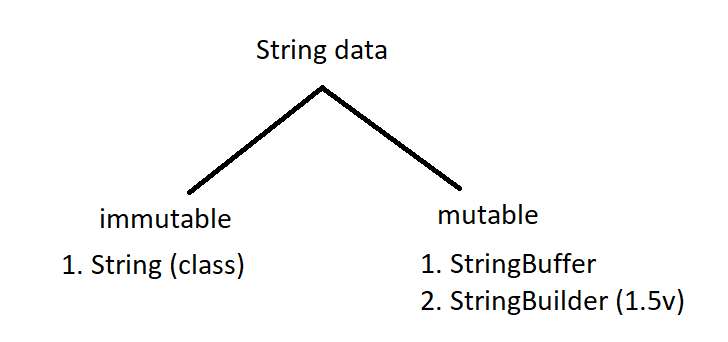
String

String is basically a inbuilt class present in java. lang. String package for which object can be created (user defined collection of characters enclosed in double quotes).

Eg: String\_Object\_Eg1



Since String is a class it may consists of instance variables and methods.

String:

class String{

// instance variables

// methods

}

To access the string methods we should create an object and call them.

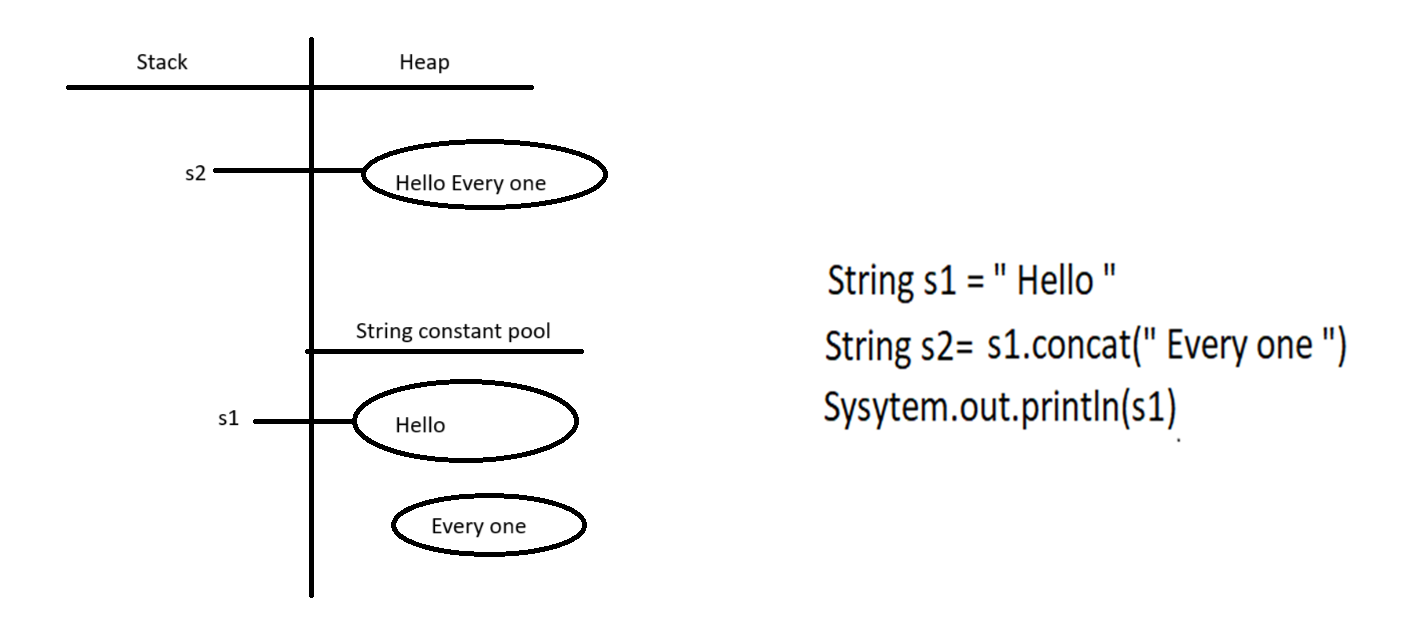
The object is the user defined string, with that reference variable of object we can call String methods

Eg: String\_Method\_Concat

If a String is declared is declared in the double quotes without new keyword, memory for it is allocated in the string constant pool. Here duplicates are allowed. If we try to create the another object in double quotes with same data then the address of the previous object is shared to this reference.

