# 101 String methods

Here is a list of commonly used 101 String methods in Java, along with examples for each:

**1. length()**

**Returns the length of the string.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello";
        System.out.println("Length: " + str.length()); // Output: 5
    }
}
```

**2. charAt()**

**Returns the character at the specified index.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello";
        System.out.println("Character at index 1: " + str.charAt(1)); // Output: e
    }
}
```

**3. substring()**

**Extracts a substring from the string.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello World";
        System.out.println("Substring: " + str.substring(6)); // Output: World
    }
}
```

**4. equals()**

Checks if two strings are equal.

```
public class Main {
    public static void main(String[] args) {
        String str1 = "Hello";
        String str2 = "Hello";
        System.out.println("Are strings equal? " + str1.equals(str2)); // Output: true
    }
}
```

**5. equalsIgnoreCase()**

**Compares two strings, ignoring case.**

```
public class Main {
    public static void main(String[] args) {
        String str1 = "HELLO";
        String str2 = "hello";
        System.out.println("Are strings equal ignoring case? " +
str1.equalsIgnoreCase(str2)); // Output: true
    }
}
```

**6. toUpperCase() and toLowerCase()**

**Converts the string to uppercase or lowercase.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello";
        System.out.println("Uppercase: " + str.toUpperCase()); // Output: HELLO
        System.out.println("Lowercase: " + str.toLowerCase()); // Output: hello
    }
}
```

**7. trim()**

**Removes leading and trailing whitespace.**

```
public class Main {
    public static void main(String[] args) {
        String str = "   Hello   ";
        System.out.println("Trimmed: '" + str.trim() + "'"); // Output: 'Hello'
    }
}
```

**8. replace()**

**Replaces occurrences of a character or substring.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello World";
        System.out.println("Replaced: " + str.replace("World", "Java")); // Output:
Hello Java
    }
}
```

**9. contains()**

**Checks if the string contains a specified sequence of characters.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello Java";
        System.out.println("Contains 'Java': " + str.contains("Java")); // Output: true
    }
}
```

**10. startsWith() and endsWith()**

**Checks if the string starts or ends with a specified substring.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello World";
        System.out.println("Starts with 'Hello': " + str.startsWith("Hello")); // Output:
true
        System.out.println("Ends with 'World': " + str.endsWith("World")); // Output:
true
    }
}
```

**11. indexOf()**

**Returns the index of the first occurrence of a character or substring.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello World";
        System.out.println("Index of 'o': " + str.indexOf('o')); // Output: 4
    }
}
```

**12. lastIndexOf()**

**Returns the index of the last occurrence of a character or substring.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello World";
        System.out.println("Last index of 'o': " + str.lastIndexOf('o')); // Output: 7
    }
}
```

**13. isEmpty()**

**Checks if the string is empty.**

```
public class Main {
    public static void main(String[] args) {
        String str = "";
        System.out.println("Is empty: " + str.isEmpty()); // Output: true
    }
```

```
}
```

**14. split()**

**Splits the string into an array based on a delimiter.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Java is fun";
        String[] words = str.split(" ");
        for(String word : words) {
            System.out.println(word); // Output: Java \n is \n fun
        }
    }
}
```

**15. concat()**

**Concatenates (joins) two strings.**

```
public class Main {
    public static void main(String[] args) {
        String str1 = "Hello";
        String str2 = "World";
        System.out.println("Concatenated: " + str1.concat(" ").concat(str2)); //
Output: Hello World
    }
}
```

**16. valueOf()**

**Converts different types to a string.**

```
public class Main {
    public static void main(String[] args) {
        int num = 100;
        System.out.println("String value: " + String.valueOf(num)); // Output: 100
    }
}
```

**17. compareTo()**

**Compares two strings lexicographically.**

```
public class Main {
    public static void main(String[] args) {
        String str1 = "Apple";
        String str2 = "Banana";
        System.out.println("Comparison result: " + str1.compareTo(str2)); // Output:
Negative number (because "Apple" < "Banana")
    }
}
```

**18. matches()**

**Tests if the string matches the given regular expression.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Java123";
        System.out.println("Matches regex: " + str.matches("[A-Za-z]+\\d+")); //
Output: true
    }
}
```

**19. toCharArray()**

**Converts the string into a character array.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello";
        char[] charArray = str.toCharArray();
        for(char ch : charArray) {
            System.out.println(ch); // Output: H \n e \n l \n l \n o
```

```
        }
    }
}
```

