Wednesday, July 15, 2020 1:01 PM

Strings In java

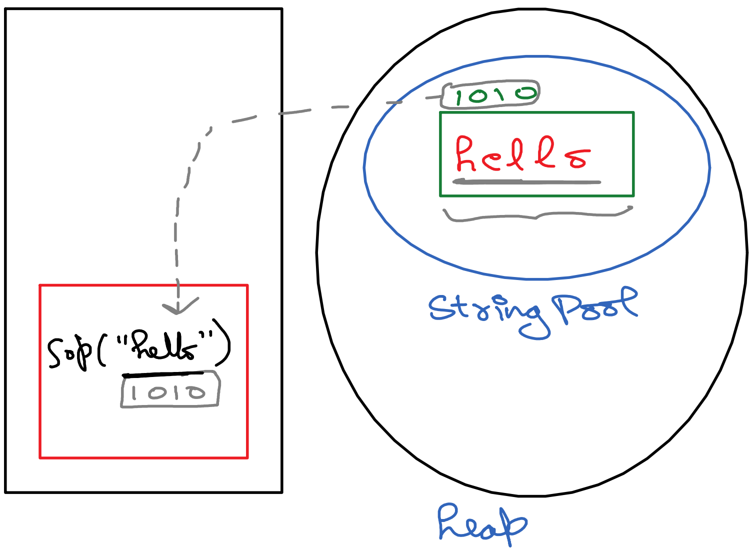
1. String is a literal(data), it is a group of characters enclosed within double quotes "string".
2. String literal is non-primitive type.
3. In java we can store String literal by creating an instance of the following classes:
   - java.lang.String ✓
   - java.lang.StringBuffer
   - java.lang.StringBuilder

→ Built-In classes

> Note:
>
> In java, every time when a String literal is created, implicitly the instance of **java.lang.String** class is created in String pool area.

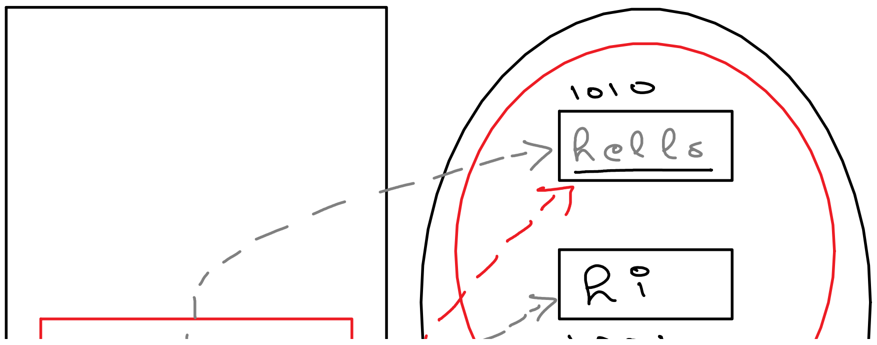
ex:1

```
class Demo1
{
   p s v m ( S[] args)
   {
        S.o.p ("hello");
                1010
   }
}
```

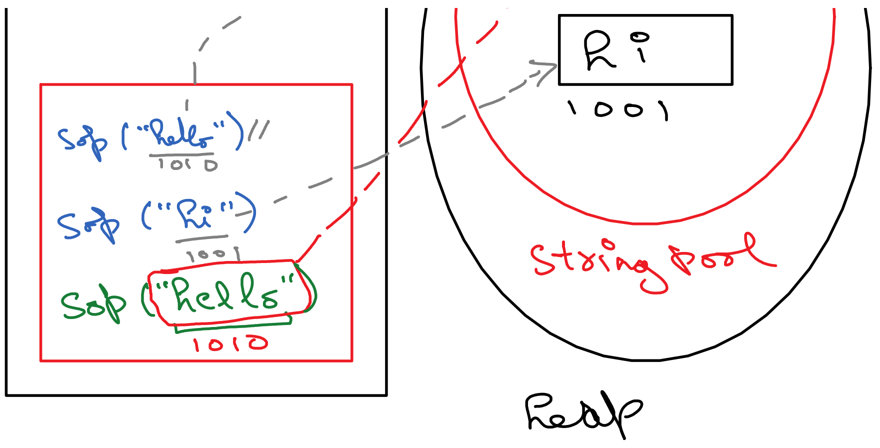


Ex2:

```
class Demo2
{
  p s v m (S[] a)
   {
```



```
 {
   s.o.p("hello");
   s.o.p(" hi ");
   s.o.p("hello");
 }
}
```

