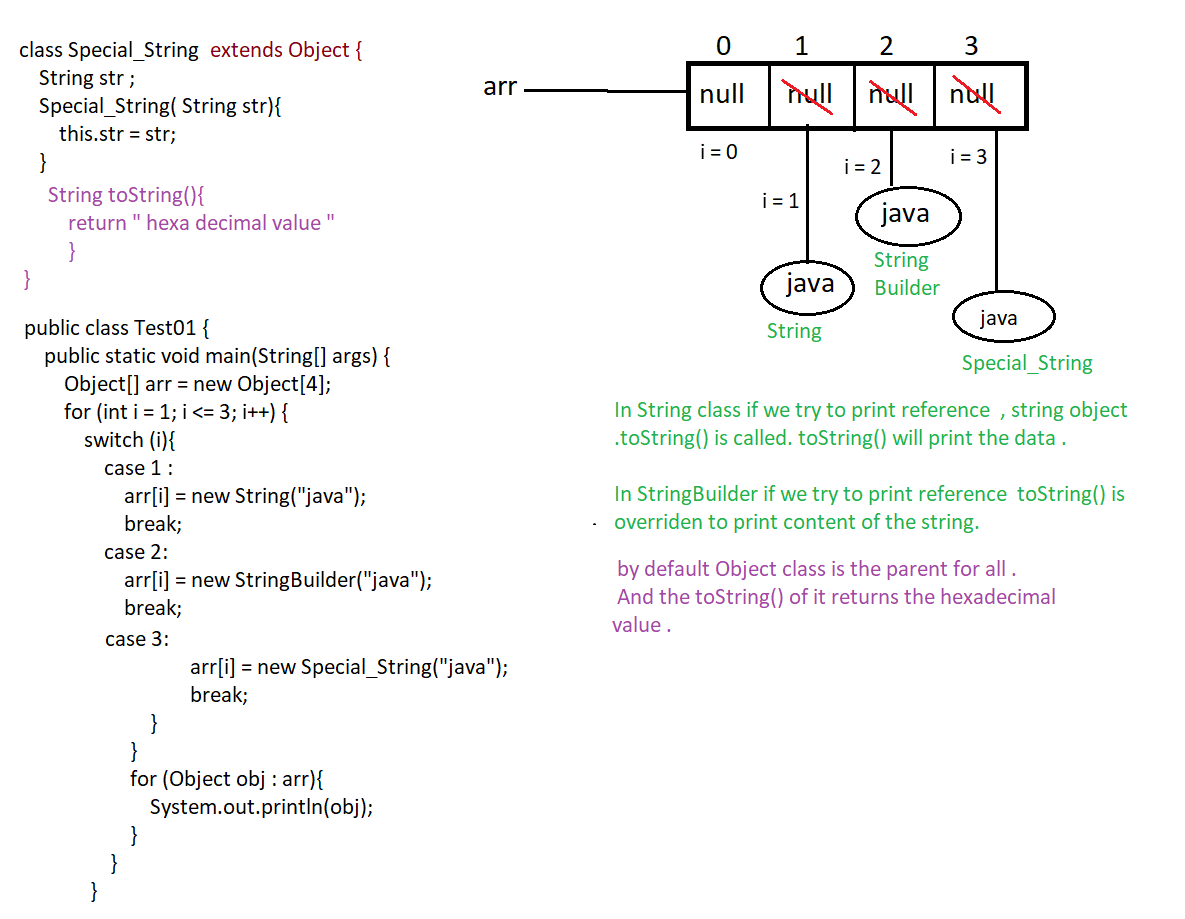
Eg: String\_Eg76

When you try to print object reference ,internally .toString() is called on it to print the reference . but manually we had written toString() which returns null value . so it is added to “text” variable .

Eg: String\_Eg77

// go through the code , it is based on above program String\_Eg76 .

Eg: String\_Eg78



Eg: String\_Eg79

// go through the command .

Eg: String\_Eg80

// go through the code.

Eg: String\_Eg81

// go through the code.