But when we try to create a object with new keyword then, object will be created in the string constant pool as well as heap also. But reference is pointed to the object in the heap. Here duplicates are not allowed, so if we try to create a another String object with new keyword and with same data ( as of the string in the heap) . reference is not shared like s.c.p .Here a new object will be created and object address is assigned to the reference.

If we try to make any operation on the String. And we are trying to do that operation with the double quoted data without new keyword as below, then one object will be created in the s.c.p without reference. And since that operation is evaluated in the runtime, object is created in the heap area with that change. If we does not collect the reference it will be cleaned by the garbage collector.

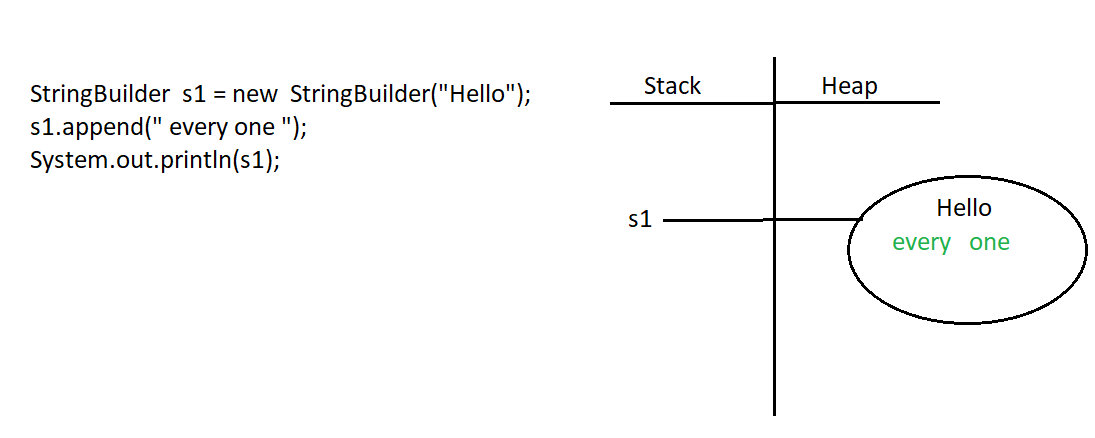


StringBuilder

String builder is mutable, when it is declared an object is created. if we want to add something to that object we can use append() method of StringBuilder class . unlike String class jvm will not create another object to add new data. instead it adds to old object.

