Immutable string: cannot be changed

**package** com.cts.thisandsuper;

**public** **class** B {

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

String s1 = "vengat"; //M1 -> s1

String s2 = "prabu"; //M2- >s2

String s3 = "vengat";//M1->s1, s3

System.***out***.println(System.*identityHashCode*(s1));

System.***out***.println(System.*identityHashCode*(s2));

System.***out***.println(System.*identityHashCode*(s3));//op address will be same as address of s1

//changing the value of s1 using concat

s1 =s1.concat(s2); // changing the value of s1 will also change the memory location of s1

//M1 - >s1 will be removed and s3 will be there still

///M1 -> s3

//M2->s2

//M3 -> new memorey will contain vengatprabhu(s1)

//So when u concatenate a literal string, the concatenated value will take a new address

System.***out***.println(System.*identityHashCode*(s1));//different memory address will be printed

System.***out***.println(System.*identityHashCode*(s3));

System.***out***.println("==============");

StringBuffer x1=**new** StringBuffer("vengat");

StringBuffer x2=**new** StringBuffer("prabu");// mutable string

System.***out***.println(System.*identityHashCode*(x1));

System.***out***.println(System.*identityHashCode*(x2));

x1 = x1.append(x2);

System.***out***.println(x1);

System.***out***.println(System.*identityHashCode*(x1));//same memory will be printed as in line 25

}

}

|  |
| --- |
|  |
|  | String buffer | String builder |
| 1) | StringBuffer is *synchronized* i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously. | StringBuilder is *non-synchronized* i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously. |
| 2) | StringBuffer is *less efficient* than StringBuilder. | StringBuilder is *more efficient* than StringBuffer. |

String class contains a method called concat() to join two strings.

StringBuilder and StringBuffer doesn’t have concat method. Instead it has a method called append();

String builder and buffer performance example:

1. **public** **class** ConcatTest{
2. **public** **static** **void** main(String[] args){
3. **long** startTime = System.currentTimeMillis();
4. StringBuffer sb = **new** StringBuffer("Java");
5. **for** (**int** i=0; i<10000; i++){
6. sb.append("Tpoint");
7. }
8. System.out.println("Time taken by StringBuffer: " + (System.currentTimeMillis() - startTime) + "ms");
9. startTime = System.currentTimeMillis();
10. StringBuilder sb2 = **new** StringBuilder("Java");
11. **for** (**int** i=0; i<10000; i++){
12. sb2.append("Tpoint");
13. }
14. System.out.println("Time taken by StringBuilder: " + (System.currentTimeMillis() - startTime) + "ms");
15. }
16. }

Wrapper class: OOPs – In java we cant write any program without a class. Then we will have methods and objets.

Java is not fully OOPS based. It is almost 95% oops based.

Remaining 5% - ?

Datatypes – int, float, double – Keywords

String – class ->methods – indexOf, charAt() – u can create an object for String class.

But for int, float - u cant create an object

To overcome this limitation – we go for wrapper classes

Int -> Integer

Float – Float class

In simple - > advantage of wrapper class is u can convert primitive datatype into OBJECTS

Condtional blocks and loops:

Conditional blocks -> if else, nested if else, switch case

Loops: for, while and do while

For loop task:

Get a string input from user

Print all the characters in the string 1 by 1 in reverse order

Eg:Sathish - >

H

S

I

H

T

A

S

Same task with while loop also

Arrays:

Group of values that belongs to same datatype and is stored in single variable is called as array.

Task:

Get a Sting input from user

Get a char input from user

String contains the character or not – contains() – don’t use contains method

To charArray - use

“Sathish”

‘s’

It is present

It is not present