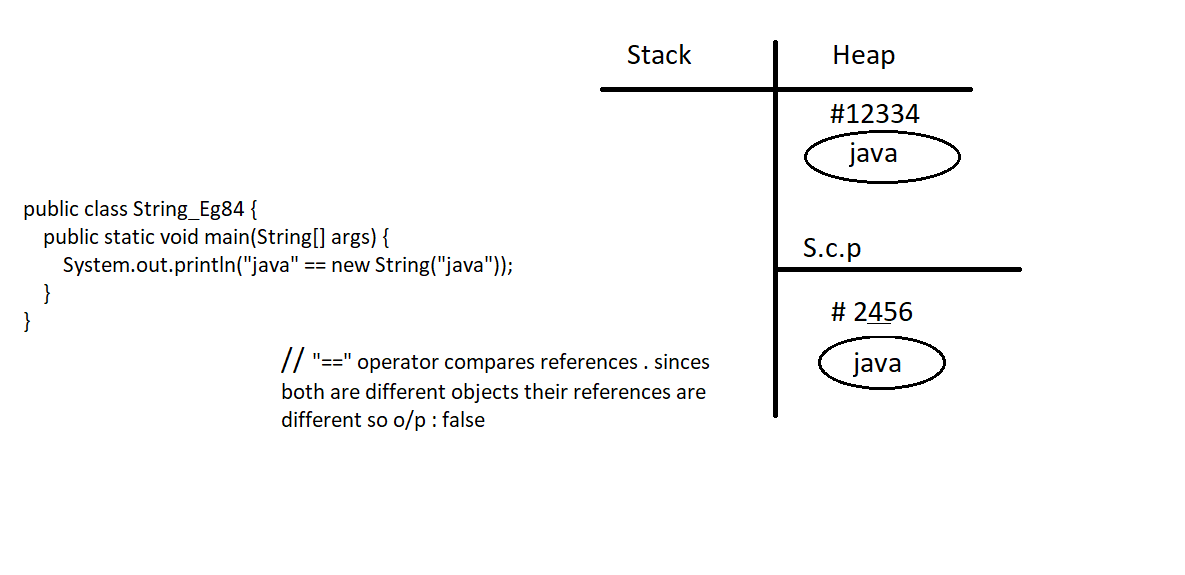
Eg: String\_Eg82

// go through the code.

Eg: String\_Eg83

// go through the code.

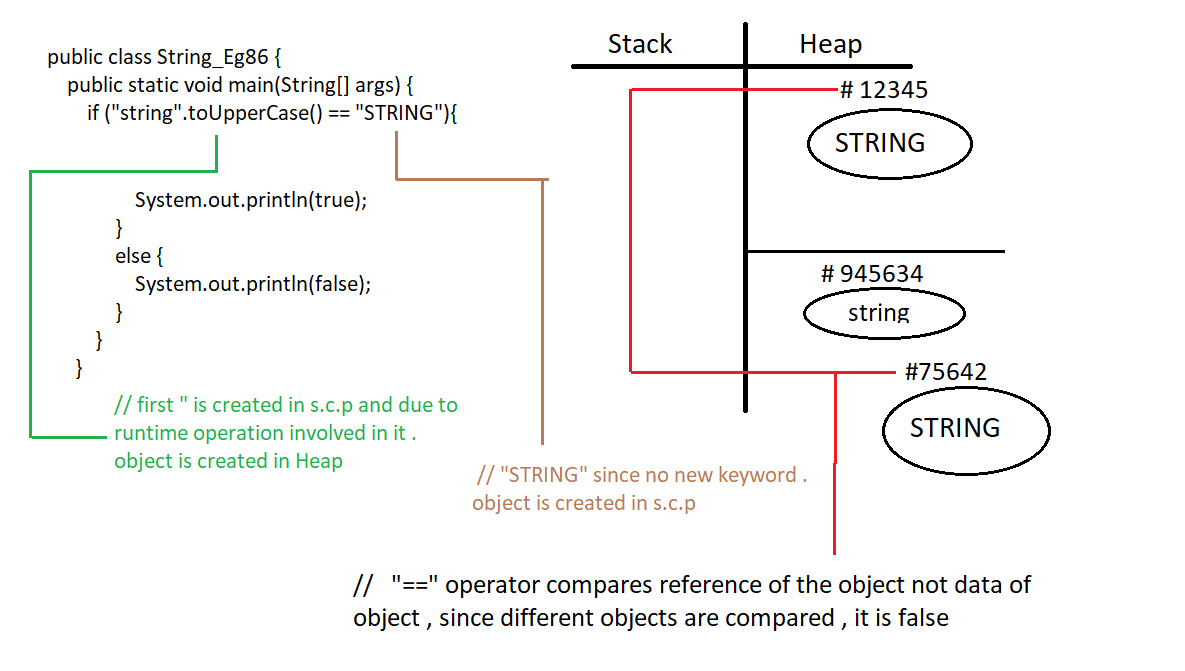
Eg: String\_Eg84



Eg: String\_Eg85

// go through the code

Eg: String\_Eg86



Note : String , StringBuilder , StringBuffer all these classes are final by default.

Eg:String\_Eg87

// go through the code.

