**String**

**String** is a final and **immutable class**, which makes it the most special. It cannot be inherited, and once created, we can not alter the object. **String** object is one of the most-used objects in any of the programs.

**String array**

A **String Array** is an **Array** of a fixed number of **String** values. A **String** is a sequence of characters. Generally, a **string** is an immutable object, which means the value of the **string** can not be changed. The **String Array** works similarly to other data types of **Array**. In **Array**, only a fixed set of elements can be stored.

**String buffer**

**StringBuffer in java** is **used** to create modifiable **String** objects. This means that we can **use StringBuffer** to append, reverse, replace, concatenate and manipulate **Strings** or sequence of characters. Corresponding methods under **StringBuffer** class are respectively created to adhere to these functions.

Example 1:

public static void main(String[] args)

    {

        StringBuffer s = new StringBuffer("GeeksforGeeks");

        int p = s.length();

        int q = s.capacity();

        System.out.println("Length of string GeeksforGeeks=" + p);

        System.out.println("Capacity of string GeeksforGeeks=" + q);

    }

Example 2:

public static void main(String[] args)

    {

        StringBuffer s = new StringBuffer("Geeksfor");

        s.append("Geeks");

        System.out.println(s); // returns GeeksforGeeks

        s.append(1);

        System.out.println(s); // returns GeeksforGeeks1

    }

String Builder

**StringBuffer** is synchronized i.e. thread safe. It means two threads can't call the methods of **StringBuffer** simultaneously. **StringBuilder** is non-synchronized i.e. not thread safe. It means two threads can call the methods of **StringBuilder** simultaneously.

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);

