

**1. String (Immutable)**

* **Immutable**: Once a String object is created, it **cannot be changed**.
* When you modify a String, a **new object** is created.

**✅ Example:**

public class StringExample {

public static void main(String[] args) {

String s1 = "Hello";

s1.concat(" World"); // This creates a new string, doesn't change s1

System.out.println(s1); // Output: Hello

// If we store the result in a new variable or overwrite s1:

s1 = s1.concat(" World");

System.out.println(s1); // Output: Hello World

}

}

**🔹 2. StringBuffer (Mutable & Thread-safe)**

* **Mutable**: The content can be changed without creating a new object.
* **Thread-safe**: Synchronized, so it's **safe for use in multithreaded environments**, but slightly slower.

**✅ Example:**

java

CopyEdit

public class StringBufferExample {

public static void main(String[] args) {

StringBuffer sb = new StringBuffer("Hello");

sb.append(" World"); // Modifies the same object

System.out.println(sb); // Output: Hello World

sb.insert(5, ","); // Insert comma at index 5

System.out.println(sb); // Output: Hello, World

sb.reverse(); // Reverse the content

System.out.println(sb); // Output: dlroW ,olleH

}

}

**🔹 3. StringBuilder (Mutable but Not Thread-safe)**

* **Mutable** like StringBuffer
* **Faster** than StringBuffer, but **not synchronized**
* Use it in **single-threaded** scenarios for better performance.

**✅ Example:**

java

CopyEdit

public class StringBuilderExample {

public static void main(String[] args) {

StringBuilder sb = new StringBuilder("Hello");

sb.append(" World");

System.out.println(sb); // Output: Hello World

sb.replace(6, 11, "Java");

System.out.println(sb); // Output: Hello Java

sb.delete(5, 10);

System.out.println(sb); // Output: Hello

}

}

**🔸 Summary Table:**

| **Feature** | **String** | **StringBuffer** | **StringBuilder** |
| --- | --- | --- | --- |
| **Mutable** | ❌ No | ✅ Yes | ✅ Yes |
| **Thread-safe** | ✅ Yes (Immutable) | ✅ Yes (Synchronized) | ❌ No |
| **Performance** | Slow (new object on change) | Slower (due to sync) | Fast (no sync) |
| **Best Use** | Constant strings | Multithreaded apps | Single-threaded apps |

**🔍 When to Use What?**

* ✅ Use **String** when the string **won’t change** (e.g., constants, keys).
* ✅ Use **StringBuffer** for **multithreaded** string operations.
* ✅ Use **StringBuilder** for **high-performance** string manipulation in a **single thread**.