@techwithvishalraj

Java

String
v/s
StringBuilder
v/s
StringBuffer

Follow

Share

String

- String class object represents sequence of char values.
- It is an immutable class.
- it is a final class.
- The java.lang.String class implements Serializable, Comparable and CharSequence Interfaces.

Difference between '==' operator and equals() method.

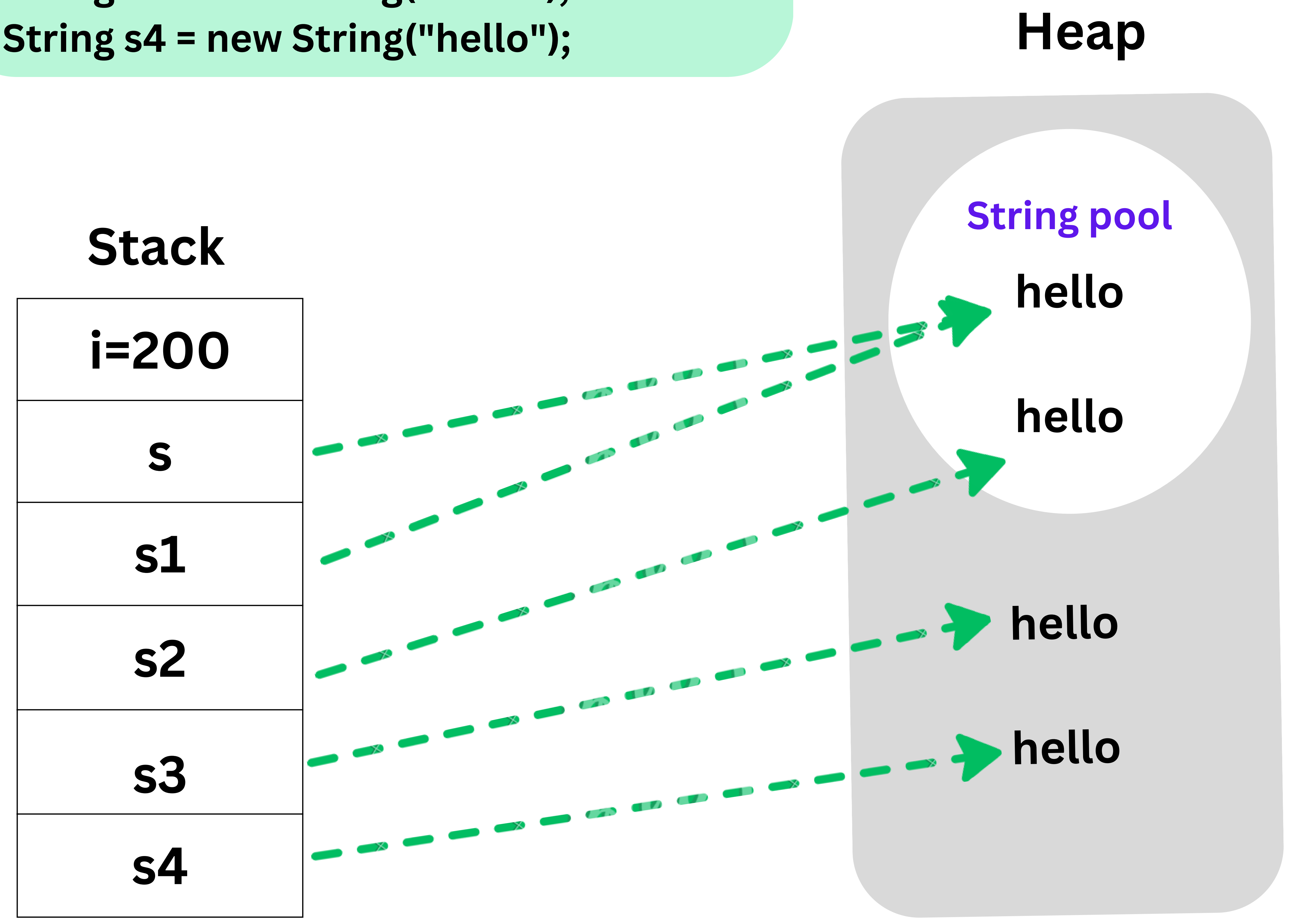
```
String s1 = "hello";
String s2 = "hello";
String s3 = new String("hello");
String s4 = new String("hello");
String s5 = new String("hello").intern();

s1 == s2 -------> true
s3 == s4 -------> false
s1 == s3 -------> false
s1.equals(s2) -------> true
s3.equals(s4) -------> true
s1.equals(s3) -------> true
s1.equals(s5) -------> false
```