**20. intern()**

**Returns a canonical representation for the string. It ensures that all strings with the same content share the same memory.**

```
public class Main {
    public static void main(String[] args) {
        String str1 = new String("Hello").intern();
        String str2 = "Hello";
        System.out.println(str1 == str2); // Output: true (because they point to the same string in the pool)
    }
}
```

**21. regionMatches()**

**Compares a specific region of one string with a specific region of another string.**

```
public class Main {
    public static void main(String[] args) {
        String str1 = "HelloWorld";
        String str2 = "WorldHello";
        boolean result = str1.regionMatches(5, str2, 0, 5);
        System.out.println("Region matches: " + result); // Output: true
    }
}
```

**22. replaceFirst()**

**Replaces the first occurrence of a regex with a replacement string.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello World";
        System.out.println("Replace first 'o': " + str.replaceFirst("o", "O")); // Output: HellO World
    }
}
```

**23. replaceAll()**

**Replaces all occurrences of a regex with a replacement string.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello World";
        System.out.println("Replace all 'l': " + str.replaceAll("l", "L")); // Output: HeLLo WorLd
    }
}
```

**24. codePointAt()**

**Returns the Unicode code point of the character at the specified index.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello";
        System.out.println("Code point at index 1: " + str.codePointAt(1)); // Output: 101 (Unicode for 'e')
    }
}
```

**25. codePointBefore()**

**Returns the Unicode code point of the character before the specified index.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello";
```

```
        System.out.println("Code point before index 1: " + str.codePointBefore(1)); //
    Output: 72 (Unicode for 'H')
    }
}
```

**26. codePointCount()**

**Returns the number of Unicode code points in the specified text range.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello";
        System.out.println("Code point count: " + str.codePointCount(0, str.length()));
    // Output: 5
    }
}
```

**27. getBytes()**

**Encodes the string into a sequence of bytes using the specified charset.**

```
import java.nio.charset.StandardCharsets;

public class Main {
    public static void main(String[] args) {
        String str = "Hello";
        byte[] byteArray = str.getBytes(StandardCharsets.UTF_8);
        for(byte b : byteArray) {
            System.out.print(b + " "); // Output: 72 101 108 108 111
        }
    }
}
```

**28. join()**

**Joins multiple strings with a specified delimiter.**

```
public class Main {
    public static void main(String[] args) {
        String joined = String.join("-", "Java", "is", "fun");
        System.out.println("Joined string: " + joined); // Output: Java-is-fun
    }
}
```

**29. format()**

**Formats the string using the specified format and arguments.**

```
public class Main {
    public static void main(String[] args) {
        String str = String.format("Hello %s, you are %d years old.", "John", 25);
        System.out.println(str); // Output: Hello John, you are 25 years old.
    }
}
```

**30. compareToIgnoreCase()**

**Compares two strings lexicographically, ignoring case differences.**

```
public class Main {
    public static void main(String[] args) {
        String str1 = "apple";
        String str2 = "Apple";
        System.out.println(str1.compareToIgnoreCase(str2)); // Output: 0 (because
    they are the same, ignoring case)
    }
}
```

**31. subSequence()**

**Returns a new character sequence that is a subsequence of this string.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello World";
        System.out.println("Subsequence: " + str.subSequence(0, 5)); // Output: Hello
```

```
    }
}
```

**32. offsetByCodePoints()**

**Returns the index within the string that is offset from the given index by the specified number of code points.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello";
        System.out.println("Offset by code points: " + str.offsetByCodePoints(0, 2)); //
Output: 2
    }
}
```

**33. hashCode()**

**Returns the hash code of the string.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello";
        System.out.println("Hash code: " + str.hashCode()); // Output: unique hash
code for the string
    }
}
```

**34. getChars()**

**Copies characters from a string into a destination character array.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello World";
        char[] dest = new char[5];
        str.getChars(0, 5, dest, 0);
        System.out.println(dest); // Output: Hello
    }
}
```

**35. getBytes(Charset charset)**

**Encodes the string into a byte array using the specified charset.**

```
import java.nio.charset.StandardCharsets;

public class Main {
    public static void main(String[] args) {
        String str = "Hello World";
        byte[] byteArray = str.getBytes(StandardCharsets.UTF_8);
        for(byte b : byteArray) {
            System.out.print(b + " "); // Output: 72 101 108 108 111 32 87 111 114 108
100
        }
    }
}
```

**36. contentEquals()**

**Checks whether the string's content matches a CharSequence or StringBuffer.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello";
        StringBuffer sb = new StringBuffer("Hello");
        System.out.println("Content equals: " + str.contentEquals(sb)); // Output: true
    }
}
```

