Method	Decription
int lastIndexOf(Char ch);	returns index of last occurrence of the specified character if the specified character is not available then return -1.

String Buffer:

- If a user wants to change the content frequently then it is recommended to go for StringBuffer.
- StringBuffer objects are mutable, so they can be modified.
- We can create a StringBuffer object by using new operator and pass the string to the object, as:

  StringBuffer sb = new StringBuffer ("Sujatha");
- The default initial capacity of a StringBuffer is"16".
- After reaching its maximum limit it will be increased to (currentcapacity+1)*2. ie; (16+1)*2.

StringBuffer Methods

Method	Description
int length()	Return the no of characters present in the StringBuffer
int capacity()	Returns how many characters a StringBuffer can hold
char charAt(int index)	Returns the character located at specified index.
void setCharAt(int index, char ch)	Replaces the character locating at specified index with the provided character.
delete(int begin,int end)	Deletes characters from begin index to end n-1 index.
deleteCharAt(int index)	Deletes the character locating at specified index
reverse()	Reverses the given StringBuffer

Method	Description
void setLength(int length)	Consider only specified no of characters and remove all the remaining characters
void ensureCapacity(int initialcapacity);	To increase the capacity dynamically based on our requirement.

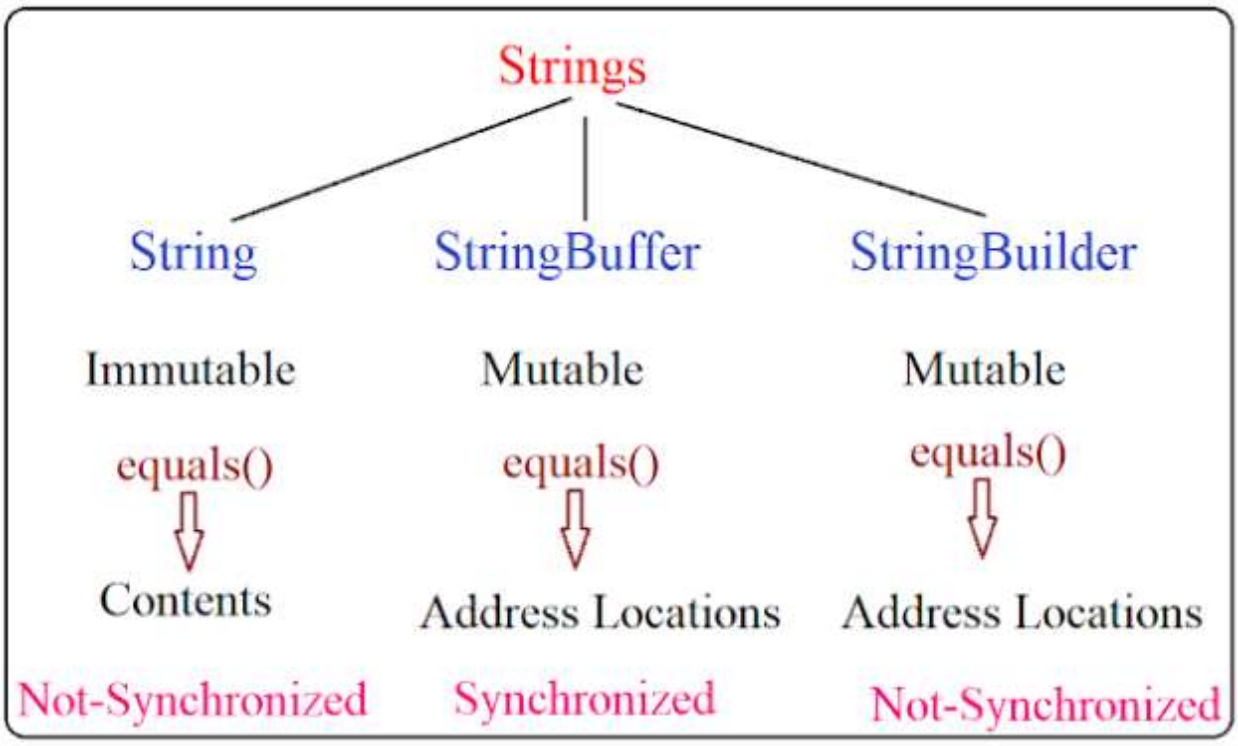
StringBuilder

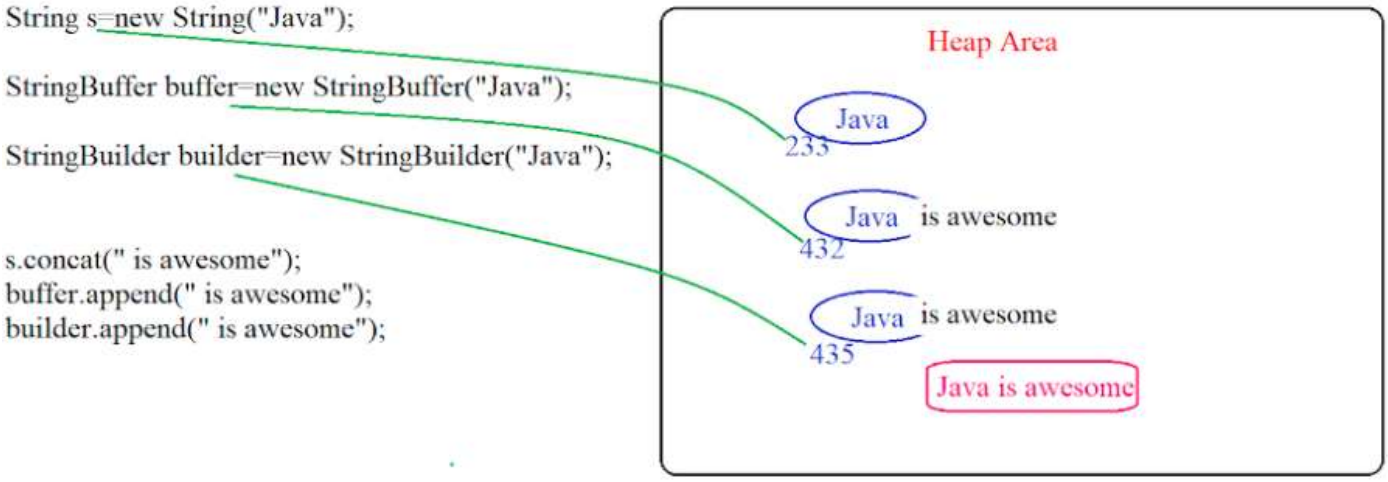
- String Builder will be having same methods as of StringBuffer except the following differences.

StringBuffer	StringBuilder
