**Is String Mutable in Java?**

No, **String in Java is immutable**. This means **once a String object is created, it cannot be changed**. Any modification creates a new String object instead of modifying the existing one.

**Why is String Immutable?**

1️⃣ **Memory Efficiency (String Pooling)**

* Java maintains a **String Pool** in the heap memory.
* If two strings have the same value, Java **reuses** the existing object instead of creating a new one.
* Example:
* String s1 = "Hello";
* String s2 = "Hello"; // Reuses the same object from the String Pool
* System.out.println(s1 == s2); // true (same memory reference)

2️⃣ **Thread-Safety**

* Since strings cannot be changed, multiple threads can safely share the same string without synchronization issues.
* Example:
* String s = "Hello";
* s.concat(" World"); // Creates a new object instead of modifying s
* System.out.println(s); // "Hello" (original string unchanged)

3️⃣ **Security Reasons**

* String is widely used in **passwords, URLs, file paths, and database connections**.
* If String were mutable, it could be modified unintentionally or maliciously.
* Example: If a password string were mutable, a hacker could modify its value while it's being used in authentication.

4️⃣ **Caching Hashcode for Performance**

* String objects store their **hashcode** for efficient lookups in hash-based collections (HashMap, HashSet).
* Since strings are immutable, their hashcode never changes.

**Example: Why String is Immutable?**

public class StringImmutableExample {

public static void main(String[] args) {

String str = "Java";

str.concat(" Programming"); // New string is created, but not assigned to 'str'

System.out.println(str); // Output: "Java" (original string unchanged)

str = str.concat(" Programming"); // Now it points to the new string

System.out.println(str); // Output: "Java Programming"

}

}

🔹 **Conclusion:** String is immutable because of security, performance, and memory optimization reasons. If you need a **mutable** string, use **StringBuilder or StringBuffer**. 🚀