**37. compareTo()**

**Compares two strings lexicographically based on the Unicode values of each character.**

```
public class Main {
    public static void main(String[] args) {
        String str1 = "abc";
        String str2 = "xyz";
        System.out.println(str1.compareTo(str2)); // Output: negative number
because "abc" < "xyz"
    }
}
```

**38. split(String regex, int limit)**

**Splits the string around matches of the given regular expression, with a limit on the number of results.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Java is fun is interesting";
        String[] words = str.split(" ", 3); // limit to 3 parts
        for (String word : words) {
            System.out.println(word);
        }
        // Output:
        // Java
        // is
        // fun is interesting
    }
}
```

**39. copyValueOf()**

**Returns a string that represents the character array.**

```
public class Main {
    public static void main(String[] args) {
        char[] data = { 'H', 'e', 'l', 'l', 'o' };
        String str = String.copyValueOf(data);
        System.out.println(str); // Output: Hello
    }
}
```

**40. lines()**

**Returns a Stream<String> of lines from the string, separated by line terminators.**

```
import java.util.stream.Stream;

public class Main {
    public static void main(String[] args) {
        String str = "Hello\nWorld\nJava";
        Stream<String> lines = str.lines();
        lines.forEach(System.out::println);
        // Output:
        // Hello
        // World
        // Java
    }
}
```

**41. strip()**

**Removes leading and trailing spaces from the string (similar to trim() but with Unicode support).**

```
public class Main {
    public static void main(String[] args) {
        String str = "   Hello World   ";
        System.out.println("'" + str.strip() + "'"); // Output: 'Hello World'
    }
```

```
    }
```

**42. stripLeading()**

**Removes only the leading whitespace from the string.**

```
public class Main {
    public static void main(String[] args) {
        String str = "   Hello World";
        System.out.println("'" + str.stripLeading() + "'"); // Output: 'Hello World'
    }
}
```

**43. stripTrailing()**

**Removes only the trailing whitespace from the string.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello World   ";
        System.out.println("'" + str.stripTrailing() + "'"); // Output: 'Hello World'
    }
}
```

**44. repeat()**

**Repeats the string a given number of times.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Java ";
        System.out.println(str.repeat(3)); // Output: Java Java Java
    }
}
```

**45. indent()**

**Adds a specified number of spaces to the beginning of each line of the string.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello\nWorld";
        System.out.println(str.indent(4));
        // Output:
        //     Hello
        //     World
    }
}
```

**46. transform()**

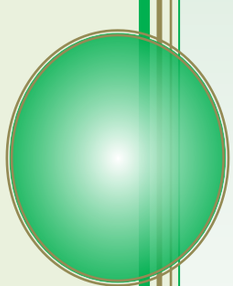
**Applies a transformation function to the string.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Java";
        String result = str.transform(s -> s.toUpperCase());
        System.out.println(result); // Output: JAVA
    }
}
```

**47. formatted()**

**Formats the string with the provided arguments (similar to String.format()).**

```
public class Main {
    public static void main(String[] args) {
        String template = "Hello, %s!";
        String result = template.formatted("World");
        System.out.println(result); // Output: Hello, World!
    }
}
```



**48. isBlank()**

**Checks if the string is empty or contains only whitespace.**

```
public class Main {
    public static void main(String[] args) {
        String str = "   ";
        System.out.println(str.isBlank()); // Output: true
    }
}
```

**49. concat() (Advanced usage with objects)**

**Concatenates multiple objects converted to strings.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello".concat(" ").concat(String.valueOf(123));
        System.out.println(str); // Output: Hello 123
    }
}
```

**50. join(CharSequence delimiter, CharSequence... elements)**

**Joins multiple CharSequence elements using the provided delimiter.**

```
public class Main {
    public static void main(String[] args) {
        String result = String.join(", ", "Java", "is", "fun");
        System.out.println(result); // Output: Java, is, fun
    }
}
```

**51. isUpperCase()**

**Java doesn't have a String.isUpperCase() method directly, but you can use Character.isUpperCase() for individual characters.**

```
public class Main {
    public static void main(String[] args) {
        char ch = 'A';
        System.out.println(Character.isUpperCase(ch)); // Output: true
    }
}
```