In string builder object is not created in the s.c.p

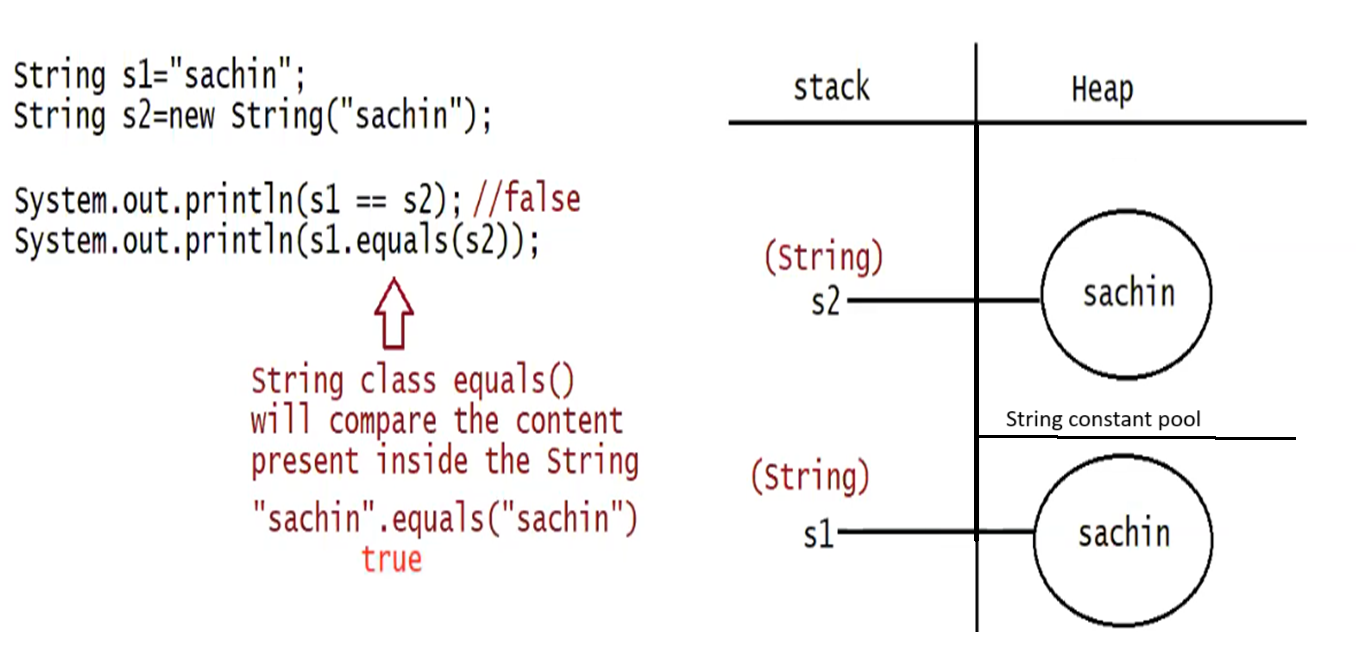
Eg: String\_Builder\_Eg3



String class equals method

Eg: String\_Equals\_Method\_Eg4

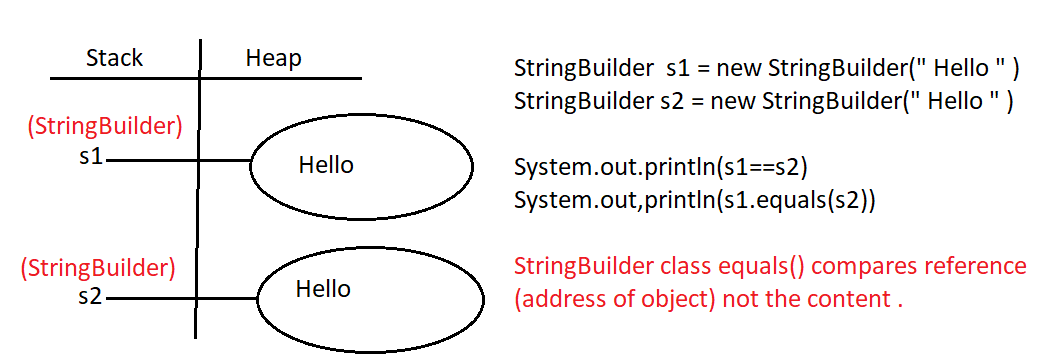
// go through the program



String \_Builder\_Equals \_Method

equals() method in String builder class compares the reference objects not the content in the objects.

Eg: String\_Builder\_Equals\_Method\_Eg5

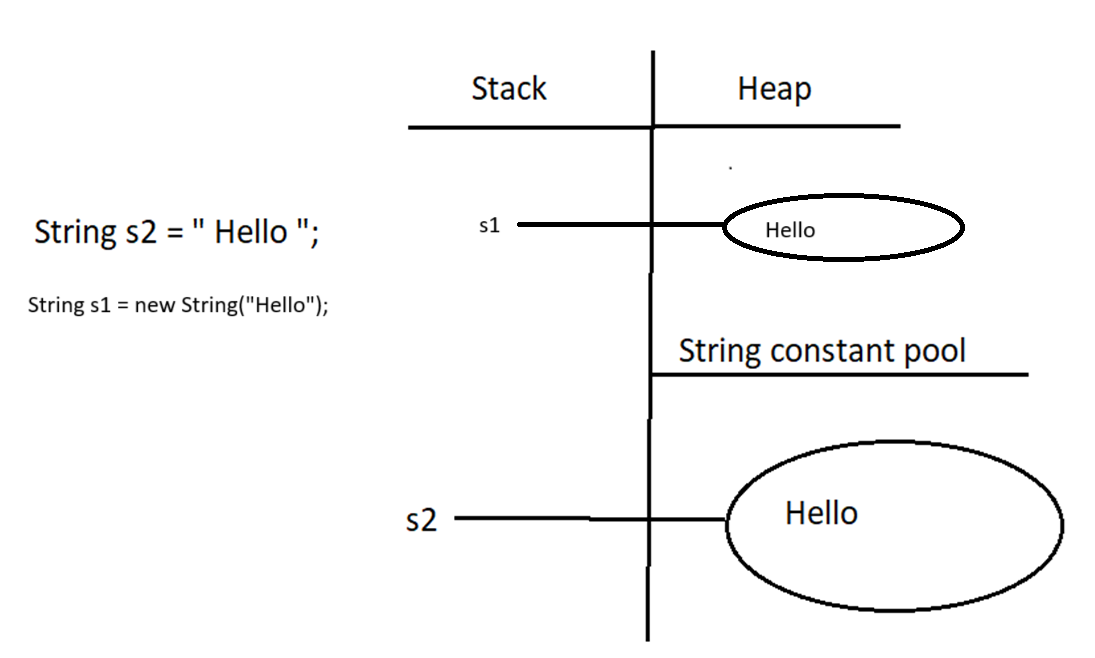


String s1 = new String (“Hello”);

