API ( application programming interface )

Some one wrote the code and he will give .class file end users will use and take the benefit

Java community wrote the code gave the String .class file to us , and we are using it . Entire java we are learning as API only .

javap java.lang.String shows method of the string

Note: if the class name and method name is same , then it is called constructor.

Some of the String class constructors are

String s = new String(); // creates empty String object

String s = new String (“ ”); // creates an object with String literals on heap and s.c.p

Eg: String s = new String (“hello”);

String s = new String(StringBuffer sb); // creates an equivalent string object for string buffer String buffer is converted to String .

Eg: String\_Eg14

String s = new String (char[] c); // creates equivalent string object for character array

// Eg: String\_Eg12

String s =new String (byte[] b); // creates equivalent string object for byte array

// Eg: String\_Eg13

Note : since String is immutable , if a change is made in runtime space (object) for it will be created in the heap.

The above concept can be said as constructor overloading . same name but different parameters

Some Important methods of String :

1. public char charAt(int index)
2. public String concat(String str)
3. public boolean equals(Object o)
4. public boolean equalsIgnoreCase(String s)
5. public String subString(int begin)
6. public String substring(int begin , int end)
7. public intLength()
8. public String replace(char old, char new )
9. public String toLowerCase()

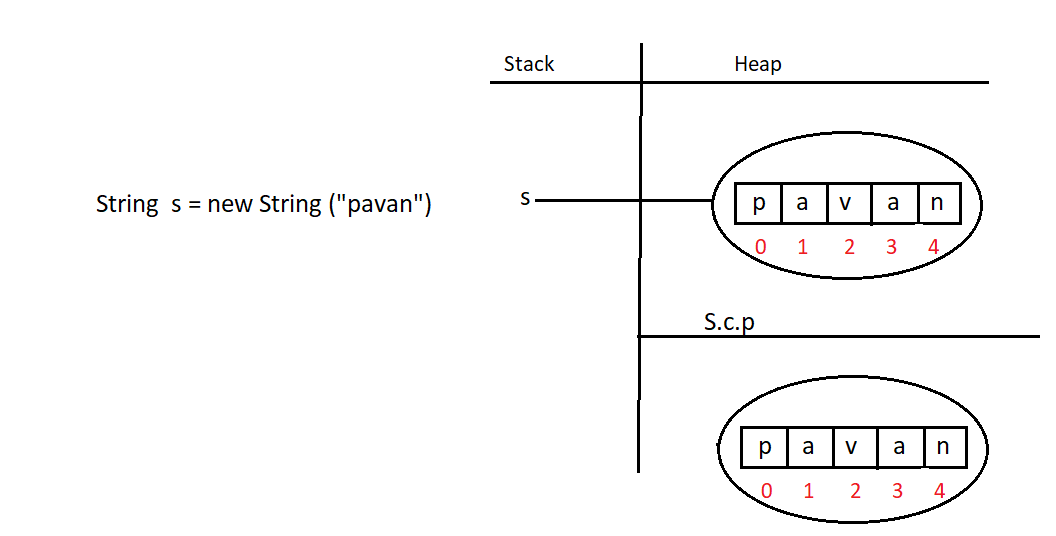
10.public String toUpperCase()

11. public String trim()

12. public int indexOf(char ch)

13. public int lastIndexOf(char ch)

14 . public String toString()



Even though the String is an object internally it is stored as array .

But if we try to access that array

System.out.println(s[3]) // compile time error

We cannot access them . it is only for memory level .

But we can access them as an array using charAt() method .

Eg: Strings\_Method\_Eg15

// go through the program.

Eg: String\_Eg16

Here length() is a method of the String class , that can be accessed with String object . Even though String internally uses array we cannot use array property for it

Eg : String s = new String (“hello”);

s.length();

length is a property of Array class , array class is only for java (check previous notes on arrays )

class [I {

//property name

int length;

}

Eg: String\_Eg17

concat() is a method of String class

String object by default is immutable (changes will reflect in new memory ) .