**52. isLowerCase()**

**Similarly, there's no String.isLowerCase(), but you can check individual characters using Character.isLowerCase().**

```
public class Main {
    public static void main(String[] args) {
        char ch = 'a';
        System.out.println(Character.isLowerCase(ch)); // Output: true
    }
}
```

**53. isLetter()**

**Checks if a given character is a letter.**

```
public class Main {
    public static void main(String[] args) {
        char ch = 'a';
        System.out.println(Character.isLetter(ch)); // Output: true
    }
}
```

**54. isDigit()**

**Checks if a character is a digit.**

```
public class Main {
    public static void main(String[] args) {
        char ch = '5';
        System.out.println(Character.isDigit(ch)); // Output: true
```

```
    }
}
```

**55. isWhitespace()**

**Checks if a character is a whitespace character.**

```
public class Main {
    public static void main(String[] args) {
        char ch = ' ';
        System.out.println(Character.isWhitespace(ch)); // Output: true
    }
}
```

**56. trim()**

**Removes leading and trailing whitespaces from the string.**

```
public class Main {
    public static void main(String[] args) {
        String str = "   Hello World   ";
        System.out.println("'" + str.trim() + "'"); // Output: 'Hello World'
    }
}
```

**57. valueOf()**

**Converts different types of values (int, double, char array, etc.) to their String representation.**

```
public class Main {
    public static void main(String[] args) {
        int num = 100;
        String str = String.valueOf(num);
        System.out.println(str); // Output: "100"

        char[] chars = {'H', 'e', 'l', 'l', 'o'};
        String str2 = String.valueOf(chars);
        System.out.println(str2); // Output: "Hello"
    }
}
```

**58. indexOf()**

**Returns the index of the first occurrence of the specified character or substring in the string.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello World";
        System.out.println(str.indexOf('o')); // Output: 4
        System.out.println(str.indexOf("World")); // Output: 6
    }
}
```

**59. lastIndexOf()**

**Returns the index of the last occurrence of the specified character or substring.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello World Hello";
        System.out.println(str.lastIndexOf('o')); // Output: 15
        System.out.println(str.lastIndexOf("Hello")); // Output: 12
    }
}
```

**60. startsWith()**

**Checks if the string starts with the specified prefix.**

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

```
        String str = "Hello World";
        System.out.println(str.startsWith("Hello")); // Output: true
    }
}
```

**61. endsWith()**

**Checks if the string ends with the specified suffix.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello World";
        System.out.println(str.endsWith("World")); // Output: true
    }
}
```

**62. toUpperCase()**

**Converts all the characters of the string to uppercase.**

```
public class Main {
    public static void main(String[] args) {
        String str = "hello world";
        System.out.println(str.toUpperCase()); // Output: HELLO WORLD
    }
}
```

**63. toLowerCase()**

**Converts all the characters of the string to lowercase.**

```
public class Main {
    public static void main(String[] args) {
        String str = "HELLO WORLD";
        System.out.println(str.toLowerCase()); // Output: hello world
    }
}
```

**64. codePointAt()**

**Returns the Unicode code point of the character at the specified index.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello";
        System.out.println(str.codePointAt(0)); // Output: 72 (Unicode for 'H')
    }
}
```

**65. codePointBefore()**

**Returns the Unicode code point before the specified index.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello";
        System.out.println(str.codePointBefore(1)); // Output: 72 (Unicode for 'H')
    }
}
```

**66. substring()**

**Extracts a substring from the string starting from the specified index, optionally ending at another index.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello World";
        System.out.println(str.substring(6)); // Output: "World"
        System.out.println(str.substring(0, 5)); // Output: "Hello"
    }
}
```

**67. contains()**

**Checks if the string contains the specified sequence of characters.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello World";
        System.out.println(str.contains("World")); // Output: true
    }
}
```

**68. split()**

**Splits a string into an array of substrings based on a regular expression (or a specific delimiter).**

```
public class Main {
    public static void main(String[] args) {
        String str = "apple,banana,orange";
        String[] fruits = str.split(",");
        for (String fruit : fruits) {
            System.out.println(fruit);
        }
        // Output:
        // apple
        // banana
        // orange
    }
}
```

**69. replace()**

**Replaces all occurrences of a character or substring with another character or substring.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello World";
        String result = str.replace("World", "Java");
        System.out.println(result); // Output: Hello Java
    }
}
```

