Feature / Property	String	StringBuilder	StringBuffer
Mutability	**Immutable** – once created, its value cannot be changed. Any modification (concat, replace, substring) creates a *new* object in memory.	**Mutable** – content can be modified (append, insert, delete) without creating new objects. Great for repeated changes.	**Mutable** – same as StringBuilder, supports append, insert, delete without new objects, but with synchronization.
Thread Safety	**Thread-safe by design** because it is immutable. Even if shared across threads, it cannot be altered.	❌ **Not thread-safe**. If used by multiple threads, they may modify the value at the same time, leading to inconsistent results.	✓ **Thread-safe** because all methods (append, insert, delete) are synchronized, making it safe for multi-threaded usage.
Performance	**Slow for modifications** because each change creates a new object. Good for reading operations but not writing.	**Fastest** for building or modifying strings because no synchronization and no new object creation.	**Slower than StringBuilder** because synchronization adds overhead, but still faster than String for modifications.
Memory Usage	Uses more memory when frequently modified because each change creates a new object in heap or string pool.	Memory-efficient for repeated changes since it reuses the same buffer.	Memory-efficient like StringBuilder, but extra synchronization adds small overhead.
Best Use Case	Best for **constant text**, configuration values, messages, or when immutability is important.	Best for **frequent string modifications**, especially inside loops, in a single thread.	Best for **frequent string modifications** in **multi-threaded programs** where thread safety is required.
Synchronization	No synchronization needed. Safe inherently due to immutability.	❌ No synchronization. Must be manually synchronized externally if used in multiple threads.	✓ All important methods are **synchronized**, preventing concurrent modifications.
Stored in String Pool?	✓ Yes, string literals go into the **String pool**, allowing reusability and memory optimization.	❌ No. Objects are stored in the heap, not the pool.	❌ No. Stored in regular heap memory.
Modification Behavior	Creates new objects every time you modify, causing memory & performance overhead in loops.	Modifies the same internal character array, making it very efficient for concatenation.	Same as StringBuilder, but methods are synchronized to control thread access.
Use in Loops / Heavy Updates	⚠ Not recommended – leads to many unnecessary objects and poor performance.	✓ Highly recommended – best performance under heavy modifications.	✓ Recommended only when multiple threads modify the same text.
Speed Ranking	🟥 **Slowest**	🟩 **Fastest**	🟨 Slightly slower than StringBuilder
Thread Environment	Works in **any environment** since immutable.	Best for **single-threaded** apps.	Best for **multi-threaded** applications.
Security	High – immutability makes it good for storing sensitive data like passwords (but still not ideal).	Lower security since values can be modified.	Higher than StringBuilder because of synchronized access.