Notes on Java String Methods with Examples

Prepared by ChatGPT

August 19, 2025

Introduction

In Java, the `String` class is one of the most commonly used classes. It provides a rich set of methods to perform different operations such as searching, comparison, modification, and manipulation of strings. Below are the most important `String` methods with examples and use cases.

1 Important String Methods

1.1 1. length()

Use: Returns the length (number of characters) of the string.

```
// Example
String str = "Hello World";
System.out.println(str.length()); // Output: 11
```

Use Case: Useful for validating user input length (e.g., password length check).

1.2 2. charAt(int index)

Use: Returns the character at a specific index (0-based).

```
String str = "Java";
System.out.println(str.charAt(2)); // Output: v
```

Use Case: Extracting specific characters, e.g., initials in a name.

1.3 3. substring(int beginIndex, int endIndex)

Use: Returns a substring from begin index (inclusive) to end index (exclusive).

```
String str = "Programming";
System.out.println(str.substring(0, 6)); // Output: Progra
```

Use Case: Extracting domain from email (e.g., `user@gmail.com` → `gmail.com`).

1.4 4. equals(Object another)

Use: Compares two strings for exact equality (case-sensitive).

```
String s1 = "Hello";
String s2 = "Hello";
System.out.println(s1.equals(s2)); // true
```

Use Case: Checking login username/password.

1.5 5. equalsIgnoreCase(String another)

Use: Compares two strings ignoring case.

```
String s1 = "Java";
String s2 = "java";
System.out.println(s1.equalsIgnoreCase(s2)); // true
```

Use Case: Case-insensitive search (e.g., entering country names).

1.6 6. compareTo(String another)

Use: Compares two strings lexicographically.

```
System.out.println("apple".compareTo("banana")); // Negative
System.out.println("grape".compareTo("grape"));  // 0
System.out.println("pear".compareTo("orange"));  // Positive
```

Use Case: Sorting strings alphabetically.

1.7 7. toUpperCase() and toLowerCase()

Use: Converts all characters to upper/lower case.

```
String str = "Java";
System.out.println(str.toUpperCase()); // JAVA
System.out.println(str.toLowerCase()); // java
```

Use Case: Normalizing text input (e.g., making search case-insensitive).

1.8 8. trim()

Use: Removes leading and trailing spaces.

```
String str = "   Hello Java   ";
System.out.println(str.trim()); // "Hello Java"
```

Use Case: Cleaning user input before saving to a database.

1.9 9. replace(char oldChar, char newChar)

Use: Replaces all occurrences of a character.

```
String str = "banana";
System.out.println(str.replace('a','o')); // bonono
```

Use Case: Replacing unwanted characters, e.g., changing file paths.

1.10 10. contains(CharSequence s)

Use: Checks if the string contains the specified sequence.

```
String str = "I love programming";
System.out.println(str.contains("love")); // true
```

Use Case: Searching for keywords inside text.

1.11 11. indexOf(String str)

Use: Returns index of first occurrence of substring, or -1 if not found.

```
String str = "programming";
System.out.println(str.indexOf("gram")); // 3
```

Use Case: Finding the position of a word in a document.

1.12 12. split(String regex)

Use: Splits the string based on a regular expression.

```
String str = "apple,banana,orange";
String[] fruits = str.split(",");
for(String f : fruits) {
    System.out.println(f);
}
```

Use Case: Splitting CSV values into tokens.

1.13 13. startsWith() and endsWith()

Use: Checks if string starts or ends with given prefix/suffix.

```
String str = "document.pdf";
System.out.println(str.endsWith(".pdf")); // true
```

Use Case: File type checking, URL validation.

1.14 14. isEmpty()

Use: Checks if a string is empty (length = 0).

```
String str = "";
System.out.println(str.isEmpty()); // true
```

Use Case: Validating empty input fields.

1.15 15. valueOf()

Use: Converts different data types to string.

```
int num = 100;
String str = String.valueOf(num);
System.out.println(str + 50); // 10050
```

Use Case: Converting numbers or objects into string form for display.

Conclusion

The `String` class in Java provides powerful methods for manipulation, searching, and validation. Mastering these methods is essential for handling text-based data in Java applications.

2 Extended String Methods

2.1 1. getBytes()

Use: Encodes this string into a sequence of bytes.

```
String str = "ABC";
byte[] bytes = str.getBytes();
for (byte b : bytes) {
    System.out.print(b + " ");
}
// Output: 65 66 67
```

Use Case: Useful when sending strings over networks in byte format.