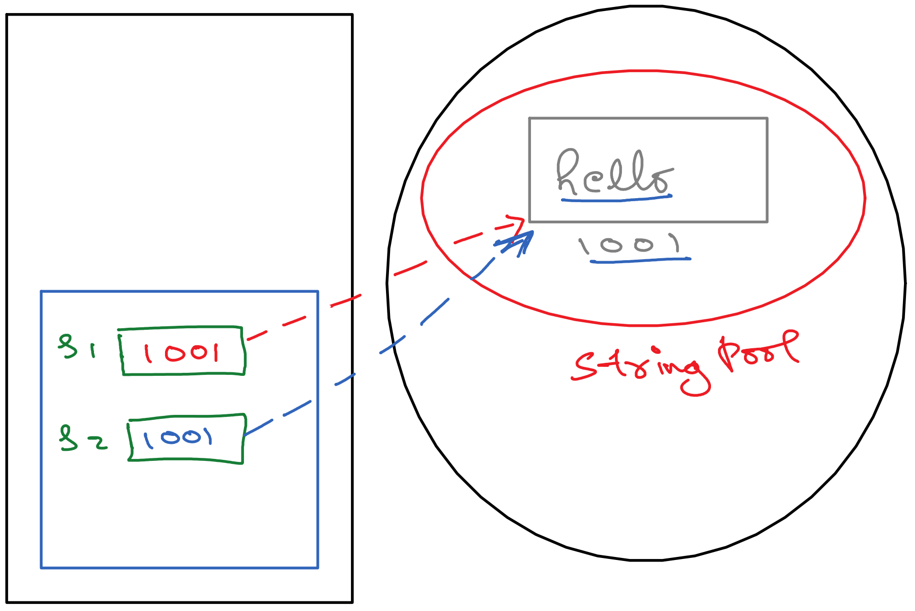


Note:

- ➢ An instance of String is created in String pool, only if String object does not exist, if the String object already exist, a new instance is not created instead the reference of already existing Object is given.
- ➢ refer the above example Demo3

exe:3

```
class Demo3
{
 p s v m(S[] a)
  {
    String s1, s2;
    s1 = "hello";
    s2 = "hello";
    Sop(s1 == s2);
  }
}
```



1. **java.lang.String:**

- ➢ it is a built-in class to store String literals
- ➢ it inherits java.lang.Object.
- ➢ String class is a **final class and public**
- ➢ String class implements :
  - □ Comparable
  - □ Serializable

