10-StringBuffer and StringBuilder

Java provides three classes to represent a sequence of characters: **String**, **StringBuffer**, and **StringBuilder**. The String class is immutable, while StringBuffer and StringBuilder classes are mutable.

- StringBuilder was introduced in JDK 1.5.
- StringBuffer was introduced in Java 1.0.

StringBuffer

StringBuffer represents growable and writable character sequences. Unlike String, which represents fixed-length, immutable character sequences, StringBuffer allows characters and substrings to be inserted in the middle or appended to the end. It will automatically grow to accommodate these additions and often reallocates more characters than needed to allow room for growth.

Advantages of StringBuffer:

- **Mutable and Thread-Safe:** Suitable for multi-threaded environments where string operations need to be performed safely across multiple threads.
- **Efficient for Most String Manipulations**: Reliable in scenarios where immutability is not a requirement.

Disadvantages of StringBuffer:

- **Slower Performance**: The synchronization overhead makes StringBuffer slower than StringBuilder.
- **Not Ideal for Single-Threaded Scenarios**: When immutability and maximum performance are required, StringBuffer may not be the best choice.

Example:

```
StringBuffer sb = new StringBuffer("navin");

System.out.println("Capacity: " + sb.capacity()); // Output: 21

System.out.println("Length: " + sb.length());     // Output: 5
```

```
sb.append(" reddy");

System.out.println(sb); // Output: navin reddy
```

Explanation:

- The initial capacity of StringBuffer is 16 bytes. When "navin" (5 characters) is added, the total capacity becomes 21.
- The append() method adds "reddy" to the existing string.

StringBuilder

StringBuilder is similar to StringBuffer but is not thread-safe, meaning it is not synchronized. It was introduced in JDK 1.5 and provides an efficient way to handle mutable strings in single-threaded environments.

Advantages of StringBuilder:

- **Mutable:** Allows efficient modification of string content without creating new objects.
- **High Performance:** Ideal for tasks that involve frequent and extensive string manipulations.
- **Efficient Memory Usage**: Minimizes memory overhead by reusing its internal buffer.

Disadvantages of StringBuilder:

- **Not Thread-Safe:** Requires manual handling of thread safety if used in multi-threaded environments.
- **Not Suitable for Immutable Scenarios:** If immutability is needed, StringBuilder is not the right choice.

Example:

```
StringBuilder sb = new StringBuilder("navin");

sb.append(" reddy");

System.out.println(sb); // Output: navin reddy

Comparison of StringBuffer and StringBuilder
```

Feature	StringBuffer	StringBuilder
Thread Safety	Synchronized (thread-safe)	Not synchronized (not thread-safe)
Performance	Slower due to synchronization overhead	Faster due to the absence of synchronization
Introduced in	Java 1.0	JDK 1.5

Conclusion

In conclusion, the choice between String, StringBuilder, and StringBuffer in Java depends on the specific requirements and constraints of your project. Each of these classes has its own set of advantages and disadvantages, making them suitable for different scenarios.