

# STRING DSA NOTES (JAVA)

## 1 Basics of String in Java

**String** = sequence of characters.

**Syntax:**

```
String s = "Om";
```

**Internally:** - Java 9+ → `byte[]` - Java 8 → `char[]`

**Immutable:** Once created, cannot change.

**Memory Model:** - **SCP (String Constant Pool):** Stores literals → reused for memory efficiency. - **Heap:** `new String()` objects → separate memory.

**String vs StringBuilder vs StringBuffer:**

Type	Mutable?	Thread-safe?	Performance	Use case
String	No	Yes	Slow	Small strings, keys
StringBuilder	Yes	No	Fast	Loops, concatenation
StringBuffer	Yes	Yes	Slower	Multithreading

## 2 Important String Operations

Operation	Example	Time Complexity
Length	<code>s.length()</code>	$O(1)$
CharAt	<code>s.charAt(0)</code>	$O(1)$
Substring	<code>s.substring(2,5)</code>	$O(N)$
Equals	<code>s1.equals(s2)</code>	$O(N)$
Replace	<code>s.replace('a','b')</code>	$O(N)$
Split	<code>s.split(" ")</code>	$O(N)$
ToCharArray	<code>s.toCharArray()</code>	$O(N)$
Concatenate	<code>s1 + s2</code>	$O(n^2)$ for multiple
StringBuilder append	<code>sb.append("x")</code>	$O(1)$ avg

**Tip:** Always use StringBuilder in loops.

---

### 3 Immutability & == vs equals

```
String s1 = "Om";
String s2 = new String("Om");

s1 == s2 // false (different objects)
s1.equals(s2) // true (same content)
```

**Immutability ensures:** - Thread safety - Security - Memory efficiency - Safe sharing of literals

---

### 4 Beginner Level Coding Patterns

- **Reverse string:** `StringBuilder.reverse()` or recursion
- **Palindrome check:** Two pointer from start & end
- **Count vowels/digits:** `Character.isDigit()` / `Character.isLetter()`
- **Remove spaces:** `s.replaceAll("\\s+", "")`
- **Frequency of characters:** `int[256]` or `HashMap`
- **First non-repeating character:** frequency array + iteration
- **Anagram check:** Sort strings + `Arrays.equals()`
- **Swap two strings without temp:**

```
a = a + b;
b = a.substring(0, a.length()-b.length());
a = a.substring(b.length());
```

### 5 Advanced Coding Patterns & Algorithms

#### 1) Reverse Recursively

```
reverseRec(s) = reverseRec(s.substring(1)) + s.charAt(0);
```