Eg: String\_Eg18

// go through the program

Assignment :

username : [mohanpavankalyan52@gmail.com](mailto:mohanpavankalyan52@gmail.com) (not case sensitive )

password : \*\*\*\*\*\*\*\*\*\* (case sensitive)

equalsIgnoreCase() method is used in usernames of many websites , example : gmail . Even though if we give case sensitive data it is ignored . only data is checked in username here .

But in password equalsIgnoreMethod is not used mostly .

Eg: String\_Eg19

public String subString(int begin)

public String substring(int begin , int end)

They are example of method overloading.

Eg: String\_Eg20

replace() method changes the specific character .

if there are multiple occurrences of the character all of them would be replaced.

Eg: String\_Eg21

during runtime if object is created in the heap and if there is no reference , it will be on the heap until execution of that particular line is completed ( just like toUpperCase() in the print statement) . after that garbage collector will clean that .

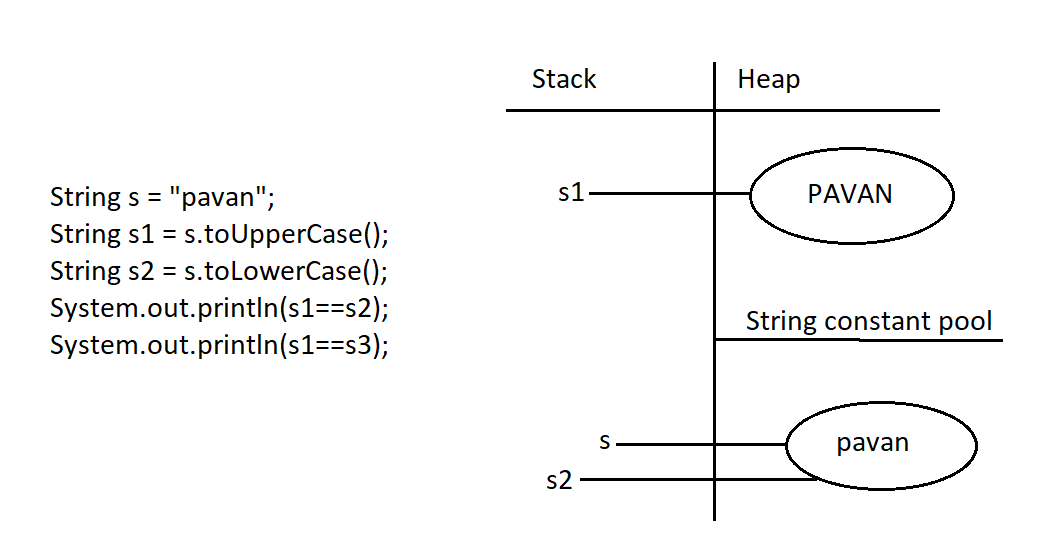
Eg: String\_Eg22

trim() method is removes the blank spaces at the starting and at the end , but not at the middle of the string.

