change frequently then we should go for String.

2. If the content will change frequently but Thread safety is required then we should go for StringBuffer.

3. If the content will change frequently and Thread safety is not required then we should go for StringBuilder.