String Builders

String vs StringBuilder

- String bir immutable class, yani bir String objesi oluşturduktan sonra değiştiremeyiz.
- Immutable bir objeyi değiştirmek istersek, Java o objeyi klonlar ve yapilan degisiklikleri klonlanmis yeni obje uzerinde gerceklestirir.
- StringBuilder ise bir mutable class, yani bir StringBuilder objesi oluşturduğumuzda üzerinde değişiklik yapabiliyoruz.
- Böylece hafızada her seferinde yeni bir alan açılmadan var olan alan üzerinde değişiklik yapılabilir. Bu da hafıza kullanımı olarak String sınıfının önüne geçirir.

```
String str= "Mehmet";
str.toUpperCase();
System.out.println(str); // Mehmet
```

```
StringBuilder sb= new StringBuilder("Mehmet");
sb.setCharAt( index: 5, ch: 'd');
System.out.println(sb); //Mehmed
```

If you use String concatenation in a loop, something like this,

```
String s = "";
for (int i = 0; i < 100; i++) {
    s += ", " + i;
}
```

then you should use a `StringBuilder` (not `StringBuffer`) instead of a `String`, because it is much faster and consumes less memory.

String Builder

1) StringBuilder sb1 = new StringBuilder() ===> Bos bir StringBuilder olusturur

2) StringBuilder sb2 = new StringBuilder("animal"); ===> Belli bir degeri olan StringBuilder olusturur

3) StringBuilder sb3 = new StringBuilder(5); ===> Ilk uzunlugu tahmin edilen bir StringBuilder olusturur.

```
StringBuilder sb = new StringBuilder(5);
```

0	1	2	3	4

```
sb.append("anim");
```

a	n	i	m	
0	1	2	3	4

Capacity 12=5*2+2

```
sb.append("als");
```

a	n	i	m	a	l	s		
0	1	2	3	4	5	6	7	...

String vs StringBuilder:

2. Constructor

```
String str="abc";  //Java Virtual Machine öncelikle o değişkeni String
// havuzunda arar bulursa, aynı referansı verir yeni String'e.

String str2= new String( original: "abc");
//yeni bir Object olusturur ve  sonra istenen degeri assign eder.
```

```
// boş SB
StringBuilder sb1= new StringBuilder();

// belli değeri olan SB
StringBuilder sb2= new StringBuilder("abc");

// capacity si 5 olan SB
StringBuilder sb3= new StringBuilder( capacity: 5);
```

String vs StringBuilder:

3. Equality

```
String s1 = new String( original: "abc");
String s2 = new String( original: "abc");
System.out.println(s1.equals(s2));  // true

StringBuilder sb1 = new StringBuilder("abc");
StringBuilder sb2 = new StringBuilder("abc");
System.out.println(sb1.equals(sb2));  // false
```

- StringBuilder does not override Object's .equals() function, which means the two object references are not the same and the result is false.

- For StringBuilder, we can use

```
System.out.println(sb1.toString().equals(sb2.toString())); //true
```

String vs StringBuilder:

3. Equality

```
String s1 = new String( original: "abc");
String s2 = new String( original: "abc");
System.out.println(s1.equals(s2));  // true

StringBuilder sb1 = new StringBuilder("abc");
StringBuilder sb2 = new StringBuilder("abc"
System.out.println(sb1.equals(sb2));  // fa
```

- StringBuilder does not override Object's .equals() function, which means the two object references are not the same and the result is false.

4

The `StringBuilder` class does not provide an overriden `equals()` method. As such, when that method is called on an instance of `StringBuilder`, the `Object` class implementation of the method is executed, since `StringBuilder extends Object`.

The source code for that is

```
public boolean equals(Object obj) {
    return (this == obj);
}
```

Which simply compares reference equality.

Share Follow

answered Sep 2, 2013 at 4:39

String vs StringBuilder:

4. Length

- Since String is immutable, its length is fixed. But StringBuilder has the setLength() method, which can change the StringBuilder object to the specified length.

```
String s3="Java ne güzel ne güzel";
System.out.println(s3.length());  //22

StringBuilder sb4 = new StringBuilder("Java ne güzel ne güzel");
System.out.println(sb4.length());  //22

sb4.setLength(7);  // Java ne
System.out.println(sb4);
```

String vs StringBuilder:

5. Performance

- StringBuilder hızlıdır ve concatenation gerçekleştirirken bir string den daha az bellek tüketir. Bunun nedeni, string in Java'da immutable olmasıdır.

Difference between String, StringBuffer and StringBuilder?

