**String Class In Java**

* A String is a class in Java and it can be seen as a collection or the sequence of characters.
* Strings are used as an object in Java.
* Java supports different methods for String Manipulation.
* Java String class is an immutable class i.e. once it is created, it can’t be modified thereafter.
* This is the reason why **StringBuffer and StringBuilder** came into the picture as they are mutable and are used to do a lot of modification to the sequence of characters even after creation.

**Java String Methods**

Given below are the String methods that are used extensively in Java programming language for manipulating the Strings.

**#1) Length**

The length is the number of characters that a given string contains. Java has a length() method that gives the number of characters in a String.

**Given below is the programming Example.**

|  |  |
| --- | --- |
| **package** codes;  **import** java.lang.String;  **public** **class** StringMethods {  **public** **static** **void** main(String[] args) {  String str = "Saket Saurav";  System.out.println(str.length());  }  }  **#2) Concatenation**  Although Java uses a ‘+’ operator for concatenating two or more strings.  A concat() is an inbuilt method for String concatenation in Java.   |  | | --- | | String str1 = "Software";          String str2 = "Testing";          System.out.println(str1 + str2);          System.out.println(str1.concat(str2)); | |

**String charAt()**

This method is used to retrieve a single character from a given String.

**The syntax is given as:**

|  |
| --- |
| **char** charAt(**int** i); |

The value of ‘i’ should not be negative and it should specify the location of a given String i.e. if a String length is 5, then the value of ‘i’ should be less than 5.

**Java String compareTo()**

This method is used to compare two Strings. The comparison is based on alphabetical order. In general terms, a String is less than the other if it comes before the other in the dictionary.

**Syntax**

public int compareTo(String string2)

**Technical Details**

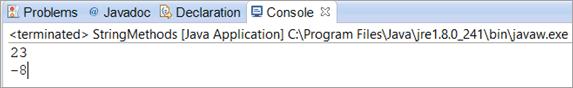
Returns: An int value: 0 if the string is equal to the other string.

< 0 if the string is lexicographically less than the other string

> 0 if the string is lexicographically greater than the other string (more characters)

|  |
| --- |
| **import** java.lang.String;  **public** **class** StringMethods {  **public** **static** **void** main(String[] args) {            String str1 = "Zeus";          String str2 = "Chinese";          String str3 = "American";          String str4 = "Indian";            System.out.println(str1.compareTo(str2));            //C comes 23 positions before Z, so it will give you 23          System.out.println(str3.compareTo(str4));            // I comes 8 positions after A, so it will give you -8      }  } |

**Output:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/02/JavaStringcompareTo.png)

**Java String indexOf()**

This method is used to perform a search operation for a specific character or a substring on the main String. There is one more method known as lastIndexOf() which is also commonly used.

indexOf() is used to search for the first occurrence of the character.

lastIndexOf() is used to search for the last occurrence of the character.

|  |
| --- |
| **package** codes;  **import** java.lang.String;    **public** **class** StringMethods {    **public** **static** **void** main(String[] args) {            String str = "Saket Saurav " + "performing a search";          System.out.println(str);          System.out.println("index of 'p' is " + str.indexOf('p'));          System.out.println("index of 'u' is " + str.indexOf('u'));          System.out.println("last index of 'S' is " + str.lastIndexOf('S'));          System.out.println("last index of 's' is " + str.lastIndexOf('s'));      }    } |

**String equals(String anotherString) method**

The equals() method of string in Java is used to verify if both the strings are equal or not.

The equals method accepts another string as an argument and then checks for the equality of both the strings.

If both the strings are equal, true is returned else false is returned.

Syntax:

string.equals(anotherString)

**String join()**

