StingBuilder & StingBuffer Classes Notes

In general, StringBuilder & StringBuffer classes can be used whenever a String is used.

StringBuilder & StringBuffer are more flexible than String because they can be modified, meaning these objects can be mutated where the String object is immutable.

* Can add
* Can insert
* Can append
* Can delete characters
* Can replace characters

StringBuffer is synchronized (only one task is allowed to execute methods) = used in concurrent programming – multiple tasks running concurrently = multithreading & parallel

When to use what – a String, a StringBuilder, or a StringBuffer

* String – when the object is not to be changed \*\* it is more efficient (in memory storage)
  + If the string is modified over & over then unused objects are left in memory waiting for the garbage collector
* StringBuilder – when a single program is using the string
  + Are not thread safe
* StringBuffer – when there is multithreading / parallel access to the string

the methods (except setCharAt) do 2 things

1. Changes the content of the string builder/buffer
2. Returns the reference of the string builder/buffer

The principal operations on a StringBuilder that are *not* available in String are the append() and insert() methods, which are overloaded so as to accept data of any type. Each converts its argument to a string and then appends or inserts the characters of that string to the character sequence in the string builder. The append method *always* adds these characters *at the end* of the existing character sequence, while the insert method adds the characters *at a specified point*/index spot.

\*\*\* Internally a string builder is an array of characters

The difference of length & capacity

* Length = the actual size of the string stored in the builder
* Capacity = the current size of the builder
  + The builder's capacity is automatically increased if more characters are added to exceed its capacity
  + The size of the array
    - If the size of the array *– the capacity -* is exceeded, the array is replaced with a new array
      * New array = 2 \* (the previous array size + 1)

|  |  |
| --- | --- |
| **StringBuilder Constructors** | |
| **Constructor** | **Description** |
| StringBuilder() | Creates an empty string builder with a capacity of 16 (16 empty elements). |
| StringBuilder(CharSequence cs) | Constructs a string builder containing the same characters as the specified CharSequence, plus an extra 16 empty elements trailing the CharSequence. |
| StringBuilder(int initCapacity) | Creates an empty string builder with the specified initial capacity. |
| StringBuilder(String s) | Creates a string builder whose value is initialized by the specified string, plus an extra 16 empty elements trailing the string. |

For example, the following code

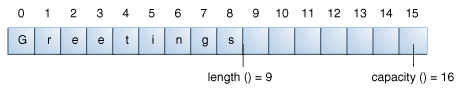
// creates empty builder, capacity 16

StringBuilder sb = new StringBuilder();

// adds 9 character string at beginning

sb.append("Greetings");

will produce a string builder with a length of 9 and a capacity of 16:



The StringBuilder class has some methods related to length and capacity that the String class does not have:

|  |  |
| --- | --- |
| **Length and Capacity Methods** | |
| **Method** | **Description** |
| void setLength(int newLength) | Sets the length of the character sequence. If newLength is less than length(), the last characters in the character sequence are truncated. If newLength is greater than length(), null characters are added at the end of the character sequence. |
| void ensureCapacity(int minCapacity) | Ensures that the capacity is at least equal to the specified minimum. |

A number of operations (for example, append(), insert(), or setLength()) can increase the length of the character sequence in the string builder so that the resultant length() would be greater than the current capacity(). When this happens, the capacity is automatically increased.

**StringBuilder Operations**

The principal operations on a StringBuilder that are not available in String are the append() and insert() methods, which are overloaded so as to accept data of any type. Each converts its argument to a string and then appends or inserts the characters of that string to the character sequence in the string builder. The append method always adds these characters at the end of the existing character sequence, while the insert method adds the characters at a specified point.

Here are a number of the methods of the StringBuilder class.

|  |  |
| --- | --- |
| **Various StringBuilder Methods** | |
| **Method** | **Description** |
| StringBuilder append(boolean b) StringBuilder append(char c) StringBuilder append(char[] str) StringBuilder append(char[] str, int offset, int len) StringBuilder append(double d) StringBuilder append(float f) StringBuilder append(int i) StringBuilder append(long lng) StringBuilder append(Object obj) StringBuilder append(String s) | Appends the argument to this string builder. The data is converted to a string before the append operation takes place. |
| StringBuilder delete(int start, int end) StringBuilder deleteCharAt(int index) | The first method deletes the subsequence from start to end-1 (inclusive) in the StringBuilder's char sequence. The second method deletes the character located at index. |
| StringBuilder insert(int offset, boolean b) StringBuilder insert(int offset, char c) StringBuilder insert(int offset, char[] str) StringBuilder insert(int index, char[] str, int offset, int len) StringBuilder insert(int offset, double d) StringBuilder insert(int offset, float f) StringBuilder insert(int offset, int i) StringBuilder insert(int offset, long lng) StringBuilder insert(int offset, Object obj) StringBuilder insert(int offset, String s) | Inserts the second argument into the string builder. The first integer argument indicates the index before which the data is to be inserted. The data is converted to a string before the insert operation takes place. |
| StringBuilder replace(int start, int end, String s) void setCharAt(int index, char c) | Replaces the specified character(s) in this string builder. |
| StringBuilder reverse() | Reverses the sequence of characters in this string builder. |
| String toString() | Returns a string that contains the character sequence in the builder. |

**Note:** You can use any String method on a StringBuilder object by first converting the string builder to a string with the toString() method of the StringBuilder class. Then convert the string back into a string builder using the StringBuilder(String str) constructor.