Note: When the string concatenation is very frequent and code is need not be thread safe , we can opt for StringBuider(1.5v)

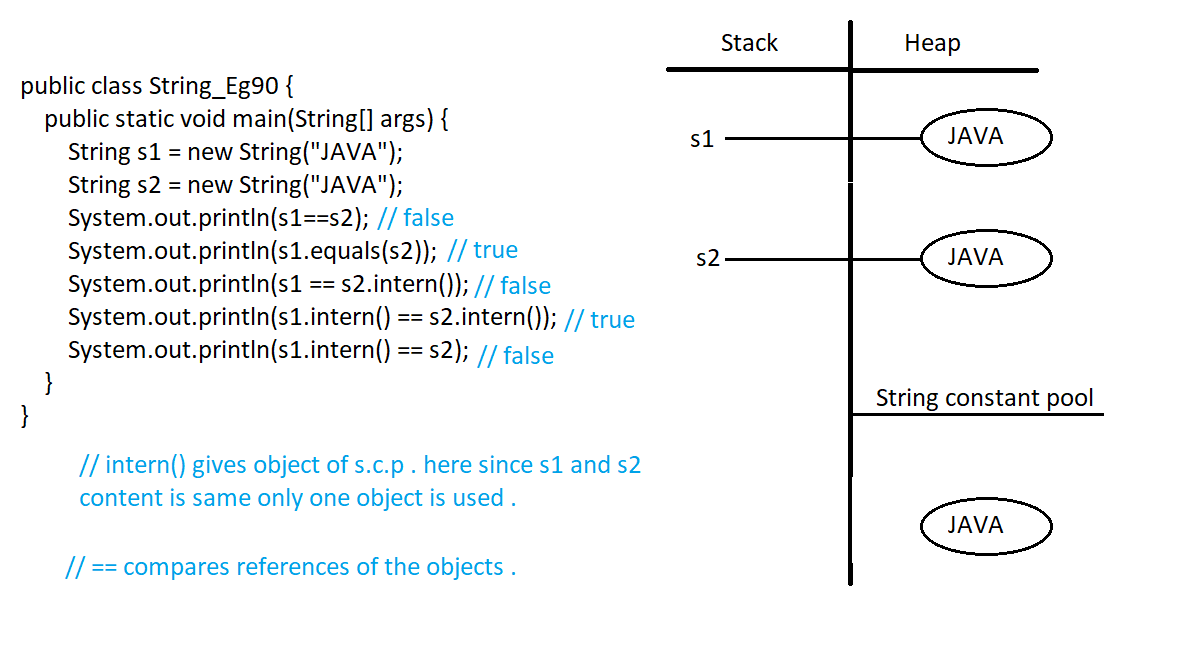
Eg: String\_Eg88

//go through the code.