String

@techwithvishalraj

- Strings are stored in the heap memory and its reference is stored in the stack memory.

```
int i = 200;
String s = "hello";
String s1 = "hello";
String s2 = new String("hello").intern();
String s3 = new String("hello");
String s4 = new String("hello");
```



@techwithvishalraj

StringBuffer

- StringBuffer class is used to create mutable (modifiable) objects.
- StringBuffer is synchronized i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously.
- All methods of StringBuffer are synchronized.
- StringBuffer is less efficient than StringBuilder.
- StringBuffer was introduced in Java 1.0

Let's see the code to check the performance of StringBuilder class.

```
public class Main{
   public static void main(String[] args){
      long startTime = System.currentTimeMillis();
      StringBuffer sb = new StringBuffer("Hello");
      for (int i=0; i<100000; i++){
         sb.append("World");
      }
      System.out.println("Time taken by StringBuffer: "
            + (System.currentTimeMillis() - startTime) + "ms");
   }
}
```

```
Time taken by StringBuffer:   17 ms
```

StringBuilder

- StringBuffer class is used to create mutable (modifiable) objects.
- StringBuilder is non-synchronized i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously.
- StringBuilder is more efficient than StringBuffer.
- Alternatively, you can manually synchronize access to a StringBuilder object using external synchronization mechanisms such as synchronized blocks.
- StringBuilder was introduced in Java 1.5.

Let's see the code to check the performance of StringBuilder class.

```
public class Main{
   public static void main(String[] args){
      long startTime = System.currentTimeMillis();
      StringBuilder sb = new StringBuilder("Hello");
      for (int i=0; i<100000; i++){
         sb2.append("World");
      }
      System.out.println("Time taken by StringBuilder: "
             + (System.currentTimeMillis() - startTime) + "ms");
   }
}
```

```
Time taken by StringBuilder:   6 ms
```

@techwithvishalraj

String v/s StringBuffer v/s StringBuilder

- StringBuffer and StringBuilder creates an empty object(sb) with the initial capacity of 16.
- If the number of character increases from its current capacity, it increases the capacity by (oldcapacity*2)+2. For example if your current capacity is 16, it will be (16*2)+2=34.

```
StringBuffer sb = new StringBuffer();
System.out.println(sb.capacity());          ------> 16

StringBuilder sb = new StringBuilder();
System.out.println(sb.capacity());          ------> 16
```

- StringBuffer and StringBuilder creates an empty object(sb) with the specified capacity as length.

```
StringBuffer sb = new StringBuffer (12);
System.out.println(sb.capacity());          ------> 12

StringBuilder sb = new StringBuilder(12);
System.out.println(sb.capacity());          ------> 12
```

- String can't afford this facility. initial length/size of string is 0.

```
String s = new String();
System.out.println(s.length());             ------> 0
```

@techwithvishalraj

String v/s StringBuffer v/s StringBuilder

- **String class overrides the equals() method of Object class. So you can compare the contents of two strings by equals() method.**

```
String s= new String("Hello");
System.out.println(s.equals("Hello"));          -----> true
```

- **StringBuffer and StringBuilder class doesn't override the equals() method of Object class.**

```
StringBuffer sb = new StringBuffer("Hello");
System.out.println(sb.equals("Hello"));         -----> false
```

```
StringBuilder sb = new StringBuilder("Hello");
System.out.println(sb.equals("Hello"));         -----> false
```

Methods in String, StringBuffer & StringBuilder

@techwithvishalraj

String	StringBuffer and StringBuilder
charAt(int index)	charAt(int index)
length()	length()
substring(int start) substring(int start, int end)	substring(int start) substring(int start, int end)

	insert(int offset, String str)
	append(String str)
	deleteCharAt(int index)
	delete(int start, int end)

indexOf(String str), lastIndexOf(String str)	indexOf(String str), lastIndexOf(String str)

startsWith(String prefix)
endsWith(String suffix)	

replace(char old, char new)	replace(int start, int end, String str)
isBlank(), isEmpty()	isEmpty()
matches("[A-Za-z]*") , contains("asd")	
	reverse()
equals(Object anObject)	
toLowerCase(), toUpperCase()	
toString()	

trim(), strip(), stripLeading()
stripTrailing()	

Thank you!

vishal-bramhankar

techwithvishalraj

Vishall0317