- StringBuffer and StringBuilder are mutable classes. StringBuffer operations are thread-safe and synchronized where StringBuilder operations are not thread-safe. So in a multi-threaded environment, we should use StringBuffer but in the single-threaded environment, we should use StringBuilder.
- StringBuilder performance is fast than StringBuffer because of no overhead of synchronization.

- StringBuilder thread-safe değildir. Yani synchronized değildir. Thread'li bir işlem kullanılacaksa StringBuilder kullanılması güvenli değildir.
- ➢ Not: StringBuffer, StringBuilder'a benzer. StringBuilder, StringBuffer'dan hizlidir. Multi-thread icin StringBuffer kullanilir.

Sorular

What is the result of the following code? (Choose all that apply)

```
13: String a = "";
14: a += 2;
15: a += 'c';
16: a += false;
17: if ( a == "2cfalse") System.out.println("==");
18: if ( a.equals("2cfalse")) System.out.println("equals");
```

A. Compile error on line 14.

B. Compile error on line 15.

C. Compile error on line 16.

D. Compile error on another line.

E. ==

F. equals

G. An exception is thrown.

İstediğimiz herseyi append edebiliriz

We know that concat method or the concatenation operator +, can be used to concatenate two strings of String class. Concatenation of strings of StringBuffer class is done using the append method of StringBuffer class. This method is overloaded to take all types of arguments. append method exists in the following forms:
public StringBuffer append(Object obj)
public StringBuffer append(String str)
public StringBuffer append(StringBuffer sb)
public StringBuffer append(CharSequence s)
public StringBuffer append(CharSequence s, int start, int end)
public StringBuffer append(char[] str)
public StringBuffer append(char[] str, int offset, int len)
public StringBuffer append(boolean b)
public StringBuffer append(char c)
public StringBuffer append(int i)
public StringBuffer append(long lng)
public StringBuffer append(float f)
public StringBuffer append(double d)

String Builder

4) charAt(); StringBuilder'da istenen index'deki karakteri verir

5) delete(4,7); StringBuilder'da istenen index'ler arasindaki karakterleri siler.

6) deleteCharAt(7); StringBuilder'da istenen index'deki tek karakteri siler

7) equals(); Iki StringBuilder'in degerlerinin karsilastirir.

NOT 1: equals() method'unda parantez icine String yazarsak hata vermez ama false doner.

NOT 2: equals() method'u == gibi calisir

String Builder

8) indexOf(); StringBuilder'da istenen karakterin index'ini verir.

9) insert(3, "Java "); StringBuilder'da istenen indexden baslayarak istenen karakteri ekler.

10) insert(3, "Java ",1,2); StringBuilder'da istenen indexden baslayarak verilen String'in istenen parcasini ekler.

11) replace(3, 8, " Ali ");

StringBuilder'da istenen index'ler arasindaki bolumun yerine verilen String'i ekler.

String Builder

12) reverse( ); StringBuilder'i tersine cevirir.

13) setCharAt(3, 'k'); StringBuilder'da istenen index'deki karakteri istedigimiz karakter yapar.

14) subSequence(3,7); StringBuilder'da istenen indexler arasindaki karakterleri dondurur.

15) toString( ); StringBuilder'i String'e cevirir.

toString()'den sonra nokta koyup String method'lari kullanilabilir.

String Builder

16) trimToSize(); StringBuilder'in capacity ile length'ini esitler.

17) compareTo(); 2 StringBuilder'in tum karakterlerinin esitligini kontrol eder. (0 ise esit)

Soru : For loop ile 1000 defa bir islem yapalim. Oncesinde ve sonrasinda zamani kontrol edip StringBuilder ve String class'larinin performanslarini karsilastiralim.

Ipucu: long TimeSb = System.nanoTime(); kullanalim

Find The First Non Repeated Character In A String

- https://javahungry.blogspot.com/2013/12/first-non-repeated-character-in-string-java-program-code-example.html

position of the last occurrence of a given character in a string.

Logic

If positions returned by the indexOf() and lastIndexOf() methods of the specified character are the same, then that character is the first non-repeated character in a string.

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

        String inputStr ="teeter";

        for(char i :inputStr.toCharArray()){
        if ( inputStr.indexOf(i) == inputStr.lastIndexOf(i)) {
            System.out.println("First non-repeating character is: "+i);
            break;
        }
        }
    }
}
```

Referanslar

- https://www.techiedelight.com/difference-between-string-stringbuilder-java/#:~:text=A%20String%20is%20immutable%20in,created%20in%20the%20string%20pool.
- https://www.journaldev.com/1321/java-string-interview-questions-and-answers

- https://stackoverflow.com/questions/18565701/stringbuilder-equals-java
- https://www.geeksforgeeks.org/different-ways-to-print-first-k-characters-of-the-string-in-java/?ref=lbp