Eg: String\_Eg89

// go through the code

Eg: String\_Eg90

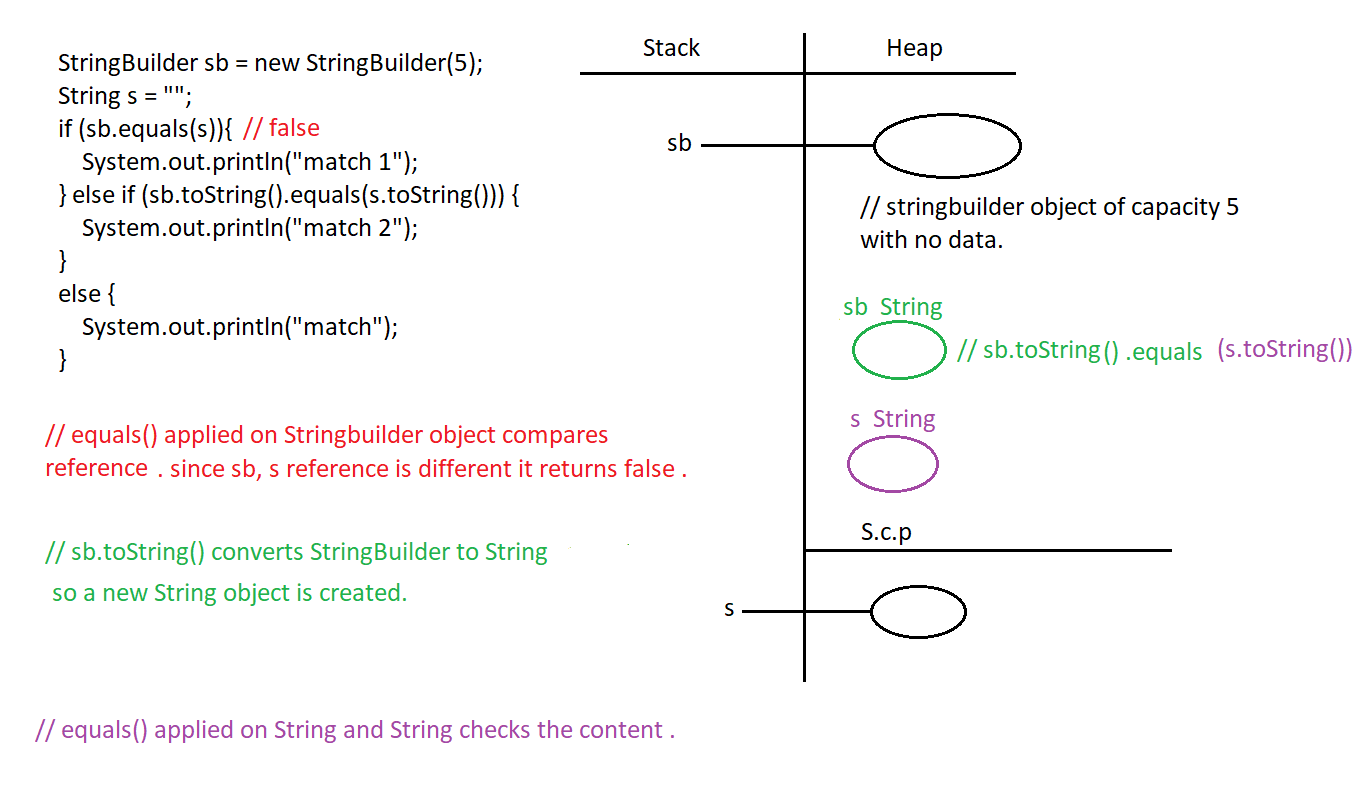


Eg: String\_Eg91

// go through the code .

Note : StringBuilder( ) constructor different types of parameters , check it in the IDE.

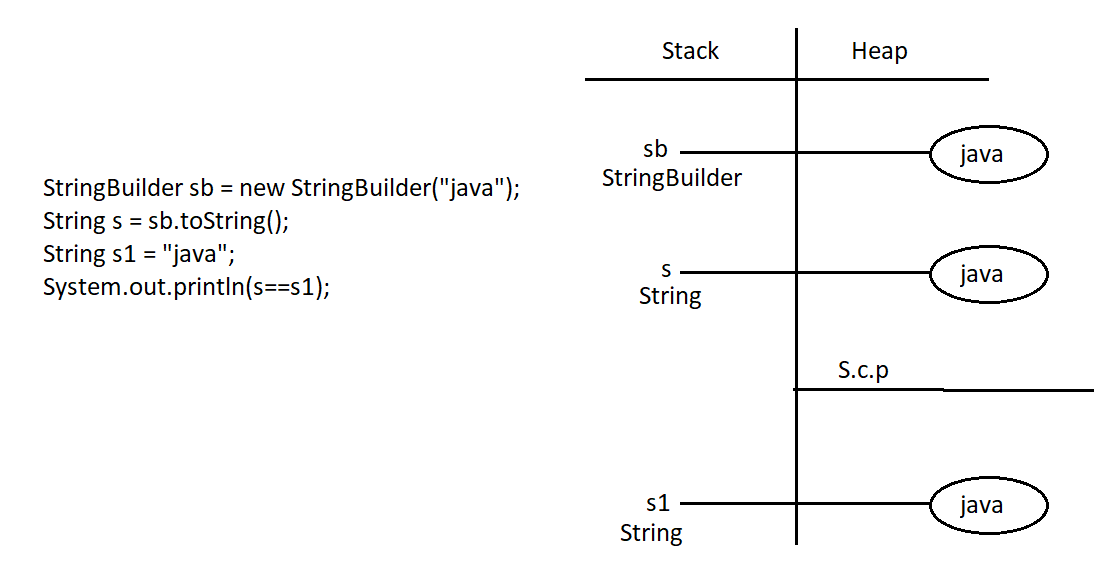
Eg: String\_Eg92



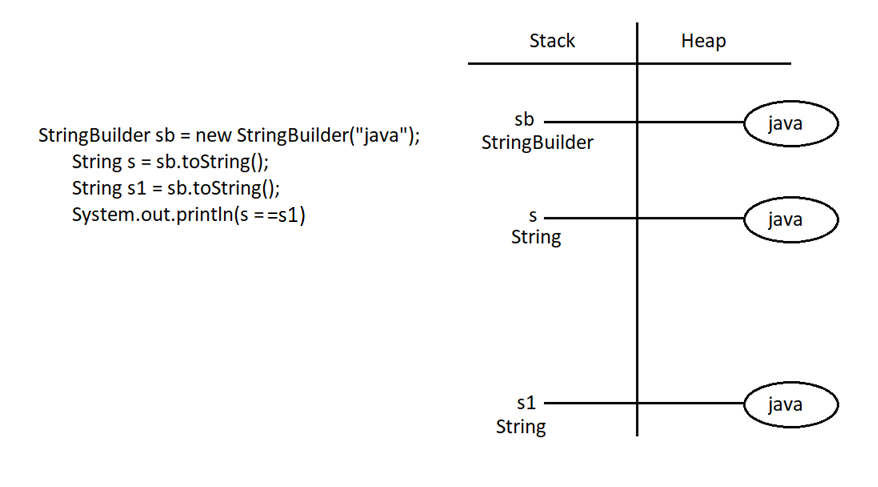
Eg: String\_Eg93

// go through the code.

