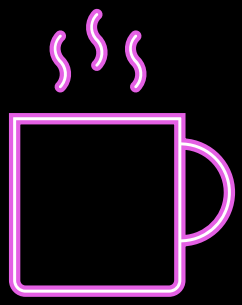
Mastering Strings in Java: Everything You Need to Know!

- From basics to best practices – let's unravel Java Strings.



1️⃣ What is a String in Java?

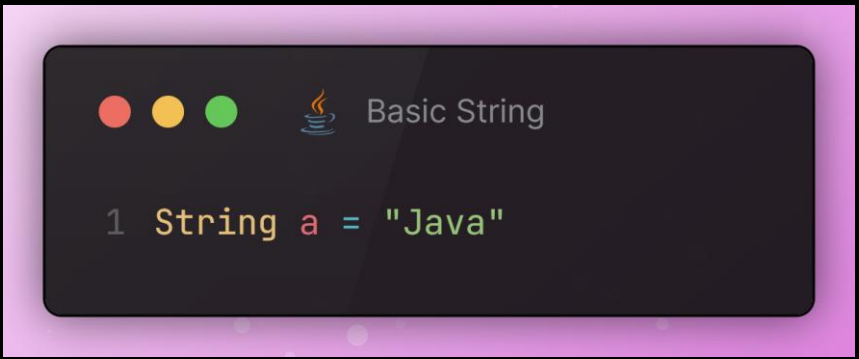
👉 String is a class in Java, not a primitive type.

👉 Immutable → Once created, it cannot be changed.

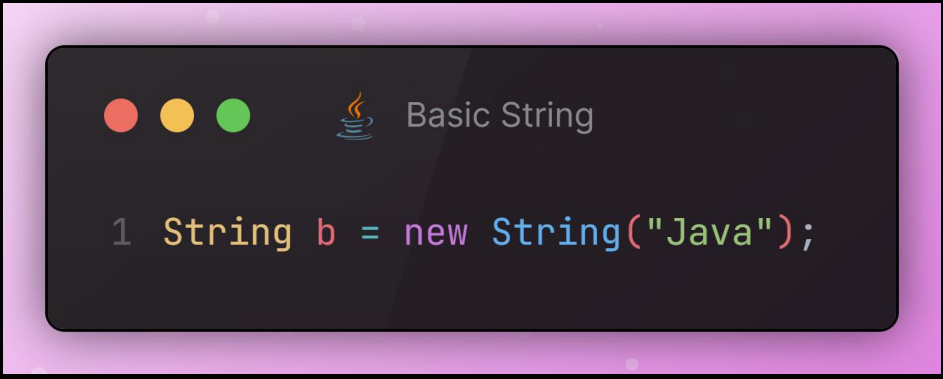
👉 Stored in the String Pool (special memory region).

2️⃣ String Creation: Two Ways

💡 Literal way (stored in string pool):



💡 Using new keyword (stored in heap):



🔍 *String pool helps save memory and improve performance.*

3️⃣ Why are Strings Immutable?

👉**Security** – Prevents unintended modifications (e.g., database connections, file paths).

👉**Thread Safety** – Can be shared across threads without synchronization.

👉**Thread Performance** – Enables String Pool optimization.

4️⃣ String Methods You MUST Know!

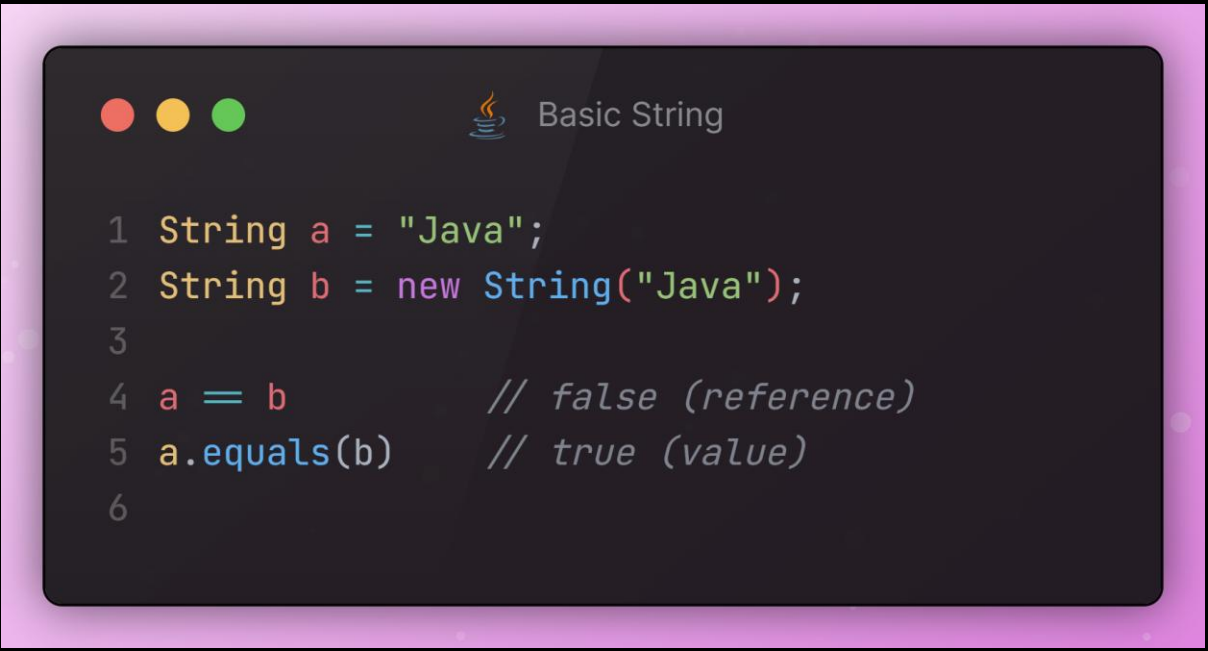
Basic String

```java
public class StringOperations {
    public static void main(String[] args) {

        // checks if a string is empty
        System.out.println("".isEmpty()); // true
        // returns the length
        System.out.println("Java".length()); // 4
        // returns the character based on index
        System.out.println("Java".charAt(0)); // J
        // starts at the first index and prints the rest
        System.out.println("Java".substring(1)); // ava
        // starts at the first index and excludes the last index
        System.out.println("Java".substring(1, 3)); // av
        // Join the two or more strings
        System.out.println("Java".concat(" Program")); // Java Program
        // Compares the two strings strict comparision
        System.out.println("Java".equals("java")); // false
        // Compares the two strings loose comparision
        System.out.println("Java".equalsIgnoreCase("java")); // True
        // Converts the string to lower case
        System.out.println("Java".toLowerCase()); // java
        // Converts the string to UPPER CASE
        System.out.println("Java".toUpperCase()); // JAVA
        // Trims out the spaces around the string
        System.out.println("   Java   ".trim()); // Java
        // Replaces every occurence of old with new character
        System.out.println("Java".replace('a', 'o')); // Jovo
        // Splits the java string into array of substrings
        String[] myArray = "Java".split("");
        for (String s : myArray) {
            System.out.println(s); // J a v a
        }
    }
}