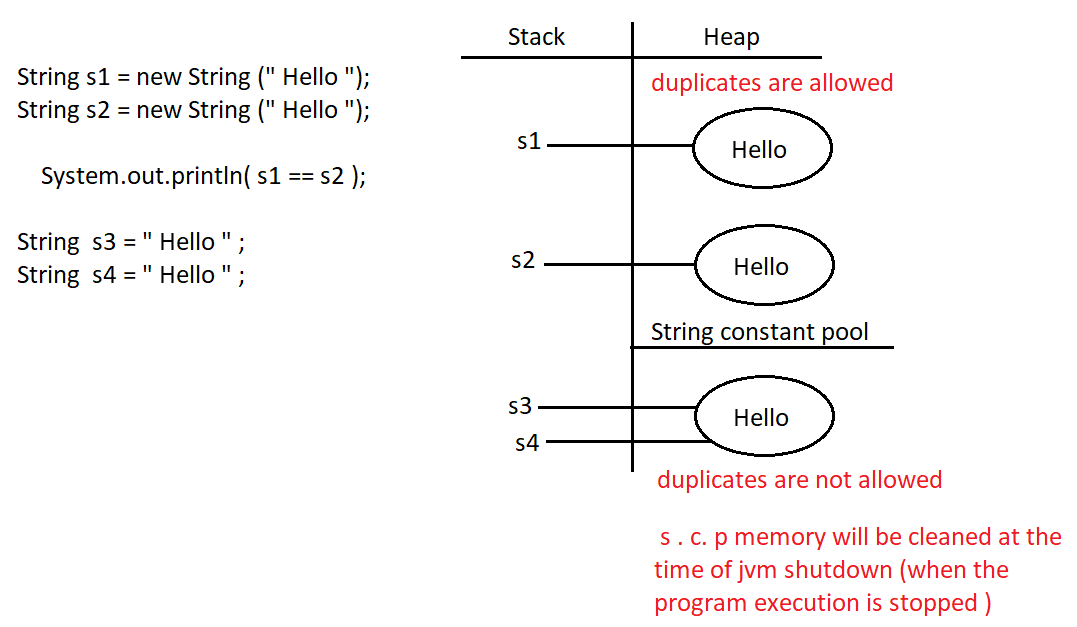
In this case two object will be created one in the heap, and the other in the String constant pool ( s . c . p ) The reference always points out to the heap.

String s2 = “Hello”;

In this case one object will be created in the string constant pool and it points out to the reference



Eg: Strings\_Eg6



Explanation:

Here an object “hello” is created in the heap and its reference point to s1 in stack, in s.c.p also an object is created with same data. And it will have no reference but garbage collector cannot delete it.

In heap area duplicates are allowed, so another object of “hello” is created in heap and points to s2, but duplicates are not allowed in the in the s.c.p, since s1 and s2 has object with same data jvm will not create another object in s.c.p.

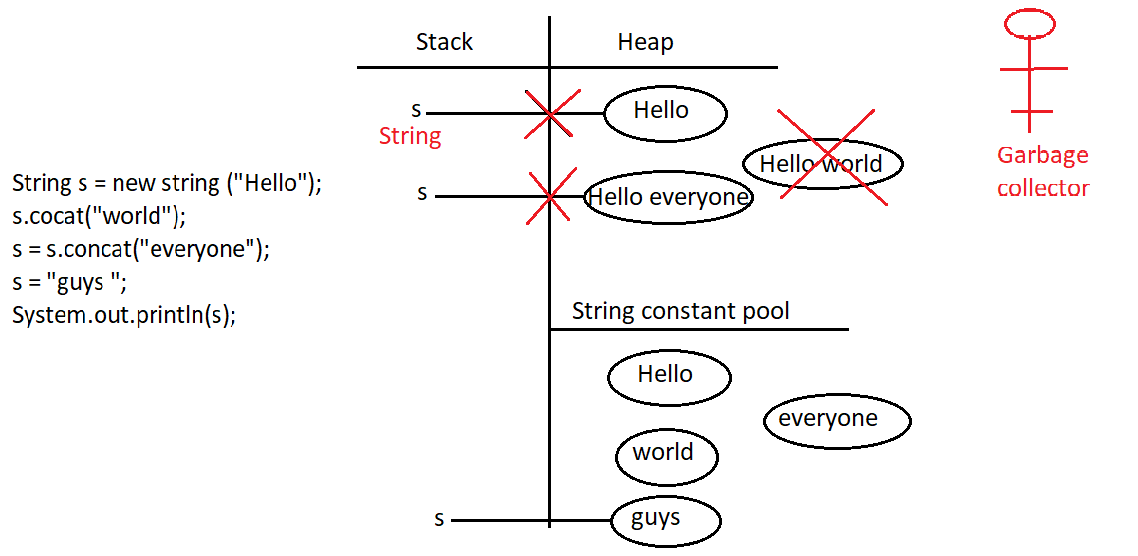
If new keyword is not used object is created in the s.c.p and it points to reference in stack. Now for s3 there is an object with same data available in s.c.p so jvm simply points out that object to s3.