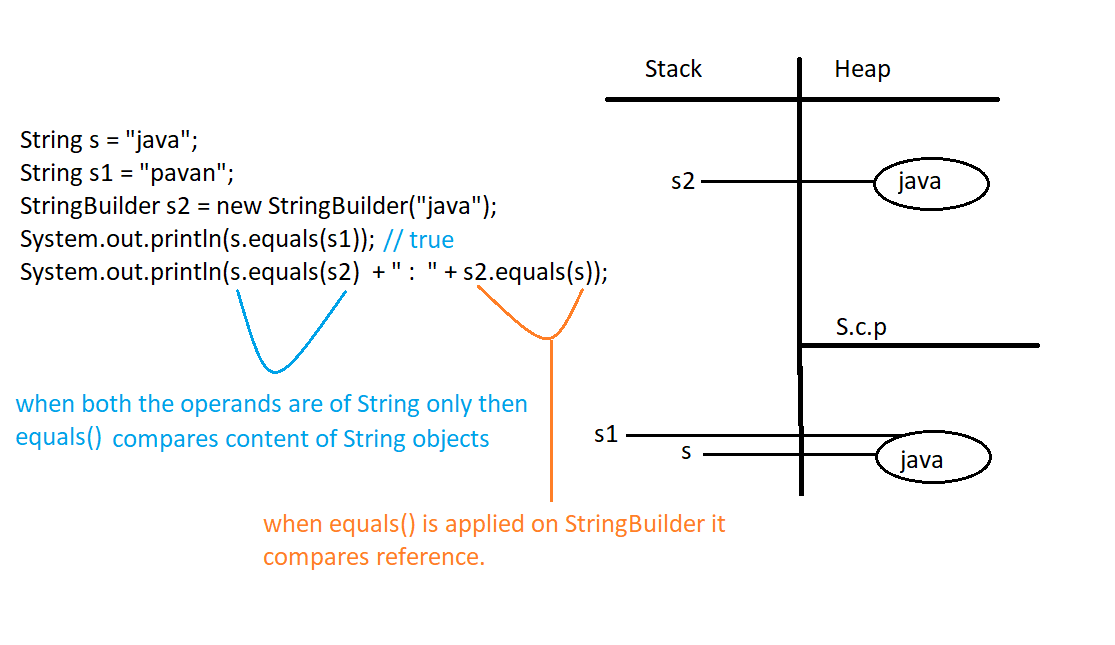
Eg: String\_Eg94



Eg: String\_Eg95



Eg:String\_Eg96



Eg:String\_Eg97

// go through the code.

