**Strings in Java — Full Connected Roadmap**

**1. What is a String? (Definition & Nature)**

A **String** is a sequence of characters (like "Hello").  
In Java:

* Strings are **objects** of the String class in java.lang package.
* They store **textual data**.
* They are **immutable** (we’ll see why soon).—cannot be modified.

Example:

String s1 = "Java"; // String literal

String s2 = new String("Hello"); // Using new keyword

These two ways look similar, but they behave differently in memory.

**2. How Strings are Stored (String Pool Concept)**

When you create a string using **string literal**:

String s1 = "Hello";

String s2 = "Hello";

* Java stores it in a **String Pool** (special memory in Heap).
* If another string with the same value is created, Java reuses the same object.
* Saves memory and improves performance.

If you use new keyword:

String s3 = new String("Hello");

* Java creates a **new object** in Heap (outside String Pool) even if same value exists.

**3. Why Strings are Immutable**

**Immutability** means once a String object is created, you cannot change its value.  
If you try to modify it, Java creates a **new object** instead of changing the old one.

Example:

String s = "Java";

s.concat(" Programming"); // doesn't change 's'

System.out.println(s); // Java

// Correct way

s = s.concat(" Programming");

System.out.println(s); // Java Programming

**Reasons for immutability:**

1. **Security** → Strings often store passwords, file paths, etc.
2. **Thread-safety** → Multiple threads can use the same string safely.
3. **Performance** → String Pool works only if strings don’t change.

**4. Ways to Create a String**

1. **String Literal** (uses String Pool)

String s1 = "Hello";

1. **Using new keyword** (creates new object)

String s2 = new String("Hello");

1. **From char array**

char[] ch = {'J','a','v','a'};

String s3 = new String(ch);

1. **From byte array**

byte[] b = {65, 66, 67};

String s4 = new String(b); // "ABC"

**5. String Comparison**

* **==** → compares references (memory location)
* **equals()** → compares content
* **compareTo()** → compares lexicographically (dictionary order)

Example:

String a = "Java";

String b = "Java";

String c = new String("Java");

System.out.println(a == b); // true (same pool object)

System.out.println(a == c); // false (different object)

System.out.println(a.equals(c)); // true (same content)

System.out.println("A".compareTo("B")); // -1 (A comes before B)

**6. Common String Methods**

| **Method** | **Purpose** | **Example** | **Output** |
| --- | --- | --- | --- |
| length() | length of string | "Java".length() | 4 |
| charAt(i) | char at index | "Java".charAt(2) | v |
| toUpperCase() | convert to caps | "java".toUpperCase() | JAVA |
| toLowerCase() | convert to lower | "JAVA".toLowerCase() | java |
| substring(start,end) | extract | "abcdef".substring(2,5) | cde |
| contains() | check substring | "Java".contains("av") | true |
| replace() | replace text | "car".replace("c","b") | bar |
| trim() | remove spaces | " hi ".trim() | hi |
| startsWith() | check start | "Java".startsWith("Ja") | true |
| endsWith() | check end | "Java".endsWith("va") | true |

**7. Searching in Strings**

* indexOf() → first occurrence
* lastIndexOf() → last occurrence

Example:

String str = "Java Programming";

System.out.println(str.indexOf("a")); // 1

System.out.println(str.lastIndexOf("a")); // 13

**8. Splitting and Joining Strings**

String data = "apple,banana,grapes";

String[] fruits = data.split(",");

for(String f : fruits) {

System.out.println(f);

}

String joined = String.join(" - ", "Java", "Python", "C++");

System.out.println(joined); // Java - Python - C++

**9. Mutable Alternatives — StringBuffer & StringBuilder**

Since Strings are immutable, Java offers:

* **StringBuffer** → Thread-safe, slower
* **StringBuilder** → Not thread-safe, faster

Example:

StringBuilder sb = new StringBuilder("Hello");

sb.append(" World");

System.out.println(sb); // Hello World

**10. String Formatting**

String name = "Bhargavi";

int age = 22;

System.out.printf("Name: %s, Age: %d", name, age);

**11. Strings & Regular Expressions**

String email = "abc@gmail.com";

System.out.println(email.matches("\\w+@gmail\\.com")); // true

**12. Conversion Between Strings & Other Types**

int num = 100;

String str = String.valueOf(num); // int -> string

int val = Integer.parseInt("123"); // string -> int

char[] arr = "Java".toCharArray(); // string -> char array