     

**JAVA : STRING STRING BUFFER AND STRING BUILDER**

equals()

String is used for content comparison

String buffer is used for reference comparison

String

The most commmonly used object is the string object

What is the difference between string and string buffer ?

String objects are immutable and string buffer objects are mutable

String s = new String ("Yash");

s.concat("Jha");

Sopln(s); The output will be yash . As strings objects are immutable . S.concat will create the new object which has the value of yash jha but we dont have any reference variable to store it so it will be stored in the garbage value and our s will still point to yash and the output will be yash only . So we try to do any changes in the object it will create the new object .

StringBuffer sb = new StringBuffer("Yash");

sb.append("Jha");

Sopln(sb);

Here the output will be yash jha as string buffer objects are mutable i.e. it is changeable thus the output will be yash jha .

Note : 📝 in StringBuffer we cannot use concat and in string we cannot use append .

== operator vs equals method

== is used for reference / address comparison if both reference variable point to the same object then it returns true.

String s1 = new String ("Yash");

String s2 = new String ("Yash");

Sopln(s1==s2); it will generate false as both the reference variable are not ponting to the same object .

Sopln(s1.equals(s2));here we will get true as bith objects content are same. ( it is so because the equals method is overriden in string to check the content and not the address or reference of the object )

equals () method is also used for address / reference compariosn but in string it is overriden for content comparison and string buffer it is not overriden so it is used fir reference/address comparison only .

StringBuffer sb1 = new StringBuffer ("Yash");

StringBuffer sb2 = new StringBuffer ("Yash");

Sopln(sb1==sb2); it will generate false as both the reference variable are not ponting to the same object .

Sopln(sb1.equals(sb2)); Here we will get galse as in StringBuffer the equals method is not overriden and it is used ti check the address / reference of the object.

Heap and String constant pool(SCP)

String s = new String ("Yash"); here two objects will be created the first will be strored in the heap area i.e. yash and its reference variable will be s and the second object will be created for yash as its a literal so its compulsory that its ibject will be cretaed. It is created as it will be used in future reference and it will be stored in string constant pool and string constant pool is available in heap area only . Until version 1.6 it was stored in method area / permanent generatin(PERMGEN) from version 1.7 onwards it was stored in heap area for memory utilization . And the second object will dont any explicit reference varaibke pointing to it rather it will have a implicut reference variable which will be maintained by JVM. Thus the second object will not be available for garabage collection as it has a implict reference variable.

String s = "Yash" ; here only one object will be created in the scp area . I.e. s-> Yash . So the JVM will check if the object is

Created is avaible in the heap area and scp or not if it is not available then only it will create a new object . Otherwise it will reuse the object .

String s1 = new String ("Yash");

String s2 = new String ("Yash");

String s3 = "Yash";

String s4 = "Yash";

How many objects will be created in the heap and scp area ?

So here total 3 objects will be created as we have used new keyword two times two objects will be created in the heap area with the reference variable s1 and s2 respectively also when we have created the first object the Yash will also will have been stored in scp area for future reference so in next step s3 and s4 will not create any ither object as the JVM checks that yash object is already available so no need to create that object so yash will be pointed by s3 and s4 .

So whenever we are using new keyword then it is compulsory new object will be created in the heap area though they have the same content in them . But in scp area there is no chance that the two objects will have the same content . As same object will be used multiple times .

String s = new String ("Yash");

s.concat("software");

s = s.conact("solutions");

Here total 6 ojects wil be created.

1. Heap area will have one object yash and scp will have object yash so two objects got created.

2. s.conacat have software as literal so it will create object in scp . So till now it has 3 object on total also string objects are immutable so any changes made will create a new object so another object will get created in the heap area also it happens due to runtime operations . as software as there is no reference variable associated with it so it is available for garbage collection.

3. Here also two objects will get created one in scp as it is a new literal i.e. solutions and also string objects are immutable so it will create another object having a reference variable s and now s will point to Yashsolutions . So in total 6 objects got created.

SCP PART 3

In the above photos ,

Constant + constant will generate result in compile time only

Constant + variable will generate result in runtime operation only so a new object will be created.

So s6+ "change me " will be performed at runtime so a new object will be created. So s7 object will be created in heap area .

s8 is associated with final variable and its not a normal variable so its value will be replaced in compile time only. Every final variable value will be repalced in compile time only.

So s9 will be same as s5 only.