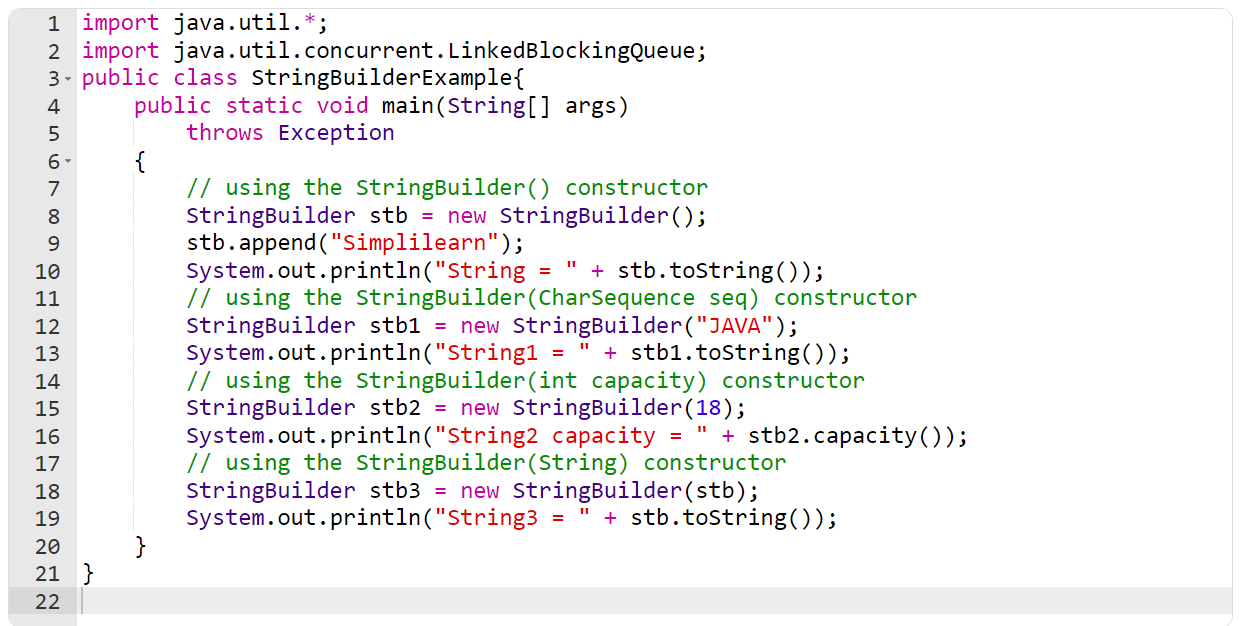
    }

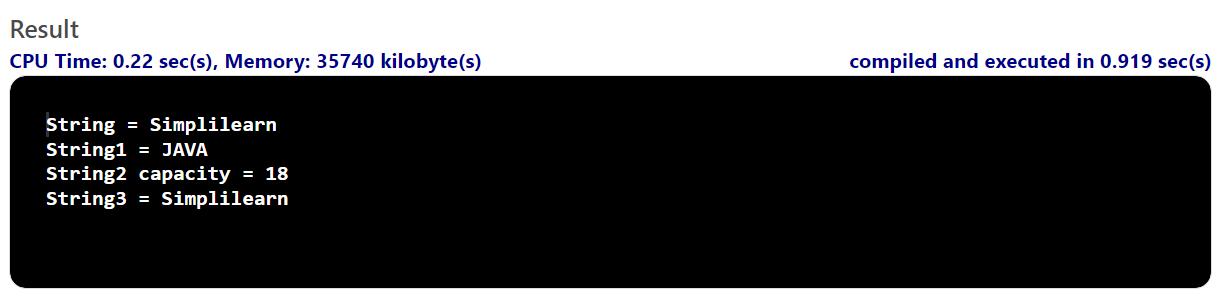
Constructors of StringBuilder in Java

The following table lists and describes the constructors of StringBuilder in Java

|  |  |
| --- | --- |
| Constructor Name | Description |
| StringBuilder() | It constructs a blank string builder with a capacity of 16 characters |
| StringBuilder(int capacity) | It creates an empty string builder with the specified capacity |
| StringBuilder(CharSequence seq) | It creates a string builder with the same characters specified as the argument |
| StringBuilder(String str) | It will construct a string builder with the string specified in the argument |

Since you now know about the constructors and class declaration of StringBuilder in Java, it’s time to look at an example where you will use some of these constructors to create various sequences of characters.





The StringBuilder in Java provides numerous methods to perform different operations on the string builder. The table depicted below enlists some primary methods from the StringBuilder class.

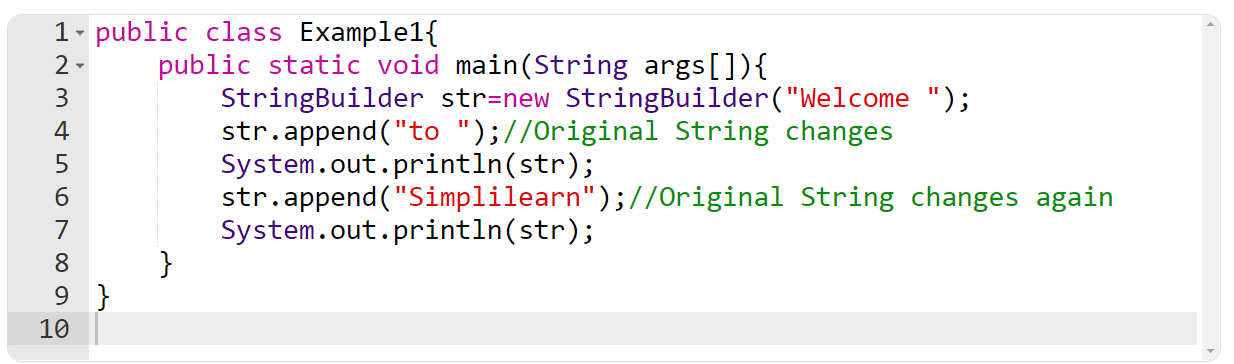
|  |  |
| --- | --- |
| Method | Description |
| StringBuilder append (String s) | This method appends the mentioned string with the existing string. You can also with arguments like boolean, char, int, double, float, etc. |
| StringBuilder insert (int offset, String s) | It will insert the mentioned string to the other string from the specified offset position. Like append, you can overload this method with arguments like (int, boolean), (int, int), (int, char), (int, double), (int, float), etc. |
| StringBuilder replace(int start, int end, String s) | It will replace the original string with the specified string from the start index till the end index. |
| StringBuilder delete(int start, int end) | This method will delete the string from the mentioned start index till the end index. |
| StringBuilder reverse() | It will reverse the string. |
| int capacity() | This will show the current StringBuilder capacity. |
| void ensureCapacity(int min) | This method ensures that the StringBuilder capacity is at least equal to the mentioned minimum. |
| char charAt(int index) | It will return the character at the specified index. |
| int length() | This method is used to return the length (total characters) of the string. |
| String substring(int start) | Starting from the specified index till the end, this method will return the substring. |
| String substring(int start, int end) | It will return the substring from the start index till the end index. |
| int indexOf(String str) | This method will return the index where the first instance of the specified string occurs. |
| int lastIndexOf(String str) | It will return the index where the specified string occurs the last. |
| Void trimToSize() | It will attempt to reduce the size of the StringBuilder. |