Eg: String\_Eg23

// go through the program

Eg: String\_Eg24

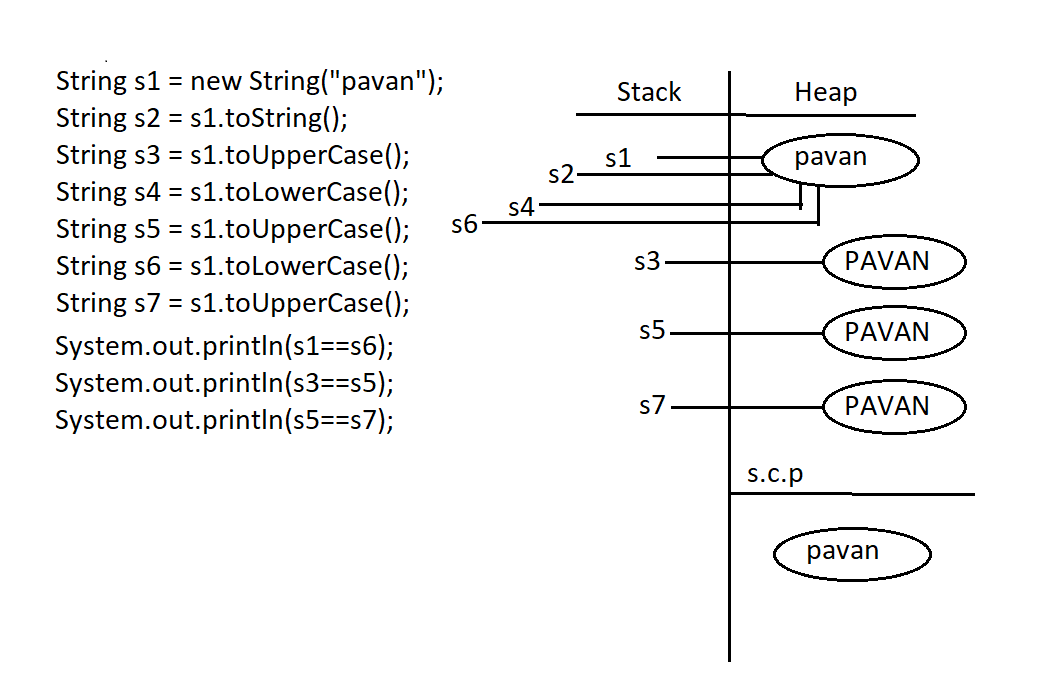


Even though runtime operation is involved , since there is already same data in s.c.p it is pointed to s2 .

Eg: String\_Eg25

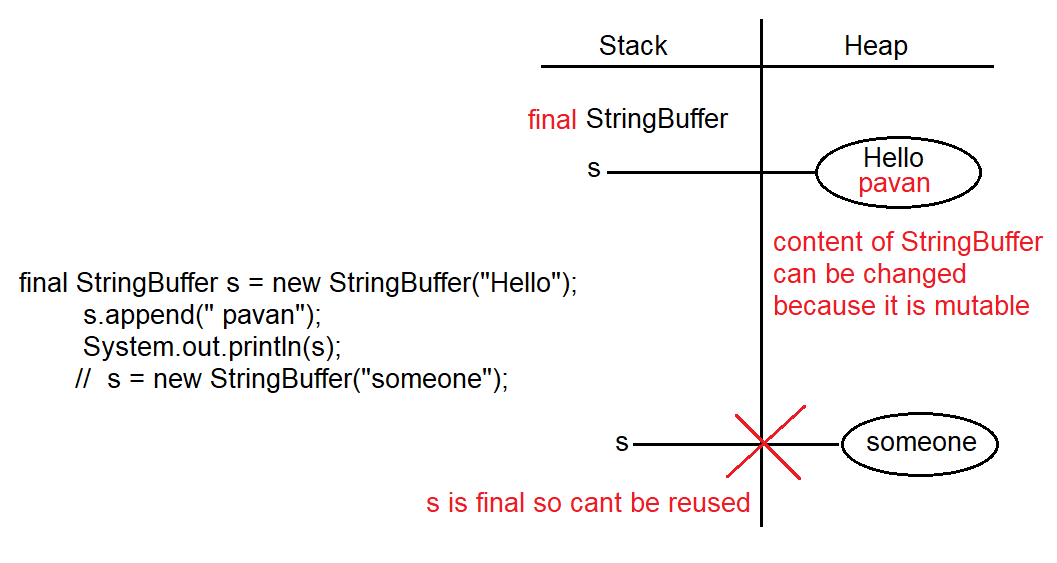
// whenever we try to print any reference , by default jvm will call toString() on the reference

Eg: String\_Eg26



In heap area you can assign multiple references only to String containing new keyword. In this case we had assigned multiple references ( s2,s4,s6) to s1 since same data . s3 , s5, s7 even though runtime operations are performed and data is also same multiple references cannot be assigned to the same object as they are not created using new keyword .

Eg: String\_Eg27



final vs immutability

* final modifier is applicable for variables , where as immutability is applicable only for objects
* if the reference variable is declared as final , it means you cannot perform reassignment for reference variable , it doesn’t mean that we cannot perform any change in that object.
* By declaring reference variable as final we wont get immutable nature
* final and immutability concept is different

StringBuilder, StringBuffer and Wrapper Classes (Byte , Short , Integer , Long , Float , Double , Character ,Boolean ) are by default mutable .

whatever methods present in StringBuilder are also present in StringBuffer also.

