Java String Methods (Functions): Full List With Examples

55 mins read Last updated: 06 Oct 2025 1213 views

Table of Contents

Introduction

Working with strings is a fundamental part of Java programming, and learning the right methods can make your code more efficient and readable. Java provides a wide range of built-in methods to manipulate, compare, search, and modify strings.

These are essential tools for real-world applications. In this guide, you'll learn commonly used Java string methods along with syntax and practical code examples. From simple length checks to advanced transformations, we'll cover it all.

If you're looking for easy-to-understand string methods in Java with examples, this post will serve as a complete reference.

What Are String Methods in Java?

String methods in Java are built-in functions provided by the String class that help perform operations on string data. Since strings in Java are immutable (they cannot be changed once created), these methods return a new string or result without modifying the original one.

You can use them to perform tasks like checking length, extracting a part of the string, replacing characters, comparing values, converting case, and more. These methods simplify common string-related tasks and reduce the need for complex logic.

Java String Methods (Functions): List

Method	Description	Return Type
charAt(int index)	Returns the character at the specified index	char
codePointAt(int index)	Returns Unicode code point at specified index	int
codePointBefore(int index)	Returns Unicode code point before specified index	int
codePointCount(int beginIndex, int endIndex)	Returns the number of Unicode code points in the specified range	int
compareTo(String anotherString)	Compares two strings lexicographically	int
compareToIgnoreCase(String str)	Compares two strings lexicographically, ignoring case	int
concat(String str)	Concatenates the specified string to the end	String
contains(CharSequence s)	Checks if the string contains the specified sequence	boolean
contentEquals(CharSequence cs)	Compares string content with CharSequence	boolean
contentEquals(StringBuffer sb)	Compares string content with StringBuffer	boolean

endsWith(String suffix)	Checks if the string ends with the given suffix	boolean
equals(Object anObject)	Checks if two strings are equal	boolean
equalsIgnoreCase(String anotherString)	Compares strings ignoring case	boolean
format(String format, Object... args)	Returns formatted string using specified format and arguments	String
getBytes()	Encodes string into byte array	byte[]
getBytes(String charsetName)	Encodes string using given charset	byte[]
getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)	Copies characters into a char array	void
hashCode()	Returns the hash code for the string	int
indexOf(int ch)	Returns the index of first occurrence of character	int
indexOf(int ch, int fromIndex)	Returns the index of character from specific index	int
indexOf(String str)	Returns index of first occurrence of substring	int
indexOf(String str, int fromIndex)	Returns index of substring from specific index	int
intern()	Returns canonical representation of	String

	is empty	
isBlank() (Java 11+)	Checks if the string is empty or contains only whitespace	boolean
join(CharSequence delimiter, CharSequence... elements)	Joins elements with a delimiter	String
lastIndexOf(int ch)	Returns last index of character	int
lastIndexOf(int ch, int fromIndex)	Returns last index from specific index	int
lastIndexOf(String str)	Returns last index of substring	int
lastIndexOf(String str, int fromIndex)	Returns last index of substring from index	int
length()	Returns the length of the string	int
lines() (Java 11+)	Returns a stream of lines from the string	Stream<String>
matches(String regex)	Tells if string matches the regex	boolean
offsetByCodePoints(int index, int codePointOffset)	Returns index after moving offset code points	int
regionMatches(boolean ignoreCase, int toffset, String other, int ooffset, int len)	Tests if substring matches another region	boolean
repeat(int count) (Java 11+)	Returns string repeated count times	String
replace(char oldChar, char newChar)	Replaces characters in string	String
replace(CharSequence target, CharSequence replacement)	Replaces target subsequence	String

replaceFirst(String regex, String replacement)	Replaces first match of regex	String
split(String regex)	Splits string by regex	String[]
split(String regex, int limit)	Splits with limit on number of substrings	String[]
startsWith(String prefix)	Checks if string starts with prefix	boolean
startsWith(String prefix, int toffset)	Checks if substring starts with prefix	boolean
strip() (Java 11+)	Removes leading and trailing whitespaces	String
stripLeading() (Java 11+)	Removes only leading whitespaces	String
stripTrailing() (Java 11+)	Removes only trailing whitespaces	String
subSequence(int beginIndex, int endIndex)	Returns a character sequence from string	CharSequence
substring(int beginIndex)	Returns substring from index to end	String
substring(int beginIndex, int endIndex)	Returns substring between two indexes	String
toCharArray()	Converts string into character array	char[]
toLowerCase()	Converts all characters to lowercase	String
toLowerCase(Locale locale)	Converts to lowercase using specific locale	String

toUpperCase(Locale locale)	Converts to uppercase using specific locale	String
toString()	Returns the string itself	String
trim()	Removes leading and trailing whitespaces	String
valueOf(boolean b)	Returns string representation of boolean	String
valueOf(char c)	Returns string of character	String
valueOf(char[] data)	Converts char array to string	String
valueOf(int i) (and other primitives)	Converts int, long, double, etc., to string	String

Note:

- Some methods like isBlank(), lines(), repeat(), strip() are available from Java 11 onwards.
- valueOf() is overloaded to work with various data types.