Similarly for s4

s.c.p memory will be cleaned only when program execution is stopped (jvm shoutdown).

So if compared s3 and s4 ( s3==s4 ) output will be true . since same reference variables and same object.

Eg: Strings\_Eg7



Explanation:

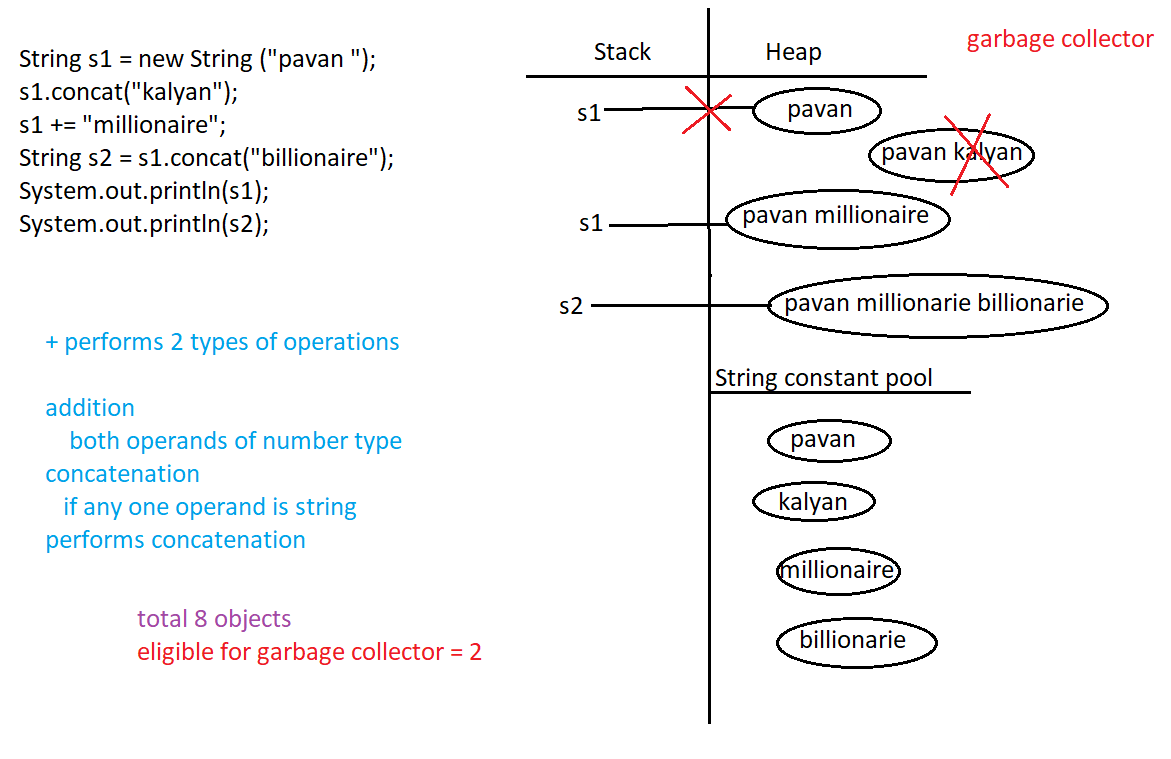
When line 1 is execute, since there is new keyword the object is created in the heap area and the s.c.p.

In line 2 since literal is used, object is created in the s.c.p,here concat method used on s invoked by jvm during runtime. because of runtime operation if an object has to be created, it will be created in heap area .since there is no reference variable to collect it , it is cleaned by the garbage collector .