StringBuffer :

If the content change frequently then it is not recommended to use String object because for every new change a new object will be created

To handle this requirement we need StringBuffer/StringBuilder

To check StringBuffer package use javap java.lang.StringBuffer

Constructors of StringBuffer :

StringBuffer s = new StringBuffer();

Creates an empty StringBuffer object with default intial capacity 16

Once StringBuffer object reaches its maximum capacity a new StringBuffer object will be created

new capacity = (current capacity +1 ) \*2;

Eg: StringBuffer\_Constructor\_Eg28

// go through the program

Eg: StringBuffer\_Constructor\_Eg29

StringBuffer s = new StringBuffer(10) // here 10 is the capacity

If the object reaches the maximum capacity ,

new capacity = (total current capacity +10) here 10 is user defined size during above declaration .

Eg: StringBuffer\_Constructor\_Eg30

StringBuffer s = new StringBuffer(“h”)

Step-1 : The capacity is 16 (standard value) + 1 (since one character “h” )

When object reaches maximum capacity ,

new capacity = (total current capacity ) \*2

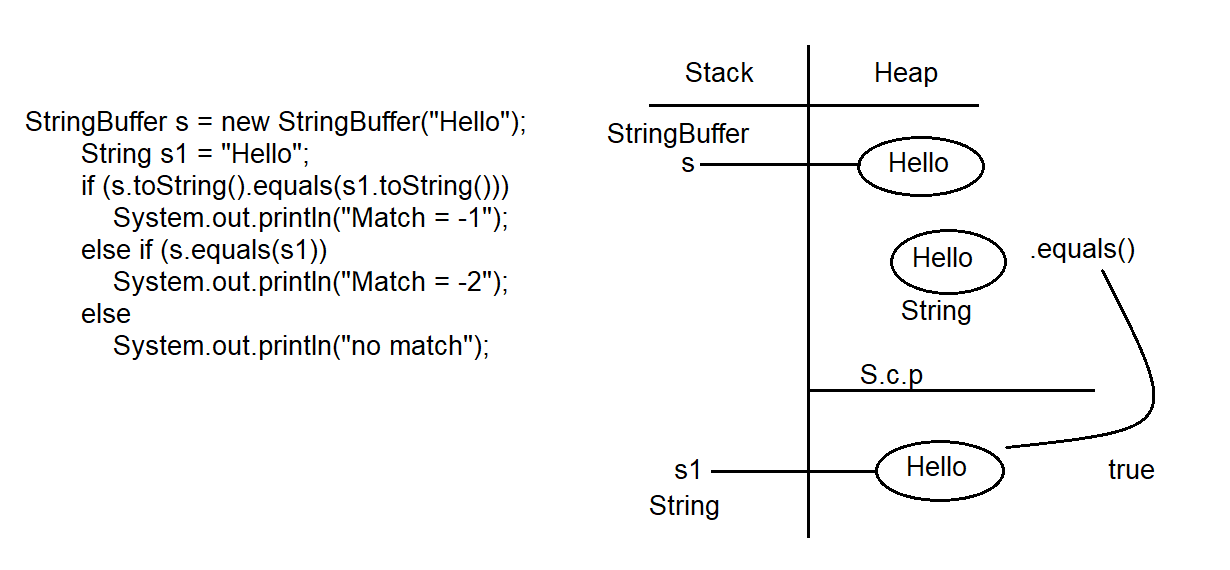
Eg: String\_Eg31

// go through the program

Eg: String\_Eg32

// go through the program

Eg:String\_Eg33



Here toString() is a String method , we are converting StringBuffer to String , even though same data but since conversion is done from StringBuffer to String a new object will be created in heap.

And on s1 we are performing toString() , even though runtime operation , there is no change in the data and no change in type , no new object will be created . A new object is created , When there is change in data during or change in the type , or both during runtime .

Eg: String\_Eg34

// if the first operation in the expression is String then all the remaining expression is concatenated .

// go through the code .