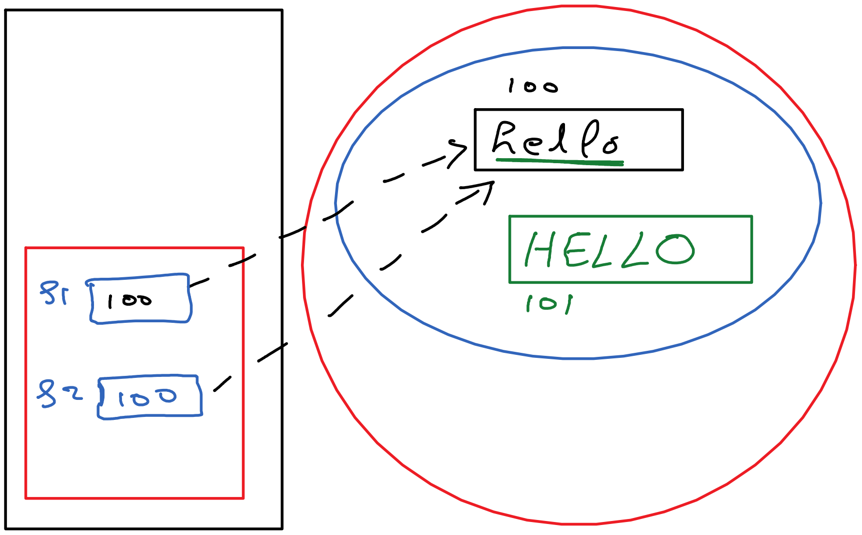
- ◻ CharSequence

➢ In String class the java.lang.Object class methods such as toString(), equals(Object) , hashCode() is overridden.

NOTE :

The instance of String class is immutable in nature. (Once the object is created we cannot modify it)

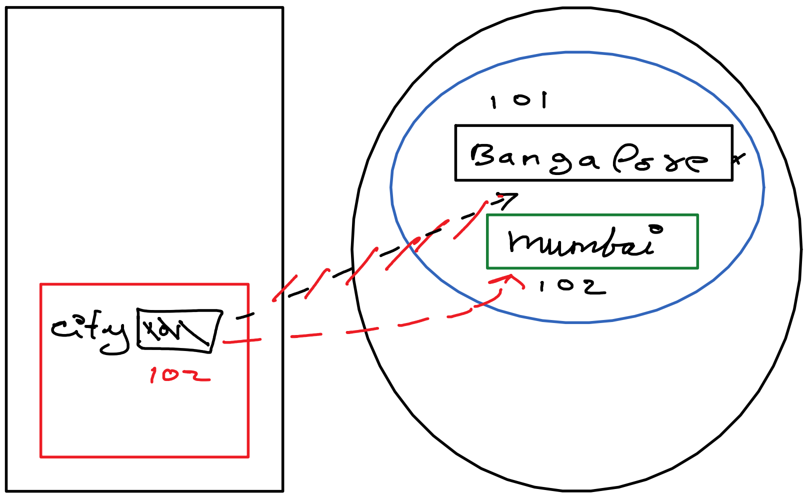
```
public static void main(String[] args) {

        String s1  = "hello" ;
        String s2  = s1 ;

        System.out.println(s1 ); // hello
        System.out.println( s2 ) ; // hello
        // hello to upper case

        s1.toUpperCase() ;    101

        System.out.println( s2 ); // hello
        System.out.println( s1 ); // hello

}
```



Note :

When we try to modify a String class object, a new object is created and the reference of new Object is given. refer the program Demo4.java and above illustration.

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

        // i want to store name of the city

        String city = "Bangalore" ;
        System.out.println( city );
        city = "mumbai" ;
        System.out.println(city);

}
```



We can create a String object using new operator and constructor

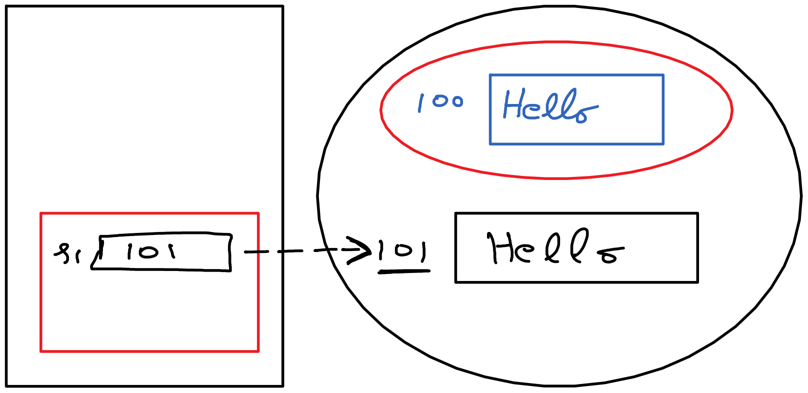
syntax:

new String("string literal")

1. when we create a String object using **new,** the String object will be created in the regular heap area and not in the String Pool.
2. Every time we use **new,** a new object is created.