Most Common String Methods in Java

Let's discuss the most commonly used string methods in Java:

1. length()

This string method in Java returns the total number of characters present in a string, including spaces and special characters. It is one of the most important string methods in Java and is widely used in input validation, loops, and string size checks.

Syntax:

Example:

```
public class Example1 {
    public static void main(String[] args) {
        String name = "WsCube Tech";
        System.out.println("Length: " + name.length());
    }
}
```

Run Code

Output:

```
Length: 12
```

2. charAt(int index)

This string function in Java returns the character located at the specified index (starting from 0). It is helpful when analyzing characters in a string individually, such as in string parsing or pattern matching.

Syntax:

```
string.charAt(index);
```

Example:

```
public class Example2 {
    public static void main(String[] args) {
        String word = "Java";
        System.out.println("Character at index 2: " + word.charA
    }
}
```

3. substring(int beginIndex, int endIndex)

This important string method in Java extracts a part of the string starting from beginIndex up to, but not including, endIndex. It's used for trimming strings, extracting specific data, or removing prefixes/suffixes.

Syntax:

```
string.substring(beginIndex, endIndex);
```

Example:

```
public class Example3 {
    public static void main(String[] args) {
        String name = "Ravi Kumar";
        System.out.println(name.substring(0, 4));
    }
}
```

Run Code

Output:

```
Ravi
```

4. equals(String anotherString)

This java string method checks whether two strings have the same sequence of characters. It is case-sensitive and returns a boolean value (true or false).

Syntax:

Example:

```
public class Example4 {
    public static void main(String[] args) {
        String a = "Hello";
        String b = "hello";
        System.out.println(a.equals(b));
    }
}
```

Run Code

Output:

```
false
```

5. toUpperCase()

This string function of Java converts all characters in a string to uppercase. It's especially useful for standardizing text for case-insensitive comparisons or formatting purposes.

Syntax:

```
string.toUpperCase();
```

Example:

```
public class Example5 {
    public static void main(String[] args) {
        String city = "delhi";
        System.out.println(city.toUpperCase());
    }
}
```

```
DELHI
```

6. trim()

This java string method removes all leading and trailing whitespaces from a string. It's widely used for cleaning input data before storage or comparison.

Syntax:

```
string.trim();
```

Example:

```
public class Example6 {
    public static void main(String[] args) {
        String msg = "  Hello Java  ";
        System.out.println("Before: [" + msg + "]");
        System.out.println("After: [" + msg.trim() + "]");
    }
}
```

Run Code

Output:

```
Before: [  Hello Java  ]
After: [Hello Java]
```

7. replace(char oldChar, char newChar)

This string function of Java returns a new string by replacing all occurrences of oldChar with newChar. It's commonly used in formatting and cleaning operations.

Example:

```
public class Example7 {
    public static void main(String[] args) {
        String text = "banana";
        System.out.println(text.replace('a', 'o'));
    }
}
```

Run Code

Output:

```
bonono
```

8. split(String regex)

This string method in Java splits the string around matches of the given regular expression and returns an array of substrings. It's used in parsing, especially when dealing with CSV, logs, or user inputs.

Syntax:

```
string.split(regex);
```

Example:

```
public class Example8 {
    public static void main(String[] args) {
        String data = "apple,banana,grape";
        String[] fruits = data.split(",");
        for (String fruit : fruits) {
            System.out.println(fruit);
        }
```

Run Code

Output:

```
apple
banana
grape
```

9. contains(CharSequence s)

This java string method checks if the specified character sequence exists within the string. It returns true or false. Commonly used in search or filtering operations.

Syntax:

```
string.contains(sequence);
```

Example:

```
public class Example9 {
    public static void main(String[] args) {
        String message = "Welcome to Java World";
        System.out.println(message.contains("Java"));
    }
}
```

Run Code

Output:

```
true
```

10. equalsIgnoreCase(String anotherString)

```
string.equalsIgnoreCase(anotherString);
```

Example:

```
public class Example10 {
    public static void main(String[] args) {
        String a = "Admin";
        String b = "admin";
        System.out.println(a.equalsIgnoreCase(b));
    }
}
```

Run Code

Output:

```
true
```

11. indexOf(String str)

This important string method in Java returns the index of the first occurrence of the specified substring. If the substring is not found, it returns -1.

Syntax:

```
string.indexOf(substring);
```

Example:

```
public class Example11 {
    public static void main(String[] args) {
```

```
}
```

Run Code

Output:

```
7
```

12. startsWith(String prefix)

This string method in Java checks if the string starts with the specified prefix. It's often used in filtering data like filenames, URLs, etc.