**70. replaceAll()**

**Replaces all substrings that match a given regular expression.**

```
public class Main {
    public static void main(String[] args) {
        String str = "123abc456";
        String result = str.replaceAll("\\d", "X"); // Replaces digits with 'X'
        System.out.println(result); // Output: XXXabcXXX
    }
}
```

**71. replaceFirst()**

**Replaces the first occurrence of a substring that matches a given regular expression.**

```
public class Main {
    public static void main(String[] args) {
        String str = "123abc456";
        String result = str.replaceFirst("\\d", "X"); // Replaces the first digit with 'X'
        System.out.println(result); // Output: X23abc456
    }
}
```

**72. intern()**

**Returns a canonical representation of the string. If the string is already in the string pool, it returns that instance. Otherwise, it adds the string to the pool and returns the reference.**

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

```
        String str1 = new String("Hello").intern();
        String str2 = "Hello";
        System.out.println(str1 == str2); // Output: true (because both refer to the
same instance)
    }
}
```

**73. charAt()**

**Returns the character at the specified index.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello";
        char ch = str.charAt(1); // 'e'
        System.out.println(ch); // Output: e
    }
}
```

**74. getChars()**

**Copies characters from a string to a character array.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello World";
        char[] arr = new char[5];
        str.getChars(0, 5, arr, 0);
        System.out.println(arr); // Output: Hello
    }
}
```

**75. concat()**

**Concatenates the specified string to the end of the current string.**

```
public class Main {
    public static void main(String[] args) {
        String str1 = "Hello";
        String str2 = " World";
        String result = str1.concat(str2);
        System.out.println(result); // Output: Hello World
    }
}
```

**76. regionMatches()**

**Tests if two string regions are equal.**

```
public class Main {
    public static void main(String[] args) {
        String str1 = "Hello World";
        String str2 = "Hello Java";
        boolean match = str1.regionMatches(0, str2, 0, 5);
        System.out.println(match); // Output: true (because "Hello" matches in
both strings)
    }
}
```

**77. matches()**

**Checks if the string matches a given regular expression.**

```
public class Main {
    public static void main(String[] args) {
        String str = "abc123";
        boolean isMatch = str.matches("[a-z]+\\d+");
        System.out.println(isMatch); // Output: true (matches the pattern)
    }
}
```

**78. toCharArray()**

**Converts the string into a character array.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello";
        char[] chars = str.toCharArray();
        for (char c : chars) {
            System.out.print(c + " ");
        }
        // Output: H e l l o
    }
}
```

**79. join()**

**Joins an array of strings using a delimiter.**

```
import java.util.StringJoiner;
public class Main {
    public static void main(String[] args) {
        String[] words = {"apple", "banana", "orange"};
        String result = String.join(", ", words);
        System.out.println(result); // Output: apple, banana, orange
    }
}
```

**80. format()**

**Returns a formatted string using the specified format string and arguments, similar to printf.**

```
public class Main {
    public static void main(String[] args) {
        String name = "John";
        int age = 25;
        String result = String.format("My name is %s and I am %d years old.", name,
age);
        System.out.println(result); // Output: My name is John and I am 25 years
old.
    }
}
```

**81. compareTo()**

**Compares two strings lexicographically.**

```
public class Main {
    public static void main(String[] args) {
        String str1 = "apple";
        String str2 = "banana";
        int result = str1.compareTo(str2);
        System.out.println(result); // Output: -1 (because "apple" is
lexicographically less than "banana")
    }
}
```

**82. compareToIgnoreCase()**

**Compares two strings lexicographically, ignoring case differences.**

```
public class Main {
    public static void main(String[] args) {
        String str1 = "apple";
        String str2 = "Apple";
        int result = str1.compareToIgnoreCase(str2);
        System.out.println(result); // Output: 0 (because "apple" and "Apple" are
considered equal, ignoring case)
    }
}
```

**83. contentEquals()**

**Checks if the string content matches a StringBuffer or CharSequence.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello";
        StringBuffer buffer = new StringBuffer("Hello");
        boolean result = str.contentEquals(buffer);
        System.out.println(result); // Output: true
    }
}
```

**84. substring()**

**Returns a substring from a given string, starting from the specified index, optionally up to an ending index.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello World";
        String result = str.substring(6); // From index 6 to the end
        System.out.println(result); // Output: World

        String result2 = str.substring(0, 5); // From index 0 to 5
        System.out.println(result2); // Output: Hello
    }
}
```

**85. hashCode()**

**Returns the hash code of the string, useful for storing strings in hash-based data structures like HashMap.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello";
        int hash = str.hashCode();
        System.out.println(hash); // Output: unique hash code for "Hello"
    }
}
```

86. startsWith()

Checks if the string starts with the specified prefix.