Eg: String\_Eg98

// go through the code

Eg:String\_Eg99

// go through the code

Eg: String\_Eg100

// go through the code

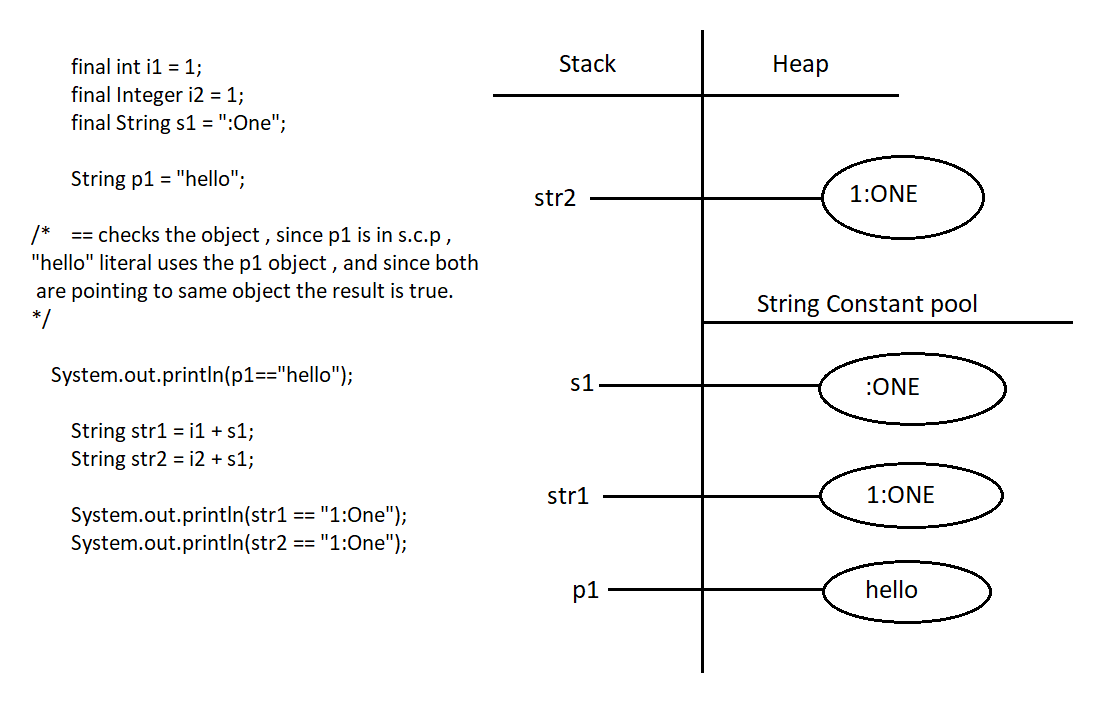
Eg: String\_Eg101

// go through the code.

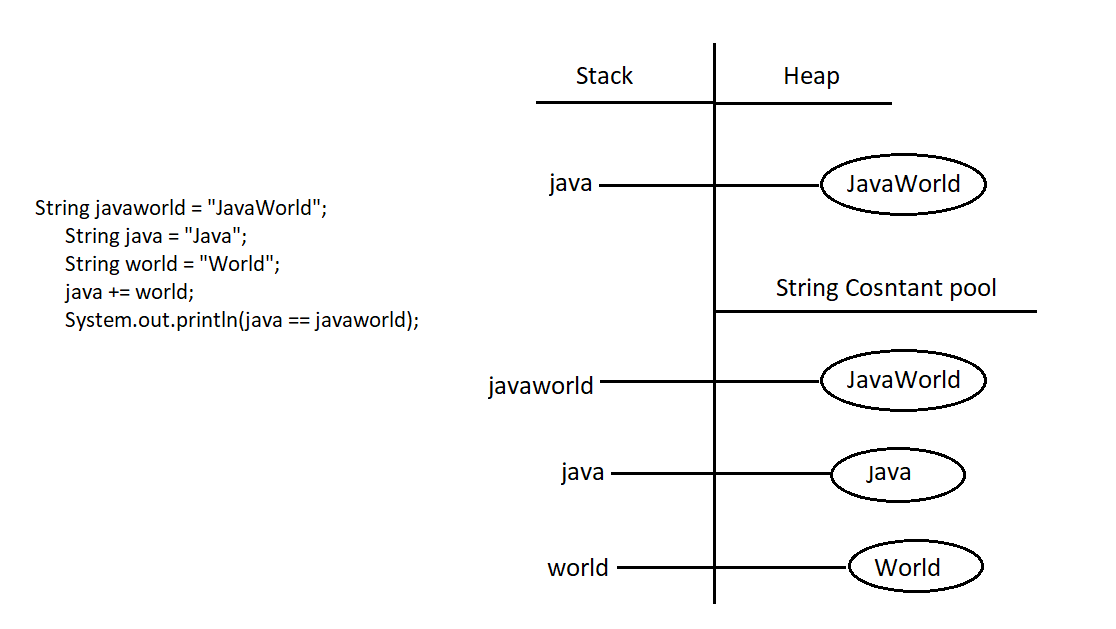
Eg: String\_Eg102

String is not wrapper class it is a Object .

