**Java 11 – String API Enhancements**

**🔹 Overview**

In **Java 11**, the String class got new **utility methods** to handle **whitespace, text cleaning, line processing, and repetition**.  
These methods make string handling easier, cleaner, and more powerful compared to older Java versions.

## New Methods in Java 11

### 1. isBlank()

* **Definition:** Checks if a string is empty ("") or contains only whitespace (spaces, tabs, newlines).
* **Syntax:**

boolean result = str.isBlank();

* **Use case:** Validating user input fields.

### 2. strip(), stripLeading(), stripTrailing()

* **Definition:** Remove whitespace (Unicode-aware).
  + strip() → removes whitespace from both ends.
  + stripLeading() → removes only at the start.
  + stripTrailing() → removes only at the end.
* **Syntax:**
* String result = str.strip();
* String result2 = str.stripLeading();

String result3 = str.stripTrailing();

* **Use case:** Cleaning input before saving or comparing.

### 3. lines()

* **Definition:** Splits a string into lines and returns a **Stream<String>**.
* **Syntax:**

str.lines().forEach(System.out::println);

* **Use case:** Reading and processing multi-line strings.

### 4. repeat(int count)

* **Definition:** Repeats the string given number of times.
* **Syntax:**

String result = str.repeat(count);

* **Use case:** Creating separators, test data, or repeated patterns.

**🔹 Difference Table**

| **Method** | **Old Way** | **New Way (Java 11)** |
| --- | --- | --- |
| Check empty/whitespace | str.trim().isEmpty() | str.isBlank() |
| Trim spaces | str.trim() | str.strip() (Unicode-aware) |
| Split lines | str.split("\n") | str.lines() |
| Repeat string | Loop + StringBuilder | str.repeat(n) |

## Code examples

**package** StringAPI;

// Java 11 examples — compile/run with JDK 11+

**import** java.util.stream.Collectors;

**public** **class** Java11StringExamples {

**public** **static** **void** main(String[] args) {

// 1) isBlank()

String a = " \n\t";

System.***out***.println("isBlank: " + a.isBlank()); // true

System.***out***.println("isEmpty: " + a.isEmpty()); // false

// 2) strip() vs trim()

String name = "\u2003Alice\u2002"; // contains Unicode spaces

System.***out***.println("trim: '" + name.trim() + "'");

System.***out***.println("strip: '" + name.strip() + "'");

// 3) lines()

String multi = "Line1\nLine2\r\nLine3\n";

multi.lines()

.map(String::strip) // clean whitespace per line

.filter(s -> !s.isBlank()) // skip blank lines

.forEach(System.***out***::println);

// 4) repeat()

String dash = "-".repeat(10);

System.***out***.println(dash); // "----------"

// Combined small pipeline

String userInput = " Bob \n \n Carol ";

**var** cleanedNames = userInput.lines()

.map(String::strip)

.filter(s -> !s.isBlank())

.collect(Collectors.*toList*());

System.***out***.println(cleanedNames); // [Bob, Carol]

}

}

## Output:

isBlank: true

isEmpty: false

' Alice trim: '

strip: 'Alice'

Line1

Line2

Line3

----------

[Bob, Carol]

## Real-Life Uses

* **Form validation:** if(name.isBlank()) { showError(); }
* **Data cleaning:** cleanName = name.strip();
* **Log processing:** log.lines().filter(...).forEach(...);
* **UI printing:** "=".repeat(40) to create separators.