Why string is immutable?

Immutable means which cannot be changed. In Java, there are certain things which are immutable to increase the program performance.

Look the following code:

String str=”Hitesh”;

String str1=” Hitesh”;

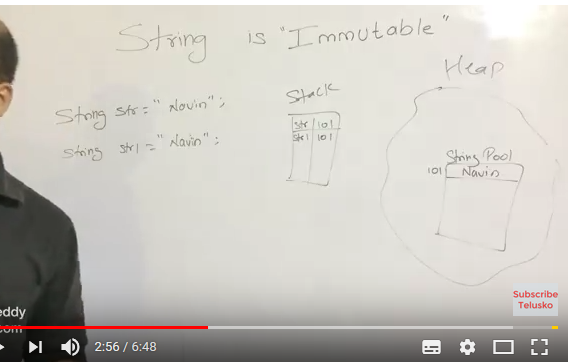
**Now, let’s first understand the storage of above objects in memory**

We have **heap memory area** and inside it we have a **‘String pool’** where the above object “Hitesh” will be stored in some address say ‘101’ where both the references i.e. “str” & “str2” pointing to this address i.e. 101.

Now, we also have a stack area where only the object references along with addresses of string pool in heap area are stored. In above cases, the object references are ‘str’ & ‘str1’ and they both are pointing to ‘101’ address in string pool area in heap memory area.

Now here comes the concept why the strings are immutable. Now, if we try to string value of ‘str’ reference variable then there is risk of impacting the value of ‘str1’ reference variable which should not be happened at any cost. So that’s the reason, strings are immutable in java.

Look at the following screen grab:



Earlier we have the following string values

String str=”Navin”;

String str1=” Navin”;

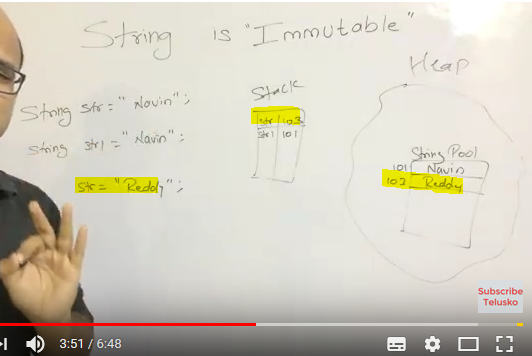
Now we change the value of “str1” reference variable as follow:

Str=”Reddy”;

Now, in that case, earlier value i.e. “Navin” will be not changed by “Reddy ” value but in fact another object i.e. “Reddy” will be created in “String pool” area. And address of “str” reference variable will be updated with address of “Reddy” object from string pool area. Hence, we can say string in java are immutable.

So every time, we change the value of ‘str’ , we are actually creating a new object in string pool area and updating the reference address in stack area.

Now, in that case the memory structure will be updated as below;

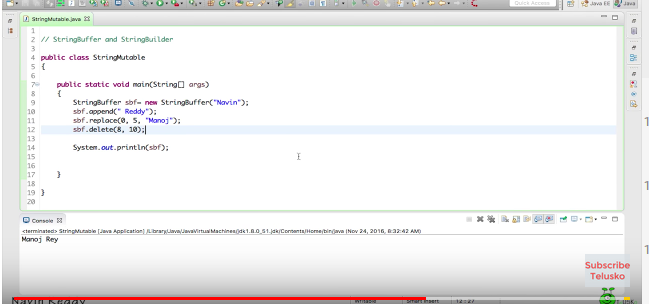


**String immutable concept is also known as “Flyweight” concept where two references using the same reference (hence encourages ‘reusable’ concept)**

**Summary: Since strings are immutable, so string old value cannot be replaced by new value, but new value (object) will be created.**

**String mutability** but now question how we can achieve the string mutability.

We can achieve the string mutability with the help of ‘StringBuffer’ and ‘StringBuilder’ classes



**What is the difference between** ‘StringBuffer’ and ‘StringBuilder’ classes/

Ans StringBuffer’ provide thread safe but the stringbuilder.