Every method present in StringBuffer is synchronized.	No method present in StringBuilder is synchronized.
At a time only one thread is allowed to operate on the StringBuffer object hence StringBuffer object is Thread safe.	At a time Multiple Threads are allowed to operate simultaneously on the StringBuilder object hence StringBuilder is not Thread safe.
It increases waiting time of the Thread and hence relatively performance is low.	Threads are not required to wait and hence relatively performance is high.
Introduced in 1.0 version.	Introduced in 1.5 versions.

Differences b/w String, StringBuffer & String Builder

- String is immutable while StringBuffer and StringBuilder is mutable object.
- StringBuffer is synchronized while StringBuilder is not which makes StringBuilder faster than StringBuffer.
- Use String if you require immutability use Stringbuffer in java if you need mutable + thread-safety and use StringBuilder in Java if you require mutable + without thread-safety.





Synchronized means getting results late but consistency in result. It takes more time

Not Synchronized means getting results fast but inconsistency in result. It takes less time.

```
package com.pack1;

public class ClassA
{
    void meth1()
    {
        System.out.println("meth1() called\n");

        String s=new String("Java");
        StringBuffer buffer=new StringBuffer("Java");
        StringBuilder builder=new StringBuilder("Java");

        System.out.println("String : "+s);
        System.out.println("StringBuffer : "+buffer);
        System.out.println("StringBuilder : "+builder);


    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
}
```

```
meth1() called

String : Java
StringBuffer : Java
StringBuilder : Java
```