In line 3 again, since literal is used object will be created in the s.c.p , and we are performing operation during runtime so it will be created in the heap area , and we are collecting in the reference variable .

In line 4 literals are used, so object will be created in the s.c.p, and same reference s is used to store the object . so s will be pointed to the object with guys data. The reference “s” pointing to the object with data “Hello every one” will be replaced with the object with data “guys” and garbage collector will remove it unreferenced object.

Eg: String\_Eg8

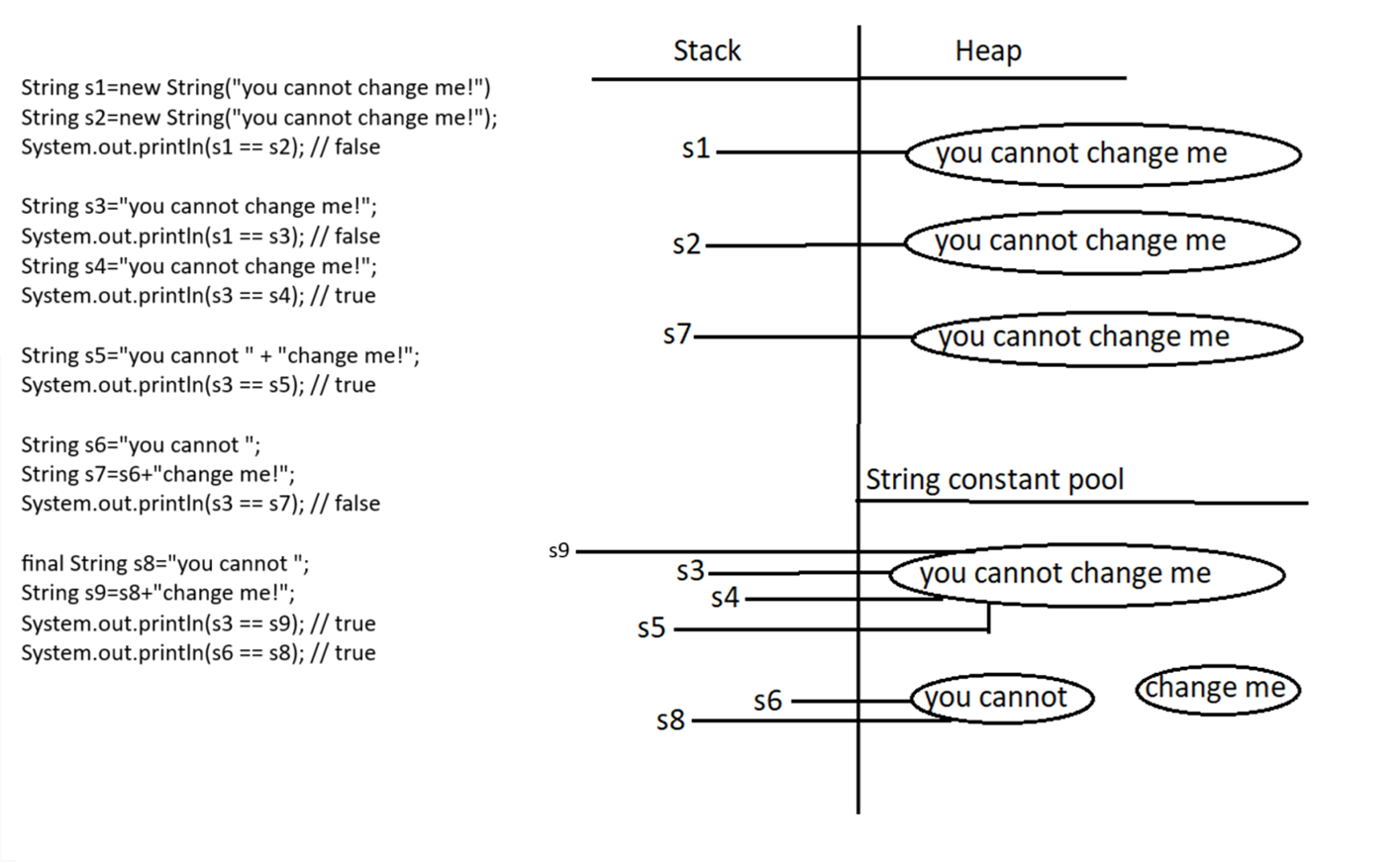


Explanation:

Process is same as previous one.