Syntax

```
string.startsWith(prefix);
```

Example:

```
public class Example12 {
    public static void main(String[] args) {
        String filename = "report2025.pdf";
        System.out.println(filename.startsWith("report"));
    }
}
```

Run Code

Output:

```
true
```

13. endsWith(String suffix)

```
string.endsWith(suffix);
```

Example:

```
public class Example13 {
    public static void main(String[] args) {
        String url = "example.com";
        System.out.println(url.endsWith(".com"));
    }
}
```

Run Code

Output:

```
true
```

14. replaceAll(String regex, String replacement)

This java string method replaces each substring that matches the given regex with the specified replacement. It's widely used in sanitization, formatting, and data cleaning.

Syntax:

```
string.replaceAll(regex, replacement);
```

Example:

```
public class Example14 {
    public static void main(String[] args) {
        String messy = "abc123xyz456";
        System.out.println(messy.replaceAll("[0-9]", ""));
```

Run Code

Output:

```
abcxyz
```

15. isEmpty()

This important string method in Java checks if the string has a length of 0. It's frequently used in form validation and input checks.

Syntax:

```
string.isEmpty();
```

Example:

```
public class Example15 {
    public static void main(String[] args) {
        String input = "";
        System.out.println(input.isEmpty());
    }
}
```

Run Code

Output:

```
true
```

Java String Methods Example

Below is an example program that combines multiple Java string methods:

```
        String fullName =     Virendra Soni    ;
        String email = "info@wscubetech.com";

        // Step 1: Trim unwanted spaces
        fullName = fullName.trim();

        // Step 2: Capitalize first letter of each word
        String[] words = fullName.split(" ");
        StringBuilder formattedName = new StringBuilder();
        for (String word : words) {
            if (!word.isEmpty()) {
                formattedName.append(Character.toUpperCase(word.
                                .append(word.substring(1).toLowerCa
                                .append(" ");
            }
        }

        // Step 3: Validate email domain
        boolean isWsCubeEmail = email.endsWith("@wscubetech.com'

        // Step 4: Final output
        System.out.println("Formatted Name: " + formattedName.to
        System.out.println("Is official email? " + isWsCubeEmail
    }
}
```

Run Code

Output:

```
Formatted Name: Virendra Soni
Is official email? true
```

Explanation:

This program first trims the extra spaces from the user's name using trim(), then uses split() and charAt() along with substring() and toUpperCase() to format the name properly. After that, it checks if the email ends with a specific domain using endsWith().

1. Strings in Java are Immutable

Whenever you call a string method like replace() or toUpperCase(), it doesn't modify the original string. Instead, it returns a new string object. Always assign the result to a new variable (or back to the same one) if you want to use the modified value.

2. Use .equals() for String Comparison, Not ==

The == operator checks reference equality (whether both strings point to the same memory location), not value equality. To compare string contents, always use .equals() or .equalsIgnoreCase().

3. Watch Out for NullPointerException

Calling a method like str.length() or str.equals("value") on a null string will throw a NullPointerException. To avoid this, use "value".equals(str) instead — this is null-safe.

4. Regex in split(), replaceAll(), and matches()

Methods like split() and replaceAll() use regular expressions. Symbols like . and | have special meanings, so escape them if you're using them as literals (e.g., split("\\.") for splitting on a dot).

5. Prefer isBlank() Over isEmpty() (Java 11+)

isEmpty() returns true only if the string length is zero, while isBlank() also returns true for strings containing only whitespaces. For real-world form validations, isBlank() is often more accurate.

6. Use StringBuilder for Repeated Modifications

If you're performing multiple modifications (e.g., in a loop), avoid using + with strings. Instead, use StringBuilder or StringBuffer for better performance and memory efficiency.

7. Method Chaining Works but Can Get Messy

While chaining methods like str.trim().toLowerCase().replace(" ", "") is possible, too much chaining without clarity can reduce readability. Use meaningful intermediate variables if needed.

FAQs About Java String Methods

Use the method toUpperCase(). Example: "hello".toUpperCase() returns "HELLO".

How can I check if a string is empty?

What does the length() method do in Java?

How do I extract part of a string?

How do I split a string in Java?

How to compare strings ignoring case?

Can I chain multiple string methods together?

What's the difference between replace() and replaceAll()?

Is there a method to repeat a string multiple times?

Updated - 06 Oct 2025 55 mins read Published : 11 Aug 2025

WsCube Tech®
Upskilling Bharat

Elevate Your Learning Journey with Cutting-Edge Education Technology.

WsCube Tech®
Upskilling Bharat

About

WsCube Tech Blog

Self-Paced Courses

Masterclass

Digital Marketing

Web Development

Cyber Security

App Development

Support

Privacy Policy

Terms & Conditions

Refund Policy

Telegram Community

@wscubetech

Version 3.0.59