Importance of SCP

It provides reusability and by this memory utilization is also there . Suppose there are 1 crore people in hyderabad who are voting with the help of scp we do not require to create 1 crore objects for people living in hyderabad . By creating 1 object the 1 crore people can reference to the same object . But there is a problem if a perosn wants to changes his place from hyderabad to vihayawada all 1 crore references will be affected as they all are pointing to hyderabad. So for this problem java people came with an idea which is immutablity which makes the strings immutable and when we assign new value to it creates new objects so if a person3 wants to change his place to Vijayawada his reference will only change

rest other references will remain same .

Important FAQS on string and string builder

Why scp concept is available for string object and not for string buffer ?

Suppose you have a habbit of going to bar and you go to bar everyday and drink something . One day you left the wallet at your home by mistake . But the bar owner knows you as you are his regular customer so he will say you can pay the money later . But when a new customer foregts his wallet the bar owner will look top to bottom at that customer . This happens with string as string objects are the most commonly used objects than StringBuffer as we may not use StringBuffer or even we can avoid using that but string objects are used most of the time.

Why string objects are immutable and string buffer objects are mutable ?

Because of scp string objects are immutable. As any peroson who wants to change the object value will ulitimateky create the new object without affecting the other references whereas in string buffer there is no concept of scp so everytime new object will be created even if the object has the same conetent .

Inaddition to string objects are there any objects immutable in java ?

All wrapper class objects are immuatable in java .

Important constructors of string class

length variable var7aboe available for array not fir string objects for string length() method is available.

s.indexOf() it returns the index value of the string . If the specified character is not available in the string it will return -1. If there are repeated characyers present in the string it will return he value of the first occurrence of the character present in the string.

s.lastIndexOf(char ch) returns the last occurred character in the string.

The trim() method is used to trim or to remove the blank spaces present at the beginning and at the end of the string but not at the middle of the string.

String s1 = new String ("yash");

String s2 = s1.toUpperCase();

String s3 = s1.toLowerCase();

Sopln(s1==s2); it will return false as we have made changes in the object by converting it into uppercase thus a new object is Created.

Sopln(s1==s3); it will return true although we have converted it into lower case but the object is still the same so no new object will be created. Thus s3 also points to the same object to which s1 points. And thus it returns true.

If the content is fixed it advisable to go for string as if it is not fixed for every new content a new object will be created which will decrease the performance of the program.

If the content keeps on changing its advisable to go for StringBuffer . In this all changes will be made in same(existing)object only.

String Buffer Constructor

1. StringBuffer sb = new StringBuffer();

Its default intial capacity is 16 i.e. it can accommodate 16 characters. If we add the 17 th character its size will increase.

Capacity = (CurrentCapacity + 1)\*2

So in this case capacity will be = (16 + 1 )× 2 = 34

StringBuffer sb = new StringBuffer();

Sopln(sb.capacity()); o/p = 16

2. StringBuffer sb = new StringBuffer(int intialCapacity);

Ex : StringBuffer sb = new StringBuffer(1000);

Sopln(sb.capacity()); o/p = 1000

IMP 3. StringBuffer sb = new StringBuffer(String s );

So here the capacity = s.length() + 16(default capacity)

4. StringBuffer sb = new StringBuffer(1000);

    sb.append("ABC");

     Sopln(sb.capacity()); o/p =1000

    sb.trimToSize();

Sopln (sb.capacity()); o/p = 3

this helps in deallocating the extra 997 unused memory. As we have allocated 1000 memory spaces and we have used only 3 the trimToSize method helps in deallocating the memory.

Need for StringBuilder

StringBuffer: Every methid inside a StringBuffer is synchronized. So at a time only one thread is allowed to work on StringBuffer object which slows down its performance and it increases the waiting time of the thread.

To verify that you can check in the compiler by writing

javap java.lang.StringBuffer.

So to conclude StringBuffer does not work for multithread environment.

When they introduce the concept of StringBuilder java people just replace StringBuffer with StringBuilder and removed synchronized from every method . So inshort non synchronized method of StringBuffer is called StringBuilder.

String vs StringBuffer vs StringBuilder

String : when the content is not changing it is highly advisable to go for string and its thread safe which means only oj e thread can operate it at a time and we cannot perform modification on it . Also every immutable objects in java is thread safe as whenever we will try to modify the object it will create a new object .

StringBuffer: it is used when the content is keep on changing i.e. it is not fixed . Also it is thread safe .

StringBuilder: it is used when the content is keep on changing and when we don't want thread safety.

Method Chaining

In method Chaining all the method calls will be executed from left to right.