In line 3 operand is involved that operation is done at runtime, and since operation is done at runtime, jvm creates the memory for it in the heap area.

Eg: String\_Eg9



Explanation:

Process is same

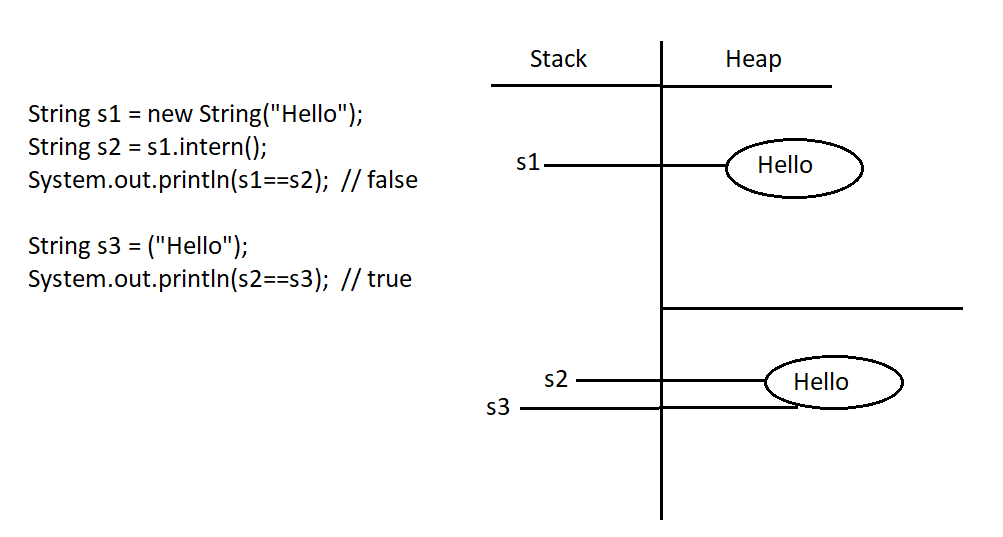
In string s5 since both are String literals, they are created in the string constant pool even though there is a concatenation. since there are no variables are operations (methods) to do at the runtime, their evaluation is done at the compile time, and after concatenation memory is also allocated at the string constant pool. since there is already similar data object, it is allocated to s5.

In String s7, literal is create in the s.c.p and since variable/operand is involved, it is evaluated during runtime by jvm, since runtime operation is involved their memory is allocated in the heap area.