```
public class Demo7 {

    public static void main(String[] args) {

        String s1 = new String("Hello") ;
        System.out.println(s1 );
    }
}
```

100



Pg2

Wednesday, May 6, 2020 8:13 AM

1. Constructors of java.lang.String :

modifier	signature	Description
public	String()	It creates an empty String object
public	String(String str)	It creates the String instance and initializes with with the given String
public	String(byte[] b)	The given byte[] will be converted into a String Object
public	String(char[] c)	The given character array will be converted into a String object
public	String(StringBuilder str)	Converts StringBuilder into String
public	String(StringBuffer str)	Converts StringBuffer into String

2. **Non-Static methods OF java.lang.String :**

Note: example programs loc : **workspace/Strings/src/pack2**

modifier	return type	signature	Description	example
public	String	toLowerCase()	to Convert String into lower case	S3.java
public	String	toUpperCase()	to convert the String into uppercase	try it yourself (S4.java)
public	String	concat(String s)	it will merge the current String with the given String and returns new String	S5.java
public	String	substring(int start)	it returns part of the string from the start index of the original String. 1. if start index is = length of the string we get empty string 2.	S7.java

			2. if start index > length of the string we get StringIndexOutOfBoundsException	
public	String	substring(int start_index, int end_index)	it returns part of String from start_index up to end_index-1 of the original String 1. if start_index and end_index both are same we get empty String Object	S8.java
public	int	length()	it will return the length of the String	try yourself
public	boolean	isEmpty()	it returns true if the length of the String is 0	try yourself
public	int	indexOf(char ch)	if the given character is present in the String it returns it's position, else it returns -1	try yourself
public	int	indexOf(char ch , int start_position)	it search for the character from the given position in the String.	try yourself
public	int	indexOf(String str)	it is used to search a string in a String, if present it returns it's position else it returns -1	try yourself
public	int	indexOf(String str , int start_pos)	it will search for the String from the given position	try yourself
public	boolean	contains(String str)	it will search for the given String in the original String, returns true if present else returns false	
public	String	trim()	it will remove spaces present before and after the String	S9.java
public	char	charAt(int index)	it returns the character at	S10.java

			the given index of the String 1. if the index is greater than or equal to length of the String or less than 0 we get StringIndexOutOfBoundsException	
public	char[]	toCharArray()	it is used to convert a String to character array	
public	String[]	split(String)	it is used to break one String into multiple Strings, and returns an array of String	
public	String	replace(char old , char new)	replaces all the occurrence of old character with the new character in the String	try yourself
public	byte[]	getBytes()	converts the String into an array of bytes	

Assignments :

1. WAJP to create a clone of a String object.
2. WAJP to convert a String to Uppercase without using built-in method
3. WAJP to count vowels present in a String entered by the user.
4. WAJP to count the number upper case letters present in the name of the user (input is user name)
5. WAJP to reverse a string.
6. WAJP to read a password from the user and check whether it is valid or not according to the given instructions.

   A password is valid only if all the below conditions are satisfied :

   - minimum length should be 8
   - the password must contain at least one upper case letter, special character and one digit.

- ○ the password should never start with special character.
- ○ it should not have space.

Note: check for all the cases (WBT)

<u>To Compare Two String Objects :</u>

SI no	**type**	**description**
1.	==	It compares reference of the String objects and not the values
2.	equals()	It compares Strings (values / state) it returns true only if both the strings are identical. (it is case sensitive) return type is boolean
3.	equalsIgnoreCase()	it will compare two Strings, their case will be ignored return type is boolean
4.	compareTo()	it is used to compare two Strings, and it returns an integer number. syntax : "string1" . compareTo("string2") case1 : it returns 0 if both are same case2 : it returns +ve integer if string1 is higher value than String2 case3 : it returns -ve integer if string1 is lesser value than string2. for example refer **S21.java** (this method is used to sort the Strings)

Pg3

Monday, May 11, 2020 8:15 AM

Note : (Disadvantage of java.lang.String class)

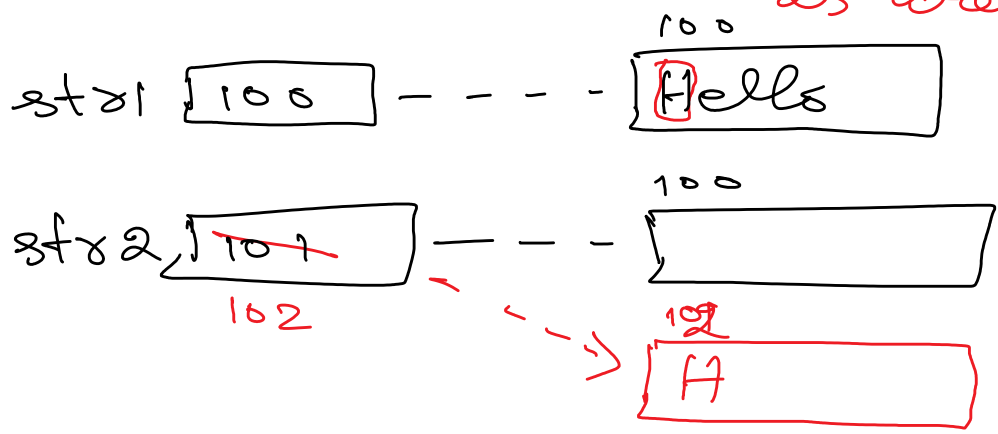
1. String class instance is immutable, hence every time we try to modify a new instance is created. Therefore, if a String requires many modifications in future it consumes loads of time cause a new String Object has to be created every time, it becomes a performance issue
   Example :

   Assume we have a String "Hello", this String has to be reversed using either any of the logic :

