**String Functions**

susanto ganguly

1. **length():** Returns the length of a string.

String str = "Hello, World!";

int length = str.length();

**// Output: 13**

1. **charAt(int index):** Returns the character at the specified index.

String str = " ";

char ch = str.charAt(1);

**// Output: 'a'**

1. **toUpperCase() and toLowerCase():** Converts a string to uppercase or lowercase.

String str = "Hello, World!";

String upper = str.toUpperCase();

String lower = str.toLowerCase();

**// Output: "HELLO, WORLD!" and "hello, world!"**

1. **substring(int beginIndex):** Returns a substring from the specified index.

String str = "Programming";

String sub = str.substring(5);

**// Output: "amming"**

1. **substring(int beginIndex, int endIndex):** Returns a substring between the specified indices.

String str = "Programming";

String sub = str.substring(5, 9);

// Output: "ammi"

1. **equals(Object obj):** Compares two strings for equality.

String str1 = "Hello";

String str2 = "Hello";

boolean isEqual = str1.equals(str2);

**// Output: true**

1. **equalsIgnoreCase(String anotherString):** Compares two strings for equality, ignoring case.

String str1 = "hello";

String str2 = "Hello";

boolean isEqual = str1.equalsIgnoreCase(str2);

**// Output: true**

1. **startsWith(String prefix):** Checks if a string starts with the specified prefix.

String str = " Programming";

boolean startsWith = str.startsWith(" ");

**// Output: true**

1. **endsWith(String suffix):** Checks if a string ends with the specified suffix.

String str = " Programming";

boolean endsWithIng = str.endsWith("Ing");

**// Output: true**

1. **indexOf(String str):** Returns the index of the first occurrence of the specified substring.

String str = " Programming";

int index = str.indexOf("Pro");

**// Output: 5**

1. **lastIndexOf(String str):** Returns the index of the last occurrence of the specified substring.

String str = " Programming ";

int lastIndex = str.lastIndexOf(" ");

**// Output: 18**

1. **replace(char oldChar, char newChar):** Replaces all occurrences of a character with another character.

String str = "Hello, World!";

String replaced = str.replace('o', '0');

susanto ganguly

**// Output: "Hell0, W0rld!"**

1. **replaceAll(String regex, String replacement):** Replaces all occurrences of a regular expression with a specified string.

String str = " is fun and is powerful";

String replaced = str.replaceAll(" ", "Python");

**// Output: "Python is fun and Python is powerful"**

1. **trim():** Removes leading and trailing white spaces.

String str = " Hello, World! ";

String trimmed = str.trim();

**// Output: "Hello, World!"**

1. **isEmpty():** Checks if a string is empty.

String str1 = "";

String str2 = "Hello";

boolean isEmpty1 = str1.isEmpty();

boolean isEmpty2 = str2.isEmpty();

**// Output: true for isEmpty1, false for isEmpty2**

1. **split(String regex):** Splits a string into an array of substrings based on a regular expression.

String str = "apple,banana,grape";

String[] fruits = str.split(",");

// Output: ["apple", "banana", "grape"]

1. **contains(CharSequence sequence):** Checks if a string contains a specified sequence of characters.

String str = "Hello, World!";

boolean containsHello = str.contains("Hello");

**// Output: true**

1. **startsWith(String prefix, int offset):** Checks if a string starts with the specified prefix at a given offset.

String str = " Programming";

boolean startsWithPro = str.startsWith("Pro", 5);

**// Output: true**

1. **endsWith(String suffix):** Checks if a string ends with the specified suffix.

String str = " Programming";

boolean endsWithIng = str.endsWith("Ing");

**// Output: true**

1. **replaceFirst(String regex, String replacement):** Replaces the first occurrence of a regular

expression with a specified string.

String str = " is fun and is powerful";

String replaced = str.replaceFirst(" ", "Python");

**// Output: "Python is fun and is powerful"**

1. **valueOf(Object obj):** Converts an object into a string representation.

int number = 42;

String str = String.valueOf(number);

**// Output: "42"**

1. **matches(String regex):** Checks if a string matches a given regular expression.

String email = "example@email.com";

boolean isEmail = email.matches("[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,4}");

**// Output: true**

1. **compareTo(String anotherString):** Compares two strings lexicographically. It returns a negative integer if the first string is less than the second, a positive integer if it's greater, and zero if they are equal.