Scp is only for String class

Equals --- string class---difference in content

Equals—String Buffer—difference in address location

Equals--String Builder— difference in address location

Double equals (==) ---String class—difference in address location

Double equals (==) ---buffer class—difference in address location

Double equals (==) ---builder class—difference in address location

(Buffer1.toString().equals(buffer2.toSting()))----buffer & builder---difference in content.

```
3  public class ClassA
4  {
5      void meth1()
6      {
7          System.out.println("meth1() called\n");
8
9          String s=new String("Java");
10         StringBuffer buffer=new StringBuffer("Java");
11         StringBuilder builder=new StringBuilder("Java");
12
13         System.out.println("------Before-------");
14         System.out.println("String : "+s);
15         System.out.println("StringBuffer : "+buffer);
16         System.out.println("StringBuilder : "+builder);
17
18         s.concat(" is awesome");
19         buffer.append(" is awesome");
20         builder.append(" is awesome");
21
22         System.out.println("\n------After-------");
23         System.out.println("String : "+s);
24         System.out.println("StringBuffer : "+buffer);
25         System.out.println("StringBuilder : "+builder);
26     }
```

```
27    void meth2()
28    {
29        System.out.println("meth2() called\n");
30
31        String s1=new String("Java");
32        StringBuffer buffer1=new StringBuffer("Java");
33        StringBuilder builder1=new StringBuilder("Java");
34
35        String s2=new String("Java");
36        StringBuffer buffer2=new StringBuffer("Java");
37        StringBuilder builder2=new StringBuilder("Java");
38
39        System.out.println(s1.equals(s2));// t
40        System.out.println(buffer1.equals(buffer2));// f
41        System.out.println(builder1.equals(builder2)+"\n");//f
42
43        System.out.println(s1==s2);
44        System.out.println(buffer1==buffer2);
45        System.out.println(builder1==builder2);
46
47        System.out.println("\n"+buffer1.toString().equals(buffer2.toString()));
48        System.out.println(builder1.toString().equals(builder2.toString()));
49
```

```
50        buffer2.append(" is awesome");
51        builder2.append(" is awesome");
52
53        System.out.println("\n"+buffer1.toString().equals(buffer2.toString()));
54        System.out.println(builder1.toString().equals(builder2.toString()));
55    }
56    void meth3()
57    {
58        System.out.println("meth3() called\n");
59        StringBuilder sb=new StringBuilder();
60        System.out.println("capacity() : "+sb.capacity());
61        System.out.println("length() : "+sb.length());
62
63        sb.append("abcdefghijklmnop");
64        System.out.println(sb);
65        sb.append("qr");
66        System.out.println(sb);
67        System.out.println("capacity() : "+sb.capacity());//(currentcapacity+1)*2
68        System.out.println("length() : "+sb.length());
69
```

```
70          System.out.println(sb.reverse());
71          System.out.println(sb.charAt(0));
72      }
73      public static void main(String[] args)
74      {
75          ClassA aobj=new ClassA();
76          //aobj.meth1();
77          //aobj.meth2();
78          aobj.meth3();
79      }
80  }
```

```
 3  public class ClassB
 4  {
 5      void meth1()
 6      {
 7          int iterations = 100000;
 8          String data = "a";
 9
10          long startTime = System.currentTimeMillis();
11          String str = "";
12          for (int i = 0; i < iterations; i++)
13          {
14              str +=data;
15          }
16          long endTime = System.currentTimeMillis();
17          long stringDuration = endTime - startTime;
18
19          startTime = System.currentTimeMillis();
20          StringBuffer stringBuffer = new StringBuffer();
21          for (int i = 0; i < iterations; i++)
22          {
23              stringBuffer.append(data);
24          }
25          endTime = System.currentTimeMillis();
26          long stringBufferDuration = endTime - startTime;
```

```
27
28          startTime = System.currentTimeMillis();
29          StringBuilder stringBuilder = new StringBuilder();
30          for (int i = 0; i < iterations; i++)
31          {
32              stringBuilder.append(data);
33          }
34          endTime = System.currentTimeMillis();
35          long stringBuilderDuration = endTime - startTime;
36
37          System.out.println("Time taken for String concatenation: " + stringDuration + " ms");
38          System.out.println("Time taken for StringBuffer concatenation: " + stringBufferDuration + " ms");
39          System.out.println("Time taken for StringBuilder concatenation: " + stringBuilderDuration + " ms");
40      }
41      public static void main(String[] args)
42      {
43          new ClassB().meth1();
44      }
45  }
```

```
Time taken for String concatenation: 2223 ms
Time taken for StringBuffer concatenation: 12 ms
Time taken for StringBuilder concatenation: 10 ms
```