```
Str1 = "Hello";

str2 = ""; //empty string


for(int i=0; i<str1.length(); i++)
{
  str2= str1.charAt(i) + str2;
}
```

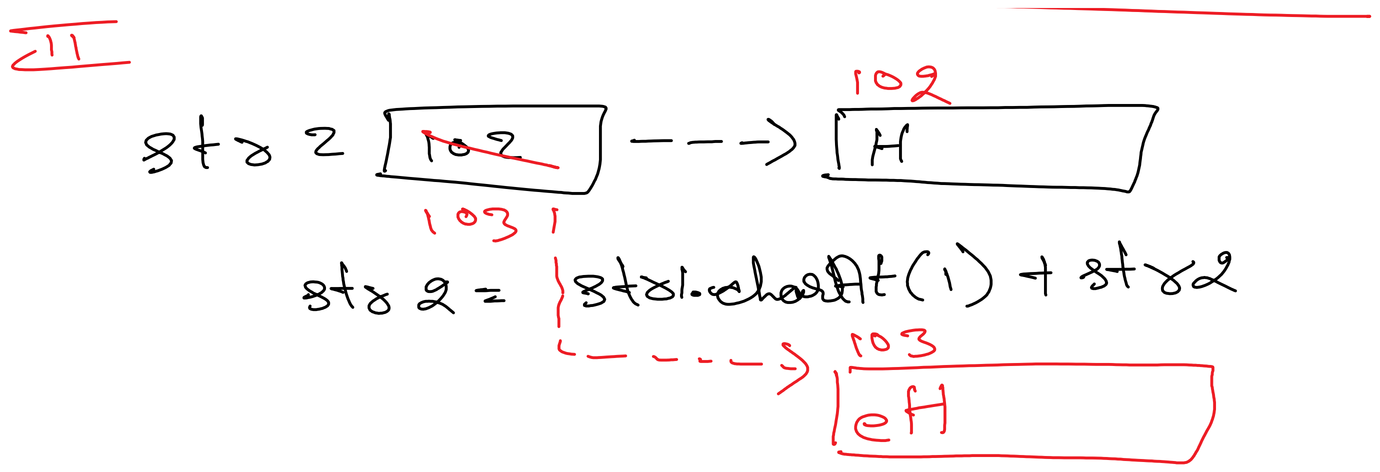
$\underbrace{\text{charAt(i) + str2}}$ → new string object is created.

I st



II

102



Hence as Observed in every iteration a new String Object is created, therefore if the length of the String is 5, minimum 5 times a String object is created, Hence it becomes a performance issue.

Note :

We can overcome this disadvantage by storing the String using instance of StringBuilder or StringBuffer.

StringBuilder and StringBuffer :

The instance of both StringBuilder and StringBuffer is mutable in nature, that is the object created can be modified.

Pg4

Monday, July 20, 2020 1:08 PM

<u>StringBuilder :</u>

Note :

1. StringBuilder is defined in java.lang package.
2. StringBuilder helps to create and store Strings in mutable object(Object which can be Modified)

Constructors Of StringBuilder :

1	StringBuilder()	it creates empty StringBuilder object
2	StringBuilder(String str)	it creates a StringBuilder Object and initializes with the given String

Note :

1. We cannot convert String to StringBuilder or StringBuilder to String implicitly or with the help of type cast operator
   refer : workspace/String/src/StringBuilder/s1.java

2. We cannot use + operator to concatenate two StringBuilder.
   refer : workspace/String/src/StringBuilder/s5.java

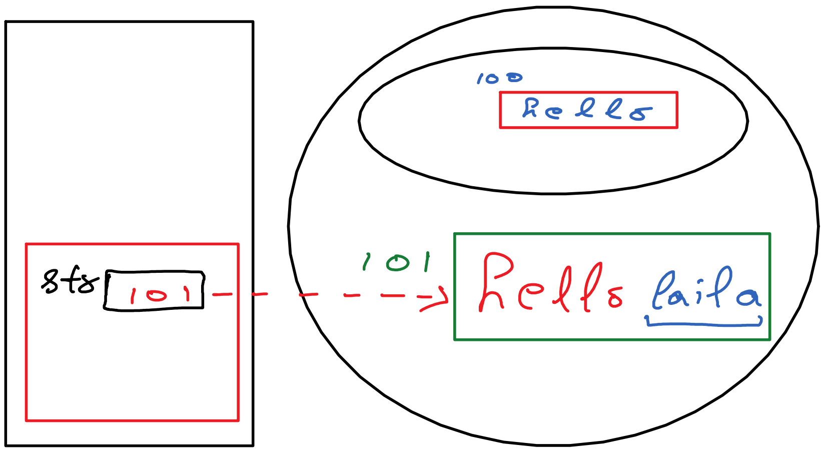
Methods Of StringBuilder :

Return Type	Signature	Description	Example
