Important Java String Class Methods for DSA

Prepared by Grok

July 16, 2025

This document lists the most relevant methods of the Java `String` class for solving Data Structures and Algorithms (DSA) problems involving strings and character manipulation. Each method is accompanied by a brief description of its functionality.

Contents

1 Creation and Conversion

`static String valueOf(char c)` Converts a single character to a String.
Useful for building strings from individual characters.

`static String valueOf(char[] data)` Converts a character array to a String.
Common in problems requiring array-to-string conversion.

`static String valueOf(int i)` Converts an integer (or other types) to a String.
Used for numeric-to-string conversions in output formatting.

`char[] toCharArray()` Converts the string to a character array.
Essential for character-level manipulation in strings.

2 Character and Substring Access

`char charAt(int index)` Returns the character at the specified index.
Used for accessing specific characters in string traversal.

`String substring(int beginIndex)` Returns a substring from beginIndex to the end.
Useful for extracting parts of a string.

`String substring(int beginIndex, int endIndex)` Returns a substring from beginIndex to endIndex (exclusive).
Common in substring-related problems.

`int length()` Returns the length of the string.
Fundamental for iterating over strings.

3 Comparison and Searching

`boolean equals(Object obj)` Checks if the string equals another object (case-sensitive).
Used for exact string matching.

`boolean equalsIgnoreCase(String anotherString)` Checks if the string equals another string, ignoring case.
Useful for case-insensitive comparisons, e.g., in palindromes.

`int compareTo(String anotherString)` Compares two strings lexicographically (case-sensitive).
Used in sorting or ordering strings.

`int compareToIgnoreCase(String str)` Compares two strings lexicographically, ignoring case.
Useful for case-insensitive sorting.

`boolean startsWith(String prefix)` Checks if the string starts with the specified prefix.
Common in pattern matching.

`boolean endsWith(String suffix)` Checks if the string ends with the specified suffix.

Used in suffix-based validation.

`int indexOf(int ch)` Returns the index of the first occurrence of the specified character.

Key for searching characters in strings.

`int indexOf(String str)` Returns the index of the first occurrence of the specified substring.

Used in substring search problems.

`int lastIndexOf(int ch)` Returns the index of the last occurrence of the specified character.

Useful for reverse searching.

`int lastIndexOf(String str)` Returns the index of the last occurrence of the specified substring.

Used in complex string parsing.

`boolean contains(CharSequence s)` Checks if the string contains the specified sequence.

Simplifies substring existence checks.

4 Manipulation and Transformation

`String toLowerCase()` Converts all characters to lowercase.

Used in case normalization, e.g., for anagrams.

`String toUpperCase()` Converts all characters to uppercase.

Used in case normalization or formatting.

`String trim()` Removes leading and trailing whitespace.

Common for cleaning input strings.

`String replace(char oldChar, char newChar)` Replaces all occurrences of oldChar with newChar.

Useful for character substitution.

`String replace(CharSequence target, CharSequence replacement)` Replaces all occurrences of target with replacement.

Used for substring replacement.

`String replaceAll(String regex, String replacement)` Replaces all substrings matching the regex with replacement.

Powerful for pattern-based replacements.

`String[] split(String regex)` Splits the string into an array based on the regex delimiter.

Common in tokenization or parsing problems.

5 Validation

`boolean isEmpty()` Checks if the string is empty (length == 0).

Used for basic input validation.

`boolean isBlank()` Checks if the string is empty or contains only whitespace.

Useful for input sanitization (Java 11+).

`boolean matches(String regex)` Checks if the string matches the specified regular expression.

Used for pattern validation, e.g., email or phone formats.

6 Concatenation and Joining

`String concat(String str)` Concatenates the specified string to the end.

Used for string building (less common due to immutability).

`static String join(CharSequence delimiter, CharSequence... elements)` Joins multiple strings with the specified delimiter.

Useful for combining strings efficiently (Java 8+).

7 Notes

- The `String` class is immutable; methods like `toLowerCase()` and `replace()` return new strings.
- For performance-critical applications, use `StringBuilder` or `StringBuffer` for frequent modifications.
- Methods like `split()`, `replaceAll()`, and `matches()` support regular expressions, ideal for pattern-based problems.