String str1 = "apple";

String str2 = "banana";

int result = str1.compareTo(str2);

**// Output: Negative value (str1 is "apple" < str2 is "banana")**

susanto ganguly

1. **compareToIgnoreCase(String str):** Compares two strings lexicographically, ignoring case.

String str1 = "apple";

String str2 = "Banana";

int result = str1.compareToIgnoreCase(str2);

**// Output: Negative value (str1 is "apple" < str2 is "Banana")**

1. **isUpperCase() and isLowerCase():** Checks if a string is in uppercase or lowercase.

String str1 = "HELLO";

String str2 = "world";

boolean isUpper = str1.isUpperCase();

boolean isLower = str2.isLowerCase();

// Output: true for isUpper, true for isLower

1. **toUpperCase() and toLowerCase():** Converts a string to uppercase or lowercase.

String str = "Hello, World!";

String upper = str.toUpperCase();

String lower = str.toLowerCase();

// Output: "HELLO, WORLD!" and "hello, world!"

1. **contains(CharSequence sequence):** Checks if a string contains a specified sequence of

characters.

String str = " Programming";

boolean contains = str.contains(" ");

// Output: true

1. **startsWith(String prefix):** Checks if a string starts with the specified prefix.

String str = " Programming";

boolean startsWith = str.startsWith(" ");

// Output: true

1. **endsWith(String suffix):** Checks if a string ends with the specified suffix.

String str = " Programming";

boolean endsWithIng = str.endsWith("Ing");

**// Output: true**

1. **join(CharSequence delimiter, CharSequence... elements):** Joins multiple strings using a delimiter.

String[] words = {"Hello", "World"};

String joined = String.join(" ", words);

**// Output: "Hello World"**

1. **replace(char oldChar, char newChar):** Replaces all occurrences of a character with another character.

String str = "Hello, World!";

String replaced = str.replace('o', '0');

**// Output: "Hell0, W0rld!"**

1. **replaceFirst(String regex, String replacement):** Replaces the first occurrence of a regular expression with a specified string.

String str = " is fun and is powerful";

String replaced = str.replaceFirst(" ", "Python");

**// Output: "Python is fun and is powerful"**

1. **trim():** Removes leading and trailing white spaces from a string.

String str = " Hello, World! ";

String trimmed = str.trim();

**// Output: "Hello, World!"**

1. **matches(String regex):** Checks if a string matches a given regular expression.

String email = "example@email.com";

boolean isEmail = email.matches("[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,4}");

**// Output: true**

1. **split(String regex, int limit):** Splits a string into an array of substrings based on a regular expression with a specified limit.

susanto ganguly

String str = "apple,banana,grape,orange";

String[] fruits = str.split(",", 2);

**// Output: ["apple", "banana,grape,orange"]**

1. **isEmpty():** Checks if a string is empty.

String str1 = "";

String str2 = "Hello";

boolean isEmpty1 = str1.isEmpty();

boolean isEmpty2 = str2.isEmpty();

**// Output: true for isEmpty1, false for isEmpty2**

1. **indexOf(String str):** Returns the index of the first occurrence of the specified substring.

String str = " Programming";

int index = str.indexOf("Pro");

**// Output: 5**

1. **lastIndexOf(String str):** Returns the index of the last occurrence of the specified substring.

String str = " Programming ";

int lastIndex = str.lastIndexOf(" ");

**// Output: 18**

1. **charAt(int index):** Returns the character at the specified index.

String str = " ";

char ch = str.charAt(1);

**// Output: 'a'**

1. **length():** Returns the length of a string.

String str = "Hello, World!";

int length = str.length();

**// Output: 13**

1. **startsWith(String prefix, int offset):** Checks if a string starts with the specified prefix at a given offset.

String str = " Programming";

boolean startsWithPro = str.startsWith("Pro", 5);

**// Output: true**

1. **toCharArray():** Converts a string to a character array.

String str = " ";

char[] charArray = str.toCharArray();

**// Output: ['J', 'a', 'v', 'a']**

1. **substring(int beginIndex, int endIndex):** Returns a substring between the specified indices.