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello World";
        boolean result = str.startsWith("Hello");
        System.out.println(result); // Output: true
    }
}
```

**87. endsWith()**

**Checks if the string ends with the specified suffix.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello World";
        boolean result = str.endsWith("World");
        System.out.println(result); // Output: true
    }
}
```

**88. codePointAt()**

**Returns the Unicode code point of the character at the specified index.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello";
        int codePoint = str.codePointAt(1); // Unicode value of 'e'
        System.out.println(codePoint); // Output: 101 (Unicode value of 'e')
    }
```

```
}
```

**89. codePointBefore()**

**Returns the Unicode code point of the character before the specified index.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello";
        int codePoint = str.codePointBefore(1); // Unicode value of 'H'
        System.out.println(codePoint); // Output: 72 (Unicode value of 'H')
    }
}
```

**90. codePointCount()**

**Returns the number of Unicode code points in the specified range of the string.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello";
        int codePointCount = str.codePointCount(0, str.length());
        System.out.println(codePointCount); // Output: 5
    }
}
```

**91. offsetByCodePoints()**

**Returns the index that is offset by a specified number of code points starting from the given index.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello";
        int index = str.offsetByCodePoints(0, 3);
        System.out.println(index); // Output: 3 (index of 'l')
    }
}
```

**92. isEmpty()**

**Checks if the string is empty.**

```
public class Main {
    public static void main(String[] args) {
        String str = "";
        boolean result = str.isEmpty();
        System.out.println(result); // Output: true
    }
}
```

**93. isBlank()**

**Checks if the string is empty or contains only whitespace characters (available from Java 11).**

```
public class Main {
    public static void main(String[] args) {
        String str = "  ";
        boolean result = str.isBlank();
        System.out.println(result); // Output: true
    }
}
```

**94. repeat() (Java 11+)**

**Repeats the string a specified number of times.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hi ";
        String repeatedStr = str.repeat(3);
        System.out.println(repeatedStr); // Output: Hi Hi Hi
```

```
    }
}
```

**95. strip() (Java 11+)**

**Removes leading and trailing whitespace, similar to trim() but also removes other types of whitespace characters.**

```
public class Main {
    public static void main(String[] args) {
        String str = "\t  Hello World  \n";
        String strippedStr = str.strip();
        System.out.println(strippedStr); // Output: "Hello World"
    }
}
```

**96. stripLeading() (Java 11+)**

**Removes leading whitespace characters.**

```
public class Main {
    public static void main(String[] args) {
        String str = "   Hello";
        String result = str.stripLeading();
        System.out.println(result); // Output: "Hello"
    }
}
```

**97. stripTrailing() (Java 11+)**

**Removes trailing whitespace characters.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello   ";
        String result = str.stripTrailing();
        System.out.println(result); // Output: "Hello"
    }
}
```

**98. indent() (Java 12+)**

**Adjusts the indentation of the string by a specified number of spaces.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello\nWorld";
        String indentedStr = str.indent(4); // Adds 4 spaces to the beginning of each
line
        System.out.println(indentedStr);
    }
}
```

**99. transform() (Java 12+)**

**Applies a function to the string and returns the result.**

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello";
        String result = str.transform(s -> s + " World");
        System.out.println(result); // Output: Hello World
    }
}
```

**100. describeConstable() (Java 12+)**

**Returns an Optional containing the string (a representation of a constant).**

```
import java.util.Optional;
public class Main {
    public static void main(String[] args) {
        String str = "Hello";
        Optional<String> optional = str.describeConstable();
```

```
                optional.ifPresent(System.out::println); // Output: Hello
            }
        }
```

**101. resolveConstantDesc() (Java 12+)**

**Returns the string itself as a constant description.**

```
        public class Main {
            public static void main(String[] args) {
                String str = "Hello";
                Object constant = str.resolveConstantDesc(null);
                System.out.println(constant); // Output: Hello
            }}
```

**The world is yours, and everything in it, it's out there, get on your grind and get it."**
**- Rick Ross.**