**StringBuilder in Java**

**1. What is StringBuilder?**

* StringBuilder is a **mutable sequence of characters** in Java.
* Unlike String (immutable) and StringBuffer (thread-safe), StringBuilder is:
  + **Mutable** → You can change content without creating new objects.
  + **Faster** → Because it is **not synchronized** (no thread safety overhead).
  + **Part of java.lang package** → No need to import separately.

**2. Why was StringBuilder introduced?**

* String objects are **immutable** → Every modification (concatenation, replace, etc.) creates a **new object in memory**.  
  🔸 Example: "A" + "B" + "C" creates multiple intermediate objects.
* For operations like:
  + **Appending**
  + **Deleting**
  + **Replacing**
  + **Reversing**
  + **Inserting**

we needed something **faster and memory efficient** → that’s why **StringBuilder** was introduced (Java 1.5).

**3. When to use StringBuilder?**

Use **StringBuilder** when:

* You need to **modify strings frequently** in **single-threaded applications**.
* Example cases:
  + Building large text (logs, SQL queries, HTML/XML).
  + Performing loops with string concatenation.
  + Reversing strings or manipulating substrings.

❌ Don’t use **StringBuilder** when:

* You need **thread-safety** → use **StringBuffer** instead.
* You don’t need modification → simple **String** is enough.

**4. How StringBuilder works internally?**

* Maintains a **char[] array buffer** inside.
* Default capacity: **16 characters**.
* If exceeded → automatically resizes (new capacity = old capacity × 2 + 2).
* length() → actual number of characters stored.
* capacity() → size of the buffer array.

**5. Important Methods in StringBuilder**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Method | Description | Example | | append() | Adds text at the end | sb.append("Hello"); | | insert(int, str) | Inserts text at index | sb.insert(5, "World"); | | replace(int, int, str) | Replace range with text | sb.replace(0, 5, "Hi"); | | delete(int, int) | Deletes chars from range | sb.delete(0, 3); | | reverse() | Reverses text | sb.reverse(); | | length() | Current length | sb.length(); | | capacity() | Buffer size | sb.capacity(); | | charAt(int) | Get character | sb.charAt(2); | | setCharAt(int, char) | Modify character | sb.setCharAt(0, 'P'); | |

**6. Code Example:**

public class SBuilder {

    public static void main(String[] args){

        //Demonstration of a StringBuilder class in Java.

        // Create a new StringBuilder with the initial content "GeeksforGeeks"

       /\* StringBuilder sb5 = new StringBuilder("GeeksforGeeks");

        System.out.println("Initial StringBuilder: " + sb5);

        // Append a string to the StringBuilder

        sb5.append(" is awesome!");

        System.out.println("After append: " + sb5);\*/

        // 1. Default constructor (capacity = 16)

        StringBuilder sb1 = new StringBuilder();

        sb1.append("Hello");

        System.out.println("sb1----Default constructor : " + sb1);

        // 2. Constructor with capacity

        StringBuilder sb2 = new StringBuilder(50);

        sb2.append("This has initial capacity 50");

        System.out.println("sb2--------Constructor with capacity: " + sb2);

        // 3. Constructor with String input

        StringBuilder sb3 = new StringBuilder("Astinil");

        sb3.append("Technologies");

        System.out.println("sb3---------Constructor with String input: " + sb3);

        // 4. Constructor with CharSequence input

        CharSequence cs = "Java";

        StringBuilder sb4 = new StringBuilder(cs);

        sb4.append("Programming");

        System.out.println("sb4--------Constructor with CharSequence input: " + sb4);

        //Methods of StringBuilder class

        // Create a new StringBuilder with the initial content "Astinil Technologies"

        StringBuilder sb = new StringBuilder("Astinil Technologies");

        System.out.println("Initial StringBuilder: " + sb);

        // 1. Append a string to the StringBuilder

        sb.append(" is awesome!");