String str = " Programming";

String sub = str.substring(5, 14);

**// Output: "Programmi**"

1. **concat(String str):** Concatenates two strings.

String str1 = "Hello, ";

String str2 = "World!";

String result = str1.concat(str2);

**// Output: "Hello, World!"**

1. **compareToIgnoreCase(String str):** Compares two strings lexicographically, ignoring case.

String str1 = "apple";

String str2 = "Banana";

int result = str1.compareToIgnoreCase(str2);

**// Output: Negative value (str1 is "apple" < str2 is "Banana")**

1. **contentEquals(StringBuffer sb):** Checks if a string is equal to the contents of a StringBuffer.

String str = "Hello, World!";

StringBuffer buffer = new StringBuffer("Hello, World!");

boolean isEqual = str.contentEquals(buffer);

susanto ganguly

**// Output: true**

1. **endsWith(String suffix):** Checks if a string ends with the specified suffix.

String str = " Programming";

boolean endsWithIng = str.endsWith("Ing");

**// Output: true**

1. **format(String format, Object... args):** Formats a string using specified format and arguments.

String name = "John";

int age = 30;

String formatted = String.format("Name: %s, Age: %d", name, age);

**// Output: "Name: John, Age: 30"**

1. **replace(CharSequence target, CharSequence replacement):** Replaces all occurrences of a specified sequence with another sequence.

String str = "The quick brown fox";

String replaced = str.replace("fox", "dog");

**// Output: "The quick brown dog"**

1. **matches(String regex):** Checks if a string matches a given regular expression.

String email = "example@email.com";

boolean isEmail = email.matches("[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,4}");

**// Output: true**

1. **replaceAll(String regex, String replacement):** Replaces all occurrences of a regular expression with a specified string.

String str = " is fun and is powerful";

String replaced = str.replaceAll(" ", "Python");

**// Output: "Python is fun and Python is powerful"**

1. **split(String regex):** Splits a string into an array of substrings based on a regular expression.

String str = "apple,banana,grape";

String[] fruits = str.split(",");

**// Output: ["apple", "banana", "grape"]**

1. **toCharArray():** Converts a string to a character array.

String str = " ";

char[] charArray = str.toCharArray();

**// Output: ['J', 'a', 'v', 'a']**

1. **strip():** Removes leading and trailing whitespace, including Unicode whitespace characters.

String str = " Hello, World! ";

String stripped = str.strip();

**// Output: "Hello, World!"**

1. **isBlank():** Checks if a string is empty or contains only whitespace characters.

String str1 = "";

String str2 = " ";

boolean isBlank1 = str1.isBlank();

boolean isBlank2 = str2.isBlank();

**// Output: true for isBlank1, true for isBlank2**

1. **repeat(int count):** Repeats a string a specified number of times.

String str = " ";

String repeated = str.repeat(3);

**// Output: " "**

1. **stripLeading() and stripTrailing():** Remove leading or trailing whitespace, respectively.

String str = " Hello, World! ";

String strippedLeading = str.stripLeading();

String strippedTrailing = str.stripTrailing();

**// Output: "Hello, World! " for strippedLeading, " Hello, World!" for strippedTrailing**

1. **lines():** Splits a multi-line string into a Stream of lines.

String multiline = "Line 1\nLine 2\nLine 3";

susanto ganguly

List<String> lines = multiline.lines().collect(Collectors.toList());

**// Output: ["Line 1", "Line 2", "Line 3"]**

1. **compareTo(String anotherString):** Compares two strings lexicographically.

String str1 = "apple";

String str2 = "banana";

int result = str1.compareTo(str2);

**// Output: Negative value (str1 is "apple" < str2 is "banana")**

1. **codePointAt(int index):** Returns the Unicode code point at the specified index.

String str = " ";

int codePoint = str.codePointAt(1);

**// Output: 97 (Unicode code point for 'a')**

1. **indexOf(String str, int fromIndex):** Returns the index of the first occurrence of a substring, starting from the specified index.

String str = " Programming ";

int index = str.indexOf(" ", 5);

**// Output: 16**

1. **lastIndexOf(String str, int fromIndex):** Returns the index of the last occurrence of a substring, searching backward from the specified index.

String str = " Programming ";

int lastIndex = str.lastIndexOf(" ", 15);

**// Output: 0**

1. **isEmpty():** Checks if a string is empty.

