Difference Between String, StringBuffer, and StringBuilder in Java

Feature	String (Immutable)	StringBuffer (Mutable, Thread-Safe)	StringBuilder (Mutable, Fast, Not Thread-Safe)
Mutability	Immutable (Cannot be changed after creation)	Mutable (Can be modified)	Mutable (Can be modified)
Performance	Slow (Creates a new object every time it is modified)	Faster than String (Uses the same object)	Fastest (No synchronization overhead)
Thread Safety	Thread-safe (Immutable)	Thread-safe (Uses synchronization)	Not thread-safe (No synchronization)
Usage in Multi-Threading	Safe to use in multi-threaded applications	Preferred for multi-threading	Not recommended for multi-threading
Efficiency in Concatenation	Inefficient (Creates multiple objects)	Efficient (Modifies the same object)	Most efficient (Better than StringBuffer)
Method Synchronization	No synchronization required	Methods are synchronized	Methods are not synchronized
Best Used For	When text does not change frequently	When thread-safety is required with frequent modifications	When high performance is needed and no thread-safety is required
Example Usage	String s = "Hello"; s += " World";	StringBuffer sb = new StringBuffer("Hello"); sb.append(" World");	StringBuilder sb = new StringBuilder("Hello"); sb.append(" World");

Conclusion:

- Use **String** if the value **does not change frequently**.
- Use **StringBuffer** when **multiple threads are modifying the same string**.
- Use **StringBuilder** when **you need fast performance in a single-threaded environment**.