        System.out.println("After append: " + sb);

        // 2. Insert a substring at a specific position

        sb.insert(13, " Java");

        System.out.println("After insert: " + sb);

        // 3. Replace a substring with another string

        sb.replace(0, 5, "Welcome to");

        System.out.println("After replace: " + sb);

        // 4. Delete a substring from the StringBuilder

        sb.delete(8, 14);

        System.out.println("After delete: " + sb);

        // 5. Reverse the content of the StringBuilder

        sb.reverse();

        System.out.println("After reverse: " + sb);

        // 6. Get the current capacity of the StringBuilder

        int capacity = sb.capacity();

        System.out.println("Current capacity: " + capacity);

        // 7. Get the length of the StringBuilder

        int length = sb.length();

        System.out.println("Current length: " + length);

        // 8. Access a character at a specific index

        char charAt5 = sb.charAt(5);

        System.out.println("Character at index 5: " + charAt5);

        // 9. Set a character at a specific index

        sb.setCharAt(5, 'X');

        System.out.println("After setCharAt: " + sb);

        // 10. Get a substring from the StringBuilder

        String substring = sb.substring(5, 10);

        System.out.println("Substring (5 to 10): " + substring);

        // 11. Find the index of a specific substring

        sb.reverse(); // Reversing back to original order for search

        int indexOfAT = sb.indexOf("Astinil");

        System.out.println("Index of 'Astinil': " + indexOfAT);

        // 12. Delete a character at a specific index

        sb.deleteCharAt(5);

        System.out.println("After deleteCharAt: " + sb);

        // 13. Convert the StringBuilder to a String

        String result = sb.toString();

        System.out.println("Final String: " + result);

    }

}

**Output:**

sb1----Default constructor : Hello

sb2--------Constructor with capacity: This has initial capacity 50

sb3---------Constructor with String input: AstinilTechnologies

sb4--------Constructor with CharSequence input: JavaProgramming

Initial StringBuilder: Astinil Technologies

After append: Astinil Technologies is awesome!

After insert: Astinil Techn Javaologies is awesome!

After replace: Welcome toil Techn Javaologies is awesome!

After delete: Welcome echn Javaologies is awesome!

After reverse: !emosewa si seigoloavaJ nhce emocleW

Current capacity: 74

Current length: 36

Character at index 5: e

After setCharAt: !emosXwa si seigoloavaJ nhce emocleW

Substring (5 to 10): Xwa s

Index of 'Astinil': -1

After deleteCharAt: Welcoe echn Javaologies is awXsome!

Final String: Welcoe echn Javaologies is awXsome!

7.**Block Diagram – How StringBuilder Works:**

String Builder

Internal char[] buffer

(Default capacity = 16)

append() insert() delete()

Updates same character array

**8. Difference Between String, StringBuffer, StringBuilder**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | Feature | String (Immutable) | StringBuffer (Mutable + Thread-safe) | StringBuilder (Mutable, Fast, Not Thread-safe) | | Mutability | ❌ No | ✅ Yes | ✅ Yes | | Performance | Slow (creates new) | Moderate (synchronized = slower) | Fast (no synchronization) | | Thread-safe | N/A | ✅ Yes | ❌ No | | Best Use | Fixed data | Multi-threaded apps | Single-threaded apps | |

**Real-Time Scenarios:**

1. **Banking Applications**
   * Generating account statements dynamically with customer name, transactions, balances.

 **E-commerce Applications**

* Building dynamic order confirmation messages or invoices.

 **Loggers**

* Concatenating log messages efficiently.

 **Data Processing**

* Reading and formatting large files, reports, or SQL queries.

## Summary

* **StringBuilder** is a **mutable, fast string manipulation class**.
* Best for **single-threaded** applications where you modify strings often.
* More efficient than String (immutable).
* Lighter and faster than StringBuffer (since no synchronization).
* Used in **banking, e-commerce, logging, reports** etc