String str1 = "";

String str2 = "Hello";

boolean isEmpty1 = str1.isEmpty();

boolean isEmpty2 = str2.isEmpty();

**// Output: true for isEmpty1, false for isEmpty2**

1. **offsetByCodePoints(int index, int codePointOffset):** Returns the index within the string that is codePointOffset code points away from the specified index.

String str = " ";

int newIndex = str.offsetByCodePoints(1, 2);

**// Output: 3 (New index after moving 2 code points from index 1)**

1. **codePointBefore(int index):** Returns the Unicode code point before the specified index.

String str = " ";

int codePoint = str.codePointBefore(2);

**// Output: 86 (Unicode code point for 'V')**

1. **codePointCount(int beginIndex, int endIndex):** Returns the number of Unicode code points in the specified range.

String str = " ";

int codePointCount = str.codePointCount(0, 3);

**// Output: 4 (There are four code points in " ")**

1. **subSequence(int beginIndex, int endIndex):** Returns a CharSequence that is a subsequence of this string.

String str = " Programming";

CharSequence sub = str.subSequence(5, 14);

**// Output: "Programmi"**

1. **stripIndent():** Removes common leading whitespace from all lines in a multi-line string.

String multiline = " Line 1\n Line 2\n Line 3";

String stripped = multiline.stripIndent();

**// Output: "Line 1\n Line 2\nLine 3"**

1. **formatted(Object... args):** Formats a string using specified arguments, similar to String.format().

String name = "John";

int age = 30;

susanto ganguly

String formatted = "Name: %s, Age: %d".formatted(name, age);

**// Output: "Name: John, Age: 30"**

1. **transform(Function<? super Character, ? extends Character> f):** Applies a function to each character in the string and returns a new string with the transformed characters.

String str = "Hello";

String transformed = str.transform(ch -> Character.toLowerCase(ch));

**// Output: "hello"**

1. **stripLeading() and stripTrailing():** Remove leading or trailing whitespace, respectively.

String str = " Hello, World! ";

String strippedLeading = str.stripLeading();

String strippedTrailing = str.stripTrailing();

**// Output: "Hello, World! " for strippedLeading, " Hello, World!" for strippedTrailing**

1. **stripLeading(CharSequence prefix):** Removes a leading prefix from a string.

String str = "Prefix: Hello";

String stripped = str.stripLeading("Prefix: ");

**// Output: "Hello"**

1. **stripTrailing(CharSequence suffix):** Removes a trailing suffix from a string.

String str = "Hello, World! Suffix";

String stripped = str.stripTrailing(" Suffix");

**// Output: "Hello, World!"**

1. **indent(int n):** Adds a specified level of indentation to each line of a multi-line string.

String multiline = "Line 1\nLine 2\nLine 3";

String indented = multiline.indent(2);

**// Output: " Line 1\n Line 2\n Line 3"**

1. **isBlank():** Checks if a string is empty or contains only whitespace characters.

String str1 = "";

String str2 = " ";

boolean isBlank1 = str1.isBlank();

boolean isBlank2 = str2.isBlank();

**// Output: true for isBlank1, true for isBlank2**

1. **stripTrailing():** Removes trailing whitespace, including Unicode whitespace characters.

String str = "Hello, World! \u2002";

String stripped = str.stripTrailing();

**// Output: "Hello, World!"**

1. **lines():** Splits a multi-line string into a Stream of lines.

String multiline = "Line 1\nLine 2\nLine 3";

List<String> lines = multiline.lines().collect(Collectors.toList());

**// Output: ["Line 1", "Line 2", "Line 3"]**

1. **repeat(int count):** Repeats a string a specified number of times.

String str = " ";

String repeated = str.repeat(3);

**// Output: " "**

1. **formatted():** Formats a string using the arguments provided in the string itself.

String formattedString = "Name: %s, Age: %d";

String formatted = formattedString.formatted("Alice", 25);

**// Output: "Name: Alice, Age: 25"**

1. **lines():** Splits a multi-line string into a Stream of lines.

String multiline = "Line 1\nLine 2\nLine 3";

List<String> lines = multiline.lines().collect(Collectors.toList());

**// Output: ["Line 1", "Line 2", "Line 3"]**