Difference Between String, StringBuffer & StringBuilder

Feature	String	StringBuffer	StringBuilder
Mutability	**Immutable** (cannot be changed)	**Mutable** (can be changed)	**Mutable** (can be changed)
Thread-Safe	Yes (because it is immutable)	Yes (synchronized)	No (not synchronized)
Performance	Slow when modified repeatedly	Slower than StringBuilder	Fastest
Memory Usage	Uses more memory if modified often	Uses less memory than String	Uses least memory
Synchronization	Not needed	Uses synchronization	No synchronization
Introduced In	Java 1.0	Java 1.0	Java 1.5
Use Case	Fixed text or rarely changed text	Multi-threaded environment	Single-threaded environment
String Modification	Creates a new object	Modifies the same object	Modifies the same object
Speed of Operations	Slow	Medium	Fast
Example Methods	concat(), length()	append(), insert(), delete()	append(), insert(), delete()

Detailed Explanation (Simple and Natural Language)

1. String

- **String objects are immutable**, which means once a String is created, it **cannot be changed**.
- If you modify a String, a **new object is created in memory**.
- Because of immutability, Strings are **automatically thread-safe**.

- Best used when the text **does not change frequently**.

Example:

```
String s = "Hello";
s = s + " World";  // New object created
```

Problem:

- If you modify Strings many times (like inside loops), it **wastes memory and time**.

2. StringBuffer

- **StringBuffer is mutable**, so it **can be changed without creating a new object**.
- It is **thread-safe** because its methods are **synchronized**.
- Suitable for **multi-threaded applications** where multiple threads work on the same data.
- Slightly **slower due to synchronization overhead**.

Example:

```
StringBuffer sb = new StringBuffer("Hello");
sb.append(" World");  // Same object modified
```

Use When:

- Multiple threads are accessing and modifying the same string data.

3. StringBuilder

- **StringBuilder is also mutable**, just like StringBuffer.
- It is **not thread-safe**, which makes it **faster**.
- Best choice for **single-threaded applications**.
- Commonly used in loops and large string operations.

Example:

```
StringBuilder sb = new StringBuilder("Hello");
sb.append(" World");  // Fast and efficient
```