StringBuilder Class in Java with Examples

Difficulty Level : Easy Last Updated : 18 Nov, 2021

The **StringBuilder** in Java represents a mutable sequence of characters. Since the String Class in Java creates an immutable sequence of characters, the StringBuilder class provides an alternative to String Class, as it creates a mutable sequence of characters. The function of StringBuilder is very much similar to the StringBuffer class, as both of them provide an alternative to String Class by making a mutable sequence of characters. However the StringBuilder class differs from the StringBuffer class on the basis of synchronization. The StringBuilder class provides no guarantee of synchronization whereas the StringBuffer class does. Therefore this class is designed for use as a drop-in replacement for StringBuffer in places where the StringBuffer was being used by a single thread (as is generally the case). Where possible, it is recommended that this class be used in preference to StringBuffer as it will be faster under most implementations. Instances of StringBuilder are not safe for use by multiple threads. If such synchronization is required then it is recommended that StringBuffer be used.

Class Hierarchy:

```
java.lang.Object
 ↳ java.lang
    ↳ Class StringBuilder
```

Syntax:

```
public final class StringBuilder
    extends Object
    implements Serializable, CharSequence
```

Constructors in Java StringBuilder:

- **StringBuilder():** Constructs a string builder with no characters in it and an initial capacity of 16 characters.
- **StringBuilder(int capacity):** Constructs a string builder with no characters in it and an initial capacity specified by the capacity argument.

- **StringBuilder(CharSequence seq):** Constructs a string builder that contains the same characters as the specified CharSequence.

- **StringBuilder(String str):** Constructs a string builder initialized to the contents of the specified string.

Below is a sample program to illustrate StringBuilder in Java:

```
// Java code to illustrate StringBuilder

import java.util.*;
import java.util.concurrent.LinkedBlockingQueue;

public class GFG1 {
    public static void main(String[] argv)
        throws Exception
    {

        // create a StringBuilder object
        // using StringBuilder() constructor
        StringBuilder str
            = new StringBuilder();

        str.append("GFG");

        // print string
        System.out.println("String = "
                           + str.toString());

        // create a StringBuilder object
        // using StringBuilder(CharSequence) constructor
        StringBuilder str1
            = new StringBuilder("AAAABBBCCCC");

        // print string
        System.out.println("String1 = "
                           + str1.toString());

        // create a StringBuilder object
        // using StringBuilder(capacity) constructor
        StringBuilder str2
            = new StringBuilder(10);
```

```
        // print string
        System.out.println("String2 capacity = "
                           + str2.capacity());

        // create a StringBuilder object
        // using StringBuilder(String) constructor
        StringBuilder str3
            = new StringBuilder(str1.toString());

        // print string
        System.out.println("String3 = "
                           + str3.toString());
    }
}
```

Output:

```
String = GFG
String1 = AAAABBBCCCC
String2 capacity = 10
String3 = AAAABBBCCCC
```

Methods in Java StringBuilder:

1. **StringBuilder append(X x)**: This method appends the string representation of the X type argument to the sequence.
2. **StringBuilder appendCodePoint(int codePoint)**: This method appends the string representation of the codePoint argument to this sequence.
3. **int capacity()**: This method returns the current capacity.
4. **char charAt(int index)**: This method returns the char value in this sequence at the specified index.
5. **IntStream chars()**: This method returns a stream of int zero-extending the char values from this sequence.

6. **int codePointAt(int index)**: This method returns the character (Unicode code point) at the specified index.

7. **int codePointBefore(int index)**: This method returns the character (Unicode code point) before the specified index.

8. **int codePointCount(int beginIndex, int endIndex)**: This method returns the number of Unicode code points in the specified text range of this sequence.

9. **IntStream codePoints()**: This method returns a stream of code point values from this sequence.

10. **StringBuilder delete(int start, int end)**: This method removes the characters in a substring of this sequence.

11. **StringBuilder deleteCharAt(int index)**: This method removes the char at the specified position in this sequence.

12. **void ensureCapacity(int minimumCapacity)**: This method ensures that the capacity is at least equal to the specified minimum.

13. **void getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)**: This method characters are copied from this sequence into the destination character array dst.

14. **int indexOf()**: This method returns the index within this string of the first occurrence of the specified substring.

15. **StringBuilder insert(int offset, boolean b)**: This method inserts the string representation of the booalternatelean argument into this sequence.

16. **StringBuilder insert()**: This method inserts the string representation of the char argument into this sequence.

17. **int lastIndexOf()**: This method returns the index within this string of the last occurrence of the specified substring.

18. **int length()**: This method returns the length (character count).

19. **int offsetByCodePoints(int index, int codePointOffset)**: This method returns the index within this sequence that is offset from the given index by codePointOffset code points.

20. **StringBuilder replace(int start, int end, String str)**: This method replaces the characters in a substring of this sequence with characters in the specified String.

21. **StringBuilder reverse()**: This method causes this character sequence to be replaced by the reverse of the sequence.

22. **void setCharAt(int index, char ch)**: In this method, the character at the specified index is set to ch.

23. **void setLength(int newLength)**: This method sets the length of the character sequence.

24. **CharSequence subSequence(int start, int end)**: This method returns a new character sequence that is a subsequence of this sequence.

25. **String substring()**: This method returns a new String that contains a subsequence of characters currently contained in this character sequence.

26. **String toString()**: This method returns a string representing the data in this sequence.

27. **void trimToSize()**: This method attempts to reduce storage used for the character sequence.

Example:

```
// Java code to illustrate
// methods of StringBuilder

import java.util.*;
import java.util.concurrent.LinkedBlockingQueue;

public class GFG1 {
    public static void main(String[] argv)
        throws Exception
```

```
    {

        // create a StringBuilder object
        // with a String pass as parameter
        StringBuilder str
            = new StringBuilder("AAAABBBCCCC");

        // print string
        System.out.println("String = "
                           + str.toString());

        // reverse the string
        StringBuilder reverseStr = str.reverse();

        // print string
        System.out.println("Reverse String = "
                           + reverseStr.toString());

        // Append ', '(44) to the String
        str.appendCodePoint(44);

        // Print the modified String
        System.out.println("Modified StringBuilder = "
                           + str);

        // get capacity
        int capacity = str.capacity();

        // print the result
        System.out.println("StringBuilder = " + str);
        System.out.println("Capacity of StringBuilder = "
                           + capacity);
    }
}
```

Output:

```
String = AAAABBBCCCC
Reverse String = CCCCBBBAAAA
Modified StringBuilder = CCCCBBBAAAA,
StringBuilder = CCCCBBBAAAA,
Capacity of StringBuilder = 27
```

Reference: https://docs.oracle.com/javase/9/docs/api/java/lang/StringBuilder.html