String Methods

Java provides a variety of **String** methods that allow you to manipulate and process text easily. Below is a list of commonly used **String** methods, along with their syntax and examples.

String Methods in Java

Method	Syntax	Description	Example
charAt(int index)	str.charAt(index)	Returns the character at the specified index.	String str = “Java”; char c = str.charAt(1); // ‘a’
length()	str.length()	Returns the number of characters in the string.	String str = “Hello”; int len = str.length(); // 5
toLowerCase()	str.toLowerCase()	Returns a new string with all characters in lowercase.	String str = “JAVA”; String lower = str.toLowerCase(); // “java”
toUpperCase()	str.toUpperCase()	Returns a new string with all characters in uppercase.	String str = “java”; String upper = str.toUpperCase(); // “JAVA”
substring(int start)	str.substring(start)	Returns a substring starting from the given index.	String str = “Hello”; String sub = str.substring(1); // “ello”
substring(int start, int end)	str.substring(start, end)	Returns a substring from start to end index (exclusive).	String str = “Hello”; String sub = str.substring(1, 4); // “ell”

Method	Syntax	Description	Example
contains(CharSequence seq)	str.contains(seq)	Checks if the string contains the specified sequence.	String str = “Java programming”; boolean result = str.contains(“program”); // true
equals(Object obj)	str.equals(obj)	Compares two strings for equality (case-sensitive).	String str1 = “hello”; String str2 = “hello”; boolean result = str1.equals(str2); // true
equalsIgnoreCase(String str)	str.equalsIgnoreCase(str)	Compares two strings for equality, ignoring case differences.	String str1 = “hello”; String str2 = “HELLO”; boolean result = str1.equalsIgnoreCase(str2); // true
indexOf(int ch)	str.indexOf(ch)	Returns the index of the first occurrence of the specified character.	String str = “Hello”; int index = str.indexOf(‘e’); // 1
lastIndexOf(int ch)	str.lastIndexOf(ch)	Returns the index of the last occurrence of the specified character.	String str = “Hello”; int index = str.lastIndexOf(‘l’); // 3
replace(CharSequence target, CharSequence replacement)	str.replace(target, replacement)	Replaces all occurrences of the target sequence with the replacement sequence.	String str = “hello”; String result = str.replace(“e”, “a”); // “hallo”

Method	Syntax	Description	Example
replaceAll(String regex, String replacement)	str.replaceAll(regex, replacement)	Replaces each substring that matches the regular expression with the given replacement.	String str = “ab123cd”; String result = str.replaceAll(“\\d”, “#”); // “ab###cd”
replaceFirst(String regex, String replacement)	str.replaceFirst(regex, replacement)	Replaces the first substring that matches the regular expression with the given replacement.	String str = “ab123cd”; String result = str.replaceFirst(“\\d”, “#”); // “ab#23cd”
trim()	str.trim()	Removes leading and trailing whitespace from the string.	String str = ” hello “; String result = str.trim(); // “hello”
split(String regex)	str.split(regex)	Splits the string into an array of substrings based on the given regular expression.	String str = “apple,banana,orange”; String[] result = str.split(“,”); // [“apple”, “banana”, “orange”]
startsWith(String prefix)	str.startsWith(prefix)	Checks if the string starts with the given prefix.	String str = “Java”; boolean result = str.startsWith(“J”); // true
endsWith(String suffix)	str.endsWith(suffix)	Checks if the string ends with the given suffix.	String str = “Java”; boolean result = str.endsWith(“a”); // true

Method	Syntax	Description	Example
concat(String str)	str.concat(str)	Concatenates the specified string to the end of the current string.	String str1 = “Hello”; String result = str1.concat(” World”); // “Hello World”
isEmpty()	str.isEmpty()	Checks if the string is empty (length 0).	String str = “”; boolean result = str.isEmpty(); // true
matches(String regex)	str.matches(regex)	Checks if the string matches the given regular expression.	String str = “abc123”; boolean result = str.matches(“\\w+”); // true
valueOf(boolean b)	String.valueOf(b)	Converts the boolean value to a string.	boolean b = true; String str = String.valueOf(b); // “true”
valueOf(char c)	String.valueOf(c)	Converts the character to a string.	char c = ‘A’; String str = String.valueOf(c); // “A”
valueOf(int i)	String.valueOf(i)	Converts the integer to a string.	int i = 123; String str = String.valueOf(i); // “123”
valueOf(long l)	String.valueOf(l)	Converts the long value to a string.	long l = 123L; String str = String.valueOf(l); // “123”
valueOf(float f)	String.valueOf(f)	Converts the float value to a string.	float f = 10.5f; String str = String.valueOf(f); // “10.5”
valueOf(double d)	String.valueOf(d)	Converts the double value to a string.	double d = 12.34; String str = String.valueOf(d); // “12.34”

Method	Syntax	Description	Example
toCharArray()	str.toCharArray()	Converts the string into a new character array.	String str = “Hello”; char[] arr = str.toCharArray(); // [‘H’, ‘e’, ‘l’, ‘l’, ‘o’]
intern()	str.intern()	Returns a canonical representation for the string.	String str = “java”; String internStr = str.intern();
regionMatches(boolean ignoreCase, int toffset, String other, int ooffset, int len)	str.regionMatches(ignoreCase, toffset, other, ooffset, len)	Compares two substrings for equality.	String str1 = “Hello”; String str2 = “hell”; boolean result = str1.regionMatches(true, 0, str2, 0, 4); // true
codePointAt(int index)	str.codePointAt(index)	Returns the Unicode code point value of the character at the specified index.	String str = “Hello”; int codePoint = str.codePointAt(0); // 72
codePointBefore(int index)	str.codePointBefore(index)	Returns the Unicode code point value of the character before the specified index.	String str = “Hello”; int codePoint = str.codePointBefore(1); // 72

Explore Topics

Introduction to Java
- Getting Started with Java
- History of Java
- Features of Java
- Java vs Other Languages
- Java Editions
- Java Setup
- JDK, JRE, JVM

Java Basics
- First Java Program
- Compiler, Interpreter, JIT Compiler
- Code to Execution
- Comments
- Variables
- Constants and Literals
- Keywords
- Data Types
- Typecasting