Using the Methods of StringBuilder in Java

Let’s have a look at examples of some examples of the StringBuilder methods.

Example 1: Applying the Append() Method of StringBuilder in Java

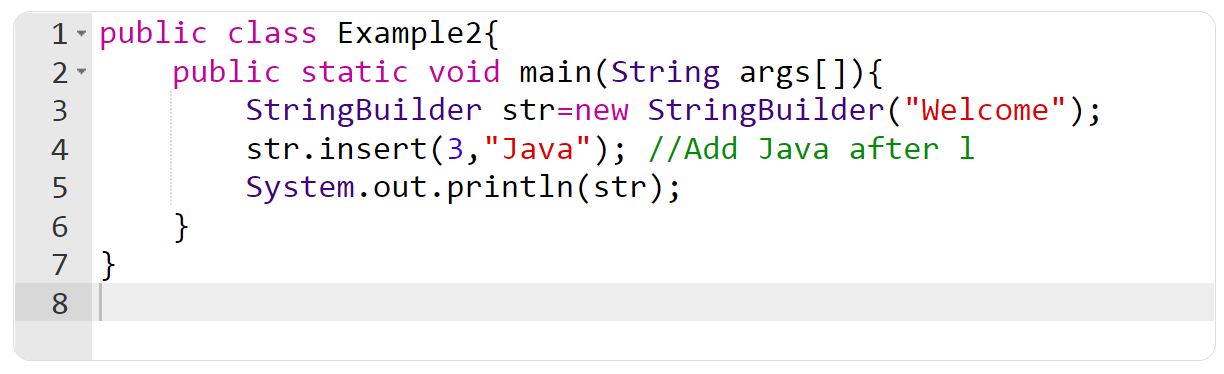
Here, you must concatenate three strings using the append() method in the below example.





Example 2: Inserting String With the Insert() Method

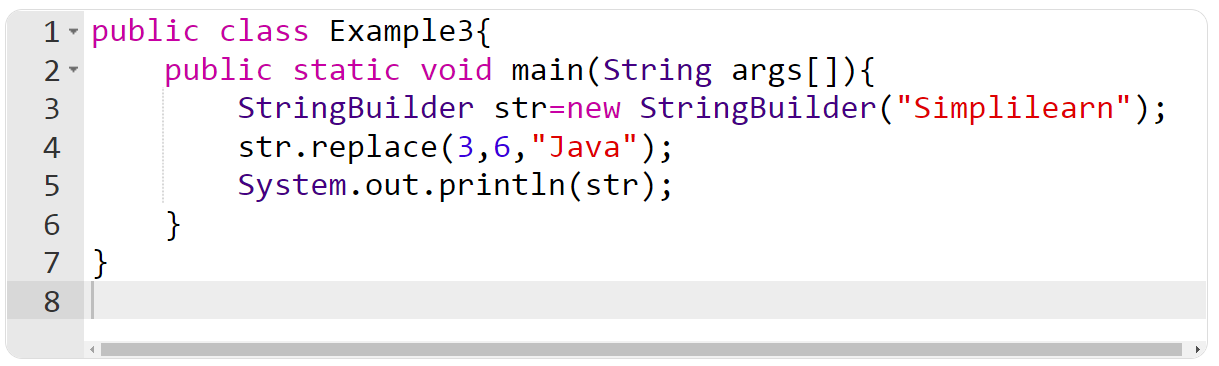
In this example, you will insert one string into another at the specified index.