```

👉 *Practice these! They're used in real interviews.*

5 String Comparison

== vs .equals()



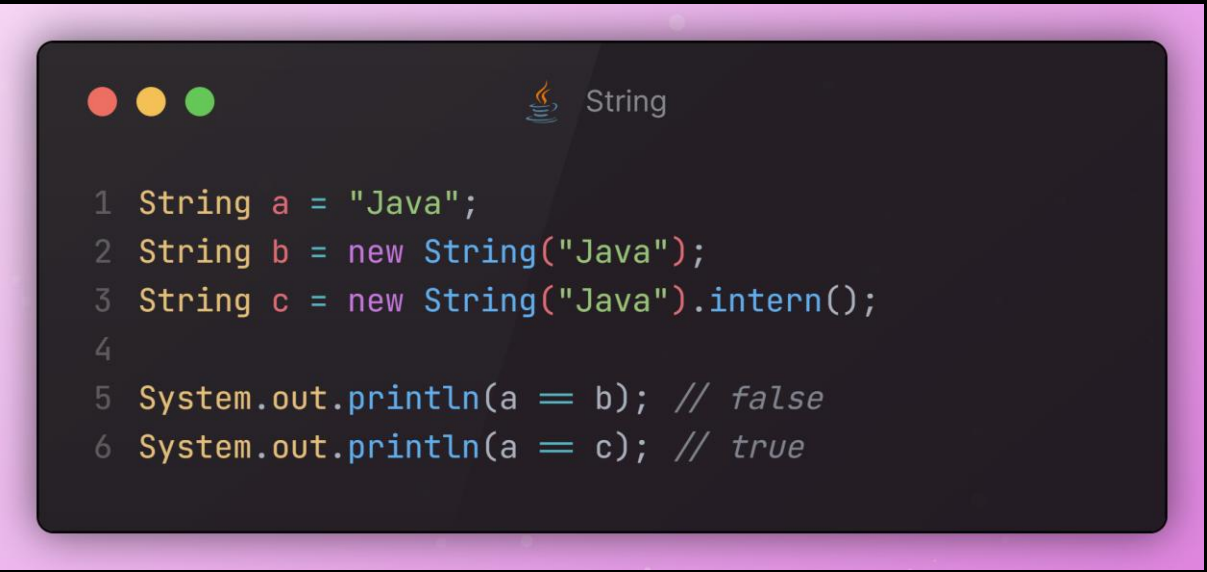
6 String vs StringBuilder vs StringBuffer

Feature	String	StringBuilder	StringBuffer
Mutability	Immutable	Mutable	Mutable
Thread Safety	Yes(Immutable)	No	Yes(Synchronized)
Performance	Slow for modifications	Fast	Slightly slower than StringBuilder
When to use	Fixed Values	Single-threaded Modifications	Multi-threaded Modifications

String

```
// StringBuilder (Faster, not thread-safe)
StringBuilder sb = new StringBuilder("Hello");
sb.append(" World");
System.out.println(sb); // "Hello World"

// StringBuffer (Thread-safe)
StringBuffer sBuffer = new StringBuffer("Hello");
sBuffer.append(" World");
System.out.println(sBuffer); // "Hello World"
```

7 String Interning



✨ *intern() ensures string is added to pool.*

🚀 Best Practices

✅ Use .equals() for string comparison

✅ Prefer StringBuilder for string manipulation in loops

✅ Avoid using == unless comparing references

✅ Use .intern() only when needed

✅ Beware of memory leaks from unused interned strings

🥳 And that’s a Wrap-Up

Let me know what Java topic you'd like next 👇