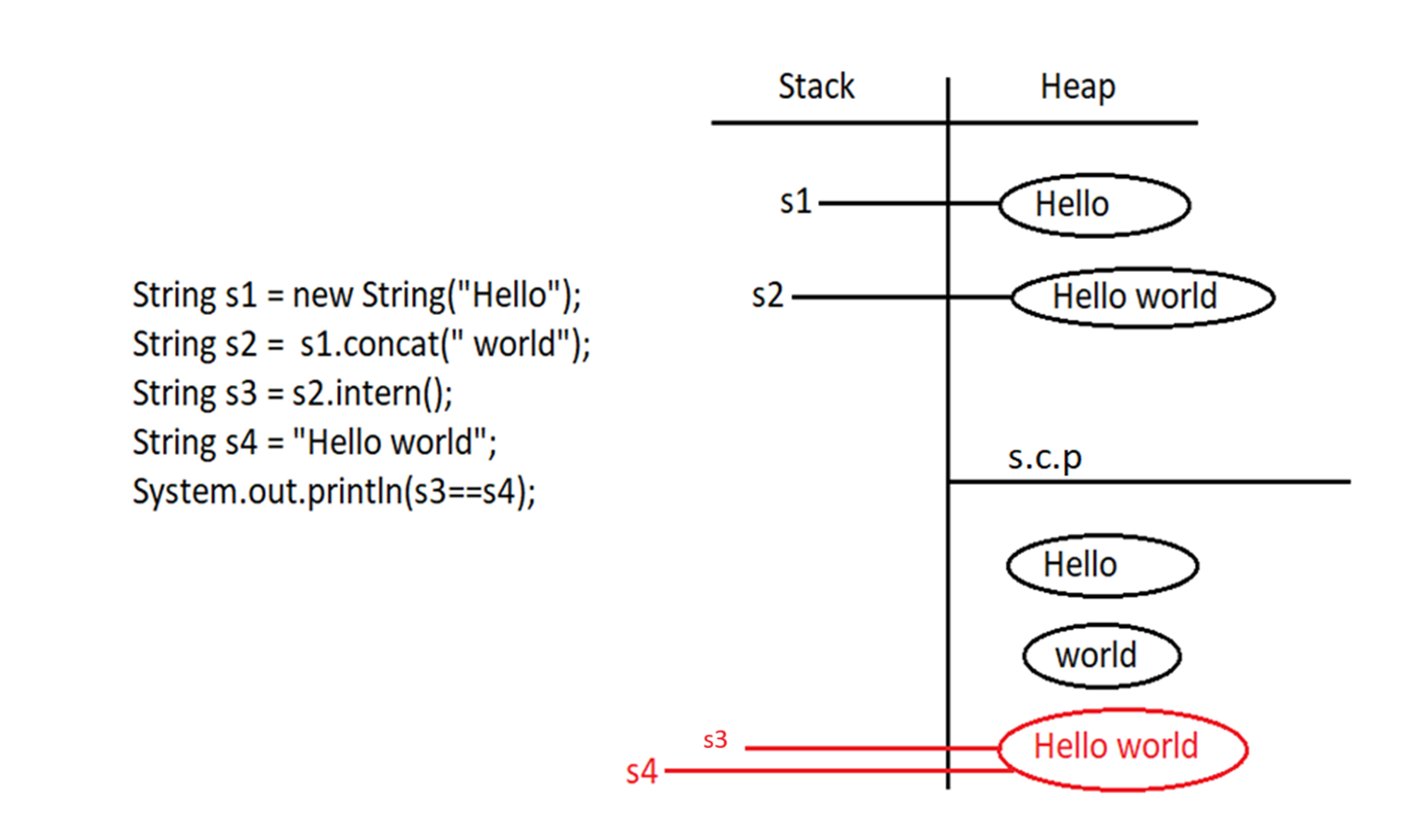
In String s9, since s8 is a final variable it value is known to the compiler, and during compile time it is evaluated and “change me“ is a literal ,memory for it is allocated in the s.c.p . And after concatenation of s8 and literal memory for s9 is allocated in s.c.p , since there is no runtime operation involved . Their concatenation result is already present in the s.c.p it is pointed to the s9.

Eg: String\_Eg10

intern() The .intern() method creates an exact copy of a string located in the heap memory and stores it in the string constant pool.



Eg: Strings\_Eg11



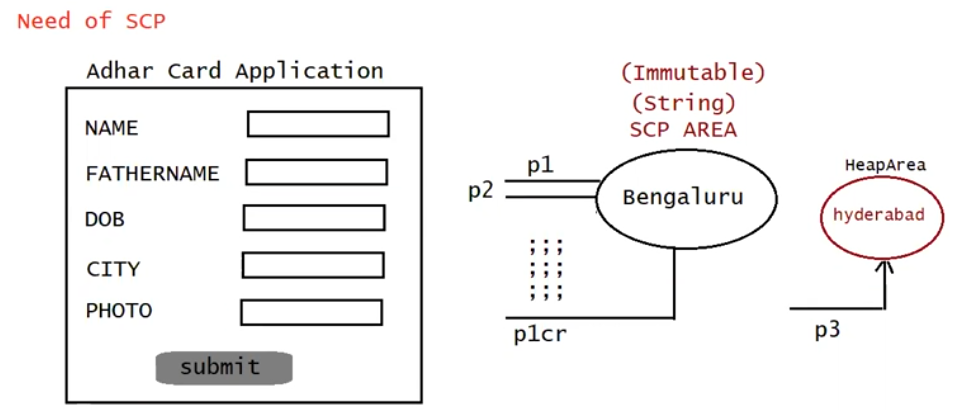
Note:

Importance of String constant pool

================================

1. In our program if any String object id 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 performances and memory utilization we can achieve this by using s.c.p
3. In s.c.p several references pointing to same object and 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 change a new string object will be created hence immutability is the main disadvantage of s.c.p .

Example :



Explanation :

Here the applicant can have city bengaluru similar to him many people lives in the city of bengaluru, creating separate objects for each and every person in not memory efficient and performance of program might also come down. here is the place where s.c.p uses, where a single copy of object can be referred to multiple people. (bengaluru city object can be used by multiple people without creating new one for every person.

But there might be a case where the person changes the city (“hyderabad”), and it should be reflected in the card, since String in immutable, it can’t be changed, a new object will be created and referred to it like above diagram . if there is already an object created with “Hyderabad”, it is referred to it.