Example 3: Using the Replace() Method of StringBuilder in Java

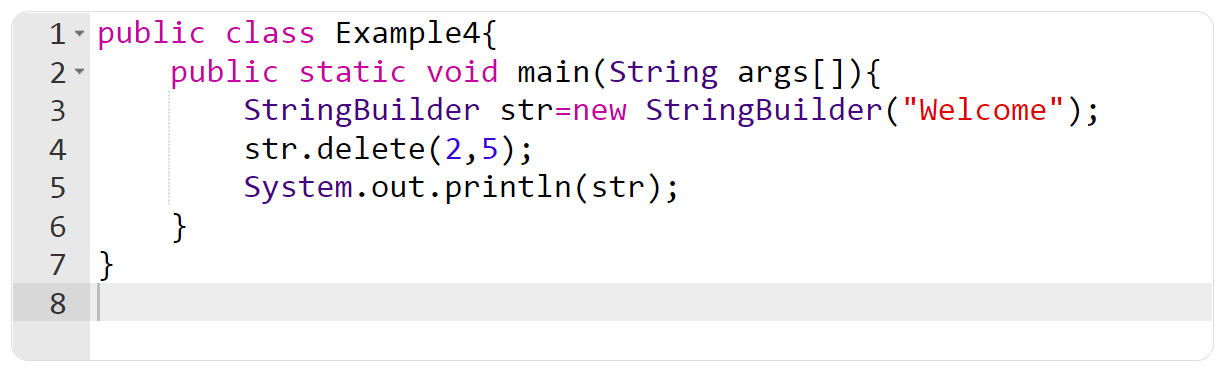
You will use the replace() method to edit Simplilearn, and insert Java from a specified start and end index.





Example 4: Deleting a Substring From the Original String

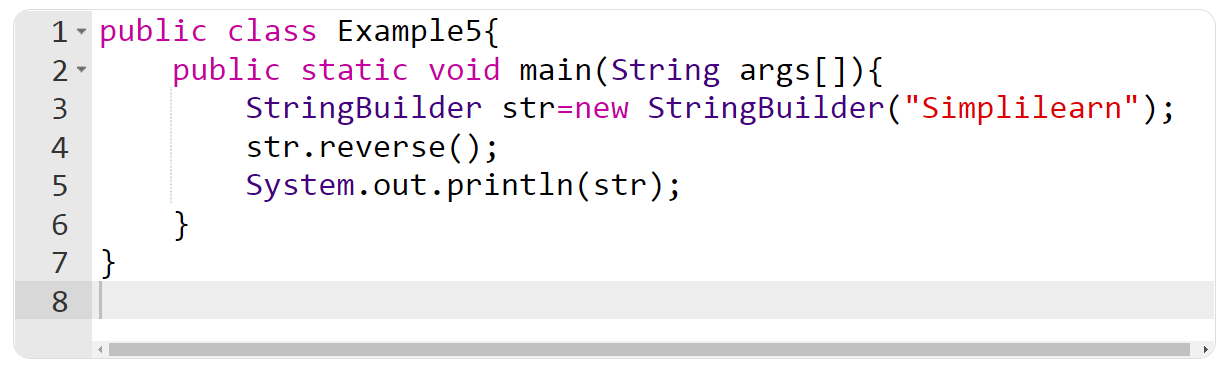
The delete() method in the below example will delete some strings according to the specified indexes.