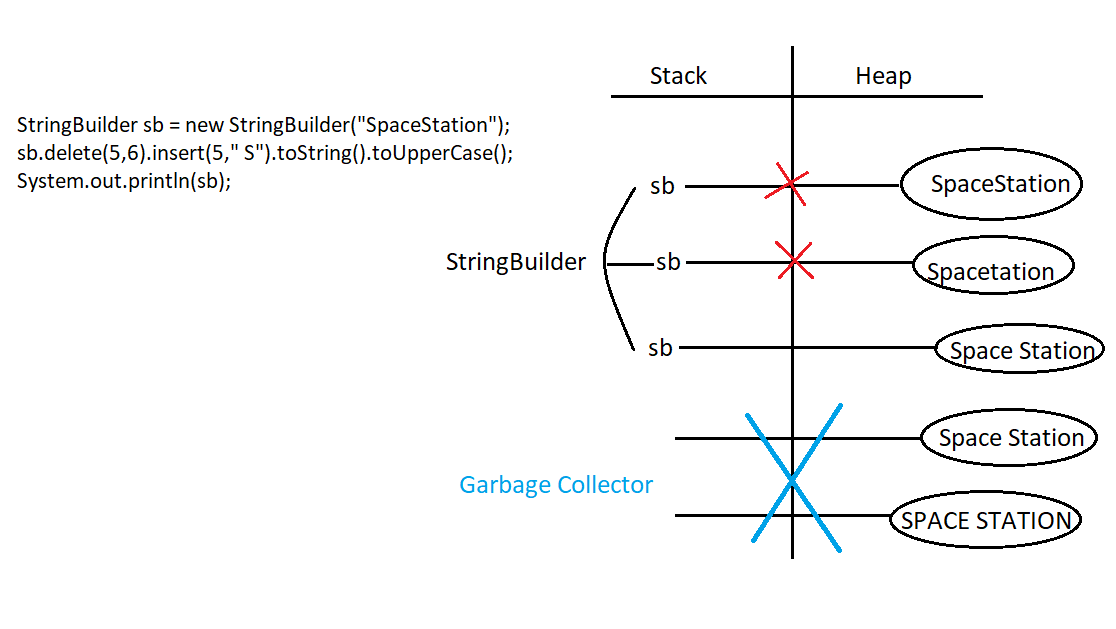
Here i2 is a wrapper class , even though wrapper classes is final memory for them is given at the runtime . if run time operation is present jvm is involved , and memory for them is allocated in the heap.



Eg: String\_Eg103



Eg: String\_Eg104



Here for StringBuilder sb same object is updated , if we try to update it / make changes to it . no new object is created .

But in in String a new object is created if we try to make a change / update , and if the updated one is not collected using variable , garbage collector will clear it .