int	length()	provides length of StringBuilder	you can try
char	charAt(int index)	it gives the character present at given index	S6.java
StringBuilder	substring(int start_index)		S7.java
StringBuilder	substring(int start_index,		S7.java

	int end_index)		
StringBuilder	append	it is overloaded, such that it can accept different types of values, it will merge the given value to the StringBuilder at the end	S8.java

Program example to Prove StringBuilder is a Mutable Object:

```
public class S9 {

        public static void main(String[] args) {

                StringBuilder str  = new StringBuilder("Hello") ;
                str.append(" Laila") ;
                System.out.println( str );
        }
}
```



Task1 :
WAJP to accept a String from the user, store it as a StringBuilder and reverse the StringBuilder without using the built in method.

Assignment Questions :

1. WAJP to convert a StringBuffer into uppercase.
2. WAJP to convert a StringBuilder into lowercase.
3. WAJP to check whether two StringBuilder are palindrome or not.

Note :

1. What is the difference between StringBuilder and StringBuffer
2. To reverse a StringBuilder / StringBuffer using recursion.

NOTE :

In StringBuilder/StringBuffer, the equals() and hashCode() method of java.lang.Object class is not overridden, Therefore it compares the reference and not the state.

Then How to compare 2 StringBuilders or 2 StringBuffers ?

Solution 1 : Convert them to String and compare.

Solution 2 : compare character by character using charAt and loop