Example 5: Applying the Reverse() Method of StringBuilder in Java

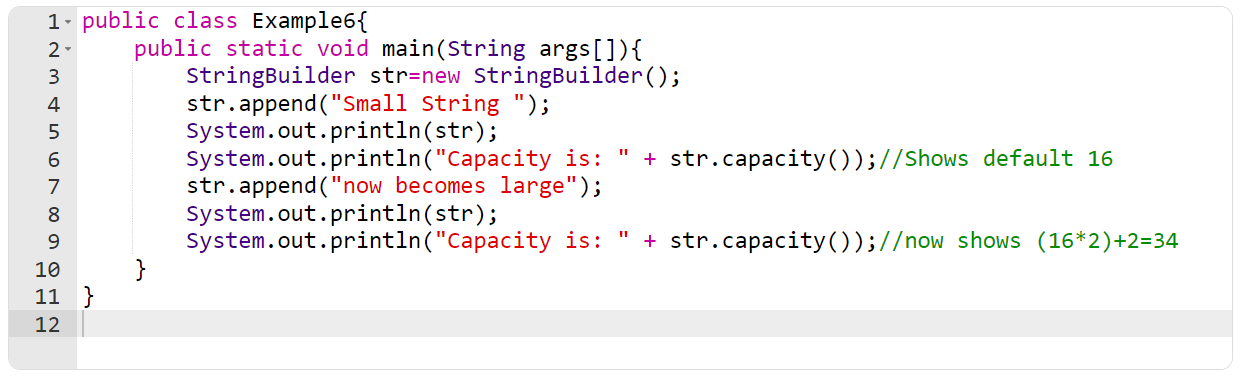
You will reverse “Simplilearn” in the example below with the reverse() method.





Example 6: Looking at the Capacity() Method

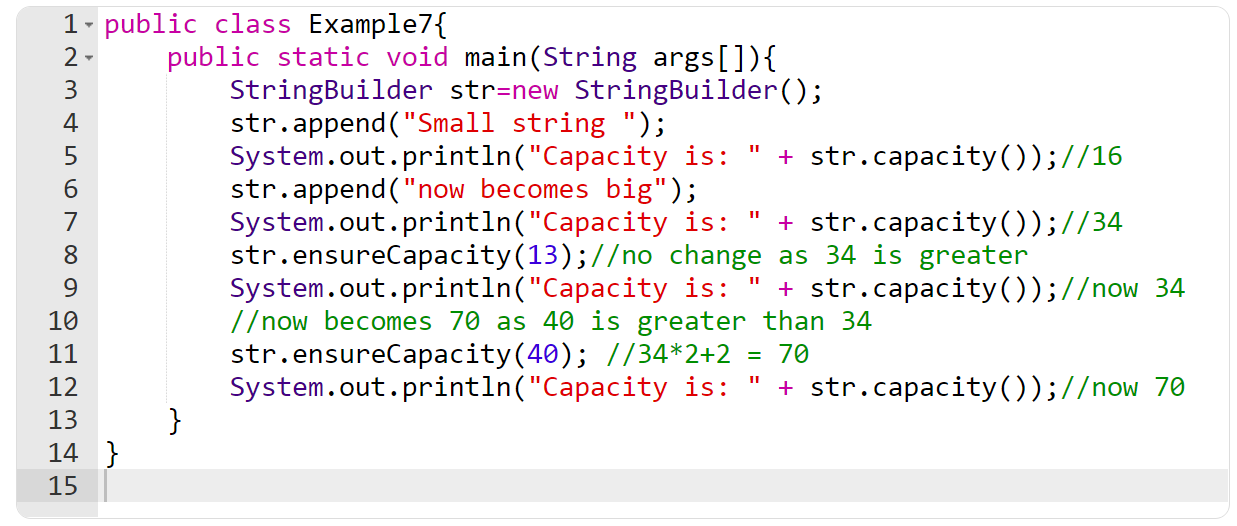
You will find out the current capacity of a StringBuilder using the capacity() method. The default capacity is 16. When the number of characters exceeds 16, it increases the capacity to n\*2+2, where n is the current capacity. Let’s look at the example.