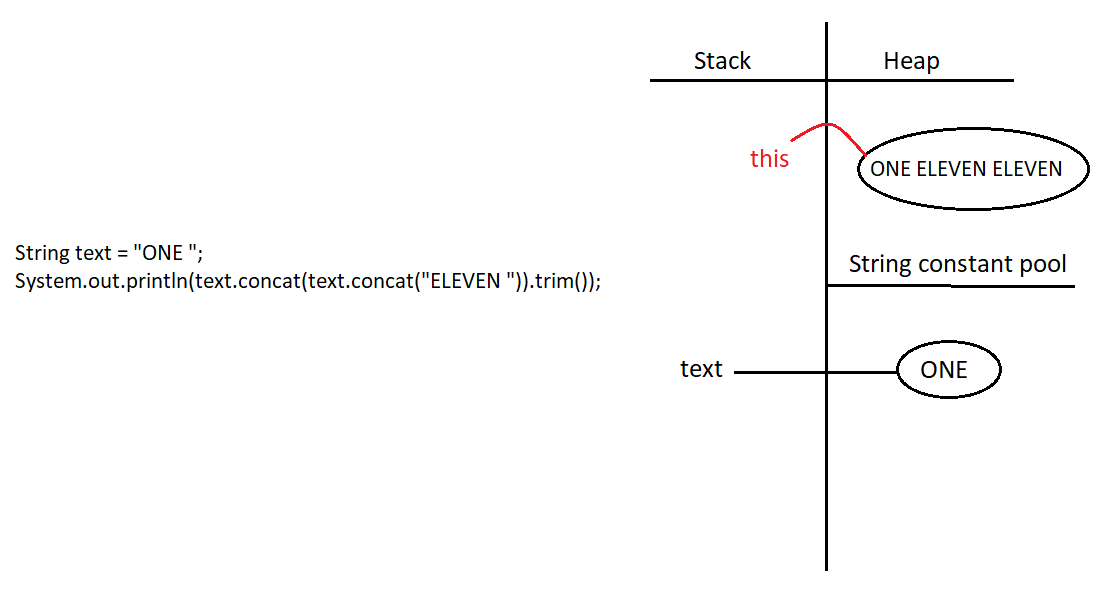
Eg: String\_Eg105

// go through the code

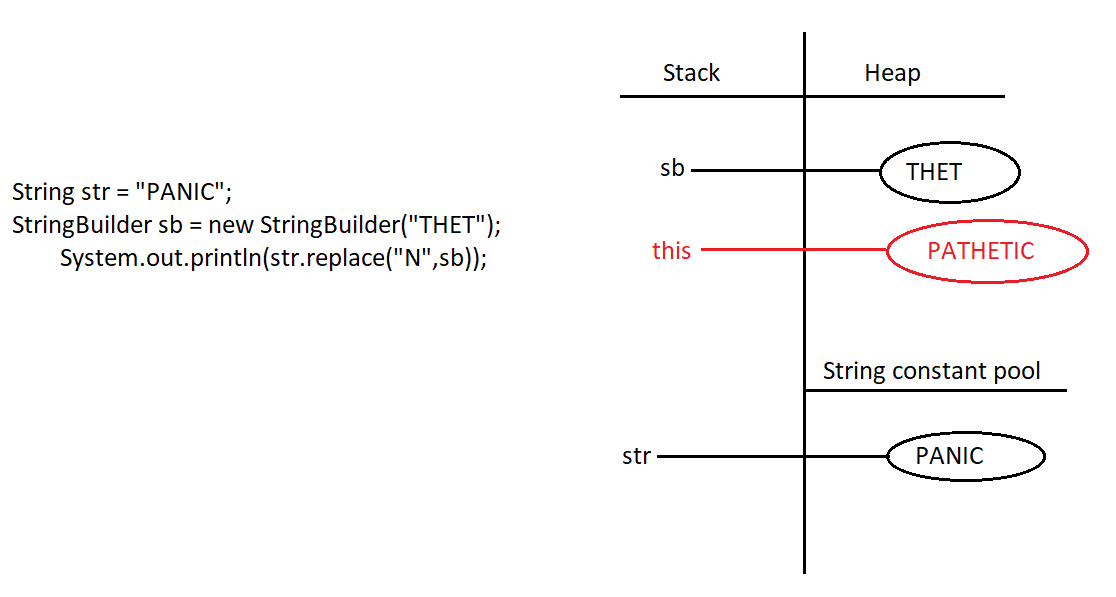
Eg: String\_Eg106

In the heap area when there is no reference and you are trying to use that , jvm maintains the reference using “this” keyword .

Current object will always be pointed by two references , one is our object reference , and the other is jvm internally maintained object reference through “this” keyword .



Eg: String\_Eg107



Eg: String\_Eg108

// go through the code

Eg: String\_Eg109

// go through the code

Eg: String\_Eg110

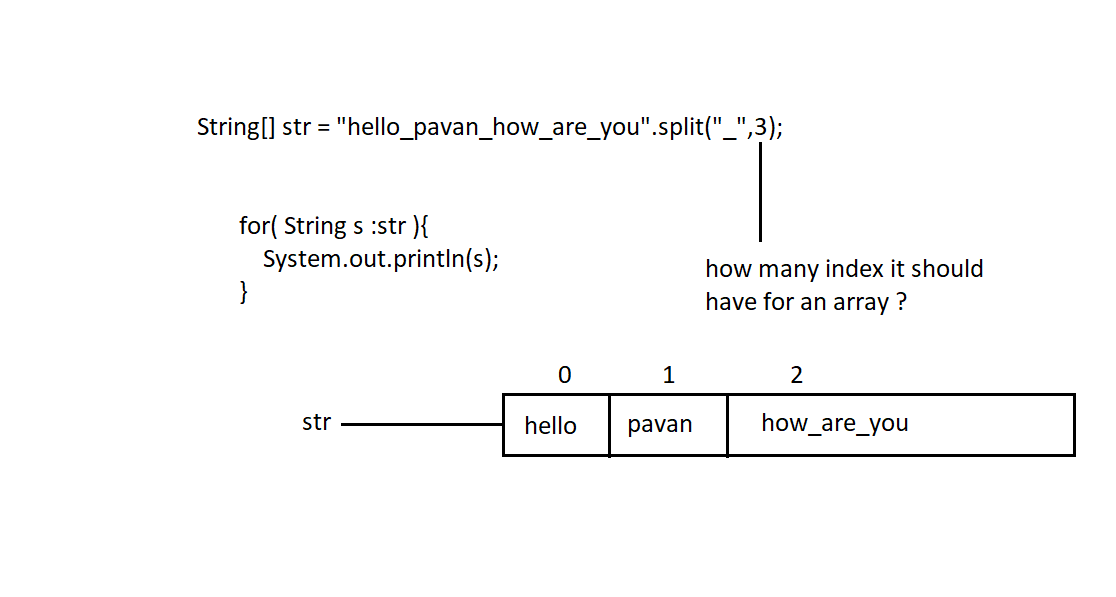
indexOf() method the index of string object starts from 0 . 1 is added to it so 2. charAt() method index also starts from 0 so chartAt(2) = > V

Eg: String\_Eg111

// go through the code

Note : methods available in StringBuffer is also available in StringBuilder.

Eg: String\_Eg112



If we don’t give array size , wherever “\_” is found in the string object , the array will have that many no of index .

Note : java.lang.String class implements the following interfaces

Serialzable

CharSequence

Comparable ( check this when interface concept completed ) .