2.2 2. toCharArray()

Use: Converts the string into a character array.

```
String str = "Java";
char[] arr = str.toCharArray();
for(char c : arr) System.out.println(c);
```

Use Case: Needed when performing character-level manipulation.

2.3 3. matches(String regex)

Use: Tests whether the string matches the given regular expression.

```
String email = "user@gmail.com";
System.out.println(email.matches(".*@gmail\\.com")); // true
```

Use Case: Validating emails, phone numbers, etc.

2.4 4. intern()

Use: Returns a canonical representation of the string (from String pool).

```
String s1 = new String("Hello");
String s2 = s1.intern();
System.out.println(s1 == s2); // false
```

Use Case: Memory optimization when storing many duplicate strings.

2.5 5. format()

Use: Returns a formatted string using placeholders.

```
String str = String.format("Name: %s, Age: %d", "Alice", 22);
System.out.println(str);
// Output: Name: Alice, Age: 22
```

Use Case: Dynamic string creation (e.g., reports, logs).

2.6 6. join(CharSequence delimiter, CharSequence... elements)

Use: Joins multiple strings with a delimiter.

```
String str = String.join("-", "2025", "08", "18");
System.out.println(str); // 2025-08-18
```

Use Case: Constructing CSV lines, dates, or file paths.

2.7 7. repeat(int count)

Use: Repeats the string multiple times (Java 11+).

```
String str = "Hi!";
System.out.println(str.repeat(3)); // Hi!Hi!Hi!
```

Use Case: Creating patterns, padding, or test strings.

2.8 8. strip()

Use: Removes leading and trailing Unicode white spaces (Java 11+).

```
String str = "  Hello  ";
System.out.println(str.strip()); // "Hello"
```

Use Case: More Unicode-aware alternative to `trim()`.

2.9 9. stripLeading() and stripTrailing()

Use: Removes spaces only from the beginning or end.

```
String str = "  Hello  ";
System.out.println(str.stripLeading());  // "Hello  "
System.out.println(str.stripTrailing()); // "  Hello"
```

Use Case: Cleaning user input where only one side needs trimming.

2.10 10. lines()

Use: Splits the string into a stream of lines (Java 11+).

```
String str = "A\nB\nC";
str.lines().forEach(System.out::println);
```

Use Case: Processing multi-line text input.

2.11 11. codePointAt(int index)

Use: Returns the Unicode code point at the given index.

```
String str = "A";
System.out.println(str.codePointAt(0)); // 65
```

Use Case: Useful when working with Unicode/emoji characters.

2.12 12. codePointBefore(int index)

Use: Returns Unicode code point before a given index.

```
String str = "AB";
System.out.println(str.codePointBefore(1)); // 65
```

2.13 13. codePointCount(int beginIndex, int endIndex)

Use: Returns number of Unicode code points in a substring.

```
String str = "Hello";
System.out.println(str.codePointCount(0, 5)); // 5
```

2.14 14. offsetByCodePoints(int index, int codePointOffset)

Use: Returns the index within the string offset by code points.

```
String str = "Hello";
System.out.println(str.offsetByCodePoints(0, 2)); // 2
```

Use Case: Useful in processing Unicode characters.

2.15 15. regionMatches()

Use: Tests if two substrings are equal.

```
String s1 = "HelloWorld";
String s2 = "World";
System.out.println(s1.regionMatches(5, s2, 0, 5)); // true
```

Use Case: Comparing parts of large strings.

2.16 16. replaceFirst(String regex, String replacement)

Use: Replaces the first substring matching the regex.

```
String str = "abc abc";
System.out.println(str.replaceFirst("abc", "xyz"));
// Output: xyz abc
```

2.17 17. replaceAll(String regex, String replacement)

Use: Replaces all substrings matching the regex.

```
String str = "abc abc";
System.out.println(str.replaceAll("abc", "xyz"));
// Output: xyz xyz
```

2.18 18. contentEquals()

Use: Compares the string to a `StringBuffer` or `CharSequence`.

```
String str = "Java";
StringBuffer sb = new StringBuffer("Java");
System.out.println(str.contentEquals(sb)); // true
```

2.19 19. concat()

Use: Concatenates the specified string to the end.

```
String s1 = "Hello";
String s2 = "World";
System.out.println(s1.concat(" " + s2)); // Hello World
```

2.20 20. valueOf(char[] data, int offset, int count)

Use: Converts part of a char array to string.

```
char[] arr = {'J','a','v','a'};
System.out.println(String.valueOf(arr, 0, 2)); // Ja
```

Conclusion

This extended list covers most other useful methods of the `String` class in Java. Mastering them allows you to handle text data in more advanced scenarios such as validation, formatting, Unicode processing, and text parsing.