

What is **StringBuilder**?

In Java, **StringBuilder** is a **mutable sequence of characters**, meaning:

- You can **change** (append, insert, delete, etc.) its content **without creating new objects**.
 - It's part of `java.lang` (so no import is needed).
-

Why is **StringBuilder** Important?

String is immutable:

java

CopyEdit

```
String s = "a";  
s += "b"; // creates a new String object each time
```

This is **slow** in loops or large operations, because:

- Every time you do `+=` or `.concat()`, a new **String** is created.

StringBuilder is efficient:

java

CopyEdit

```
StringBuilder sb = new StringBuilder("a");  
sb.append("b"); // modifies the same object
```

- Faster and more memory efficient for string manipulations.
-

Key **StringBuilder** Functions:

Method	Description	Example
<code>append(String s)</code>	Adds text to the end	<code>sb.append("abc");</code>
<code>insert(int offset, s)</code>	Inserts text at position <code>offset</code>	<code>sb.insert(2, "xyz");</code>
<code>delete(int start, end)</code>	Deletes characters from <code>start</code> to <code>end-1</code>	<code>sb.delete(1, 3);</code>
<code>deleteCharAt(int index)</code>	Deletes a single character	<code>sb.deleteCharAt(0);</code>
<code>reverse()</code>	Reverses the string	<code>sb.reverse();</code>
<code>replace(start, end, s)</code>	Replaces part with new string	<code>sb.replace(0, 2, "xy");</code>
<code>charAt(int index)</code>	Returns char at index	<code>sb.charAt(1);</code>
<code>setCharAt(index, ch)</code>	Changes the char at a specific index	<code>sb.setCharAt(1, 'z');</code>
<code>length()</code>	Returns length	<code>sb.length();</code>
<code>toString()</code>	Converts it to a regular <code>String</code>	<code>sb.toString();</code>



Example Code:

java

CopyEdit

```
public class Main {
    public static void main(String[] args) {
        StringBuilder sb = new StringBuilder("Hello");

        sb.append(" World");           // Hello World
        sb.insert(5, ",");             // Hello, World
        sb.replace(0, 5, "Hi");        // Hi, World
        sb.deleteCharAt(2);            // Hi World
        sb.reverse();                  // dlroW iH
    }
}
```

```
        System.out.println(sb.toString()); // Output: dlroW iH
    }
}
```






Performance Note:

Use `StringBuilder`:

- In **loops** that involve string operations.
 - For **string concatenation** in competitive coding, web apps, large text processing.
-



StringBuffer vs StringBuilder

Feature	StringBuffer	StringBuilder
Thread-safe	Yes	 No
Slower		 Faster
Use case	Multi-threaded	Single-threaded