Example 7: Ensure Minimum Capacity With the Ensurecapacity() Method

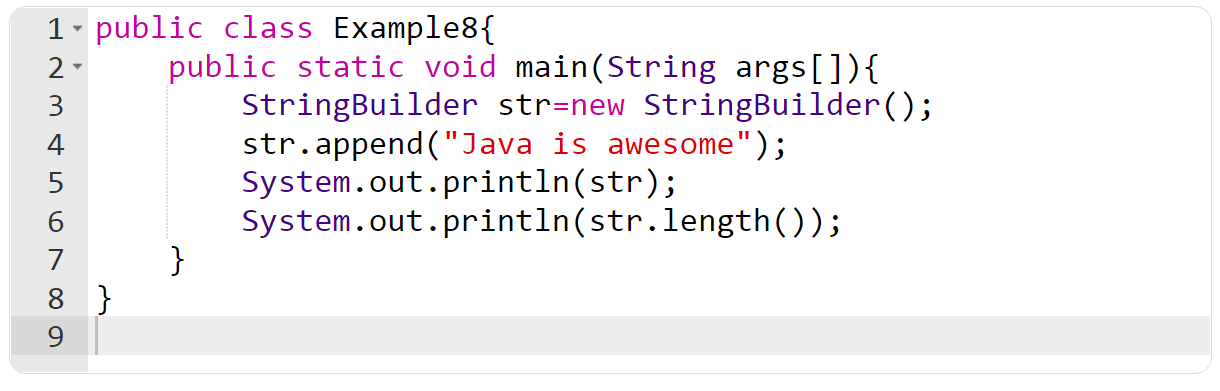
In this example, you will use the ensureCapacity() method of StringBuilder in Java to ensure that there is the minimum capacity before proceeding with other operations.





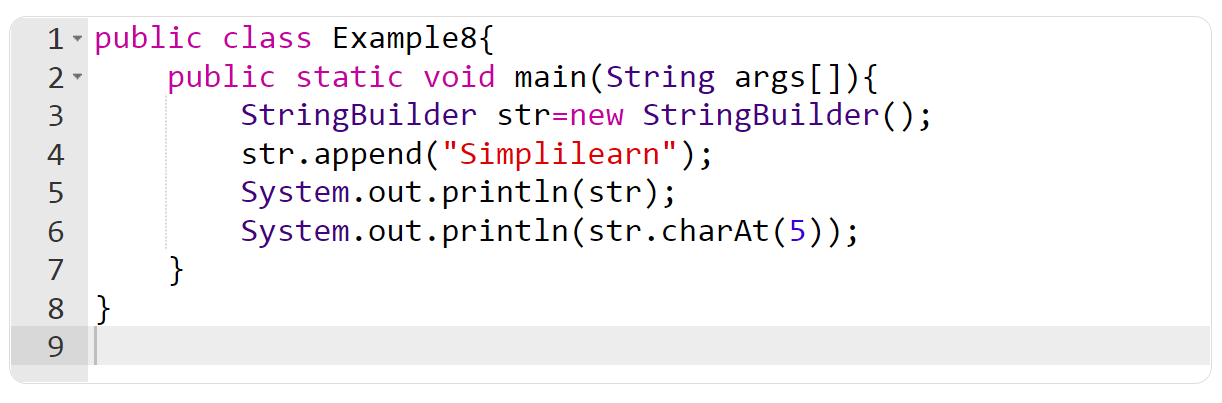
Example 8: Using the Length() Method of Stringbuilder in Java

In this example, you will use the length() method to find the total number of characters in a string.





Example 9: Looking at the Charat() Method

Using the charAt() method, you will find the character present at the specified index in the string.



Example 10: The Indexof() Method of StringBuilder in Java

Next, use the indexOf() method to find the index of the specified string from the original string.

