# String in Java

**1. What is a String in Java?**

* A **String** is a **sequence of characters**.
* In Java, String is a **class** in java.lang package.
* Strings are **immutable** → once created, they cannot be changed.
* Java provides **String, StringBuilder, and StringBuffer** to handle text.

Example:

String name = "Prashanth"; // String literal

String city = new String("Hyderabad"); // Using new keyword

**2. When are Strings used?**

Strings are one of the **most used classes** in Java.  
You use them when dealing with:

* **User input/output** → names, addresses, messages
* **File handling** → reading/writing text data
* **Database queries** → SQL statements are strings
* **Web applications** → URLs, JSON/XML, API calls
* **Enterprise applications** → logging, formatting, reports

**3. Where are Strings stored?**

Java maintains a **String Constant Pool (SCP)** in **heap memory**.

* If you create a string using **literal** → JVM checks SCP.
  + If it already exists → same reference is reused.
  + Else → new object is created in SCP.
* If you create a string using **new keyword** → new object in **heap memory**, not in SCP.

Example:

String s1 = "Hello";  // goes to SCP

String s2 = "Hello";  // reuses same reference

String s3 = new String("Hello"); // new object in heap

**4. How are Strings immutable?**

* Once created, the content of a String **cannot be modified**.
* Any modification creates a **new object**.

👉 Example:

String str = "Java";

str.concat(" World");

System.out.println(str); // Output: Java (not changed)

String newStr = str.concat(" World");

System.out.println(newStr); // Output: Java World

**5. Why Strings are immutable?**

* **Security**: For passwords, URLs, etc.
* **Caching**: SCP reuses strings → saves memory.
* **Thread-safety**: Safe to share between multiple threads.
* **Hashcode immutability**: Useful in HashMap keys.

**6. Important String Methods**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Method | Description | Example | | length() | Returns length | "Prashanth".length() → 9 | | charAt(int index) | Returns character | "Java".charAt(2) → v | | substring(begin, end) | Extract part of string | "Hello".substring(1,4) → "ell" | | equals() | Compares content | "abc".equals("abc") → true | | equalsIgnoreCase() | Ignores case | "ABC".equalsIgnoreCase("abc") → true | | == | Compares reference | "abc" == "abc" → true | | toUpperCase() | Convert to uppercase | "java".toUpperCase() → "JAVA" | | toLowerCase() | Convert to lowercase | "JAVA".toLowerCase() → "java" | | trim() | Removes leading/trailing spaces | " hi ".trim() → "hi" | | split(" ") | Splits string into array | "a,b,c".split(",") → ["a","b","c"] | | replace("a","x") | Replace characters | "java".replace("a","x") → "jxvx" | |

**7. String Comparison**

👉 Example:

String s1 = "Hello";

String s2 = "Hello";

String s3 = new String("Hello");

System.out.println(s1 == s2);   // true (SCP same reference)

System.out.println(s1 == s3);   // false (different object)

System.out.println(s1.equals(s3)); // true (same content)

**8. Code Example:**

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or

// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.

public class StringExample {

    public static void main(String[] args) {

        // Declaring Strings using String literals

        String s1 = "Astinil Technologies";

        String s2 = "DAKR Solutions";

        // Declaring Strings using new keyword

        String s3 = new String("Startahb Technologies");

        String s4 = new String("Astinil Tecnologies");

        // Printing all the Strings

        System.out.println(s1);

        System.out.println(s2);

        System.out.println(s3);

        System.out.println(s4);

        // Creating Byte ASCII Array

        byte ascii[] = { 71, 70, 71 };

        // Creating String using byte array

        String firstString = new String(ascii);

        System.out.println(firstString);

        // Creating String using byte array with  Start index to End Index

        String secondString = new String(ascii, 1, 2);

        System.out.println(secondString);

        // Character Array

        char characters[] = { 'G', 'f', 'g' };

        // Creating new String using Character Array

        String firstString1 = new String(characters);

        // Creating new String using another String

        String secondString1 = new String(firstString1);

        System.out.println("first string-------"+firstString1);

        System.out.println("second string-------"+secondString1);

    }

}

**Output:**

Astinil Technologies

DAKR Solutions

Startahb Technologies

Astinil Tecnologies

GFG

FG

first string-------Gfg

second string-------Gfg

## 9. ****String in Real-Time Scenarios****

### 🏦 Banking Application

* **Login Authentication**: Compare entered password with stored password.
* **Transaction Details**: Store remarks → "Transfer to Rajahmundry Branch".
* **Statement Generation**: "Dear Prashanth, Your balance is Rs. 5000.".
* **Database Queries**: "SELECT \* FROM accounts WHERE name='Prashanth'".

## 10. ****Block Diagram – String Working****

Here’s a conceptual block diagram:

String Object

char[] value (characters) int hash

Stored in SCP (if literal)

Stored in Heap

**11. Advantages of String in Java**

✔ Easy to use (lots of built-in methods)  
✔ Immutable → secure, reliable  
✔ Supported by core Java libraries  
✔ Used everywhere (I/O, APIs, DB, etc.)

**Summary**

* String is a **class** in Java.
* It is **immutable**, stored in **Heap + SCP**.
* Frequently used in **real-world apps** (Banking, APIs, Web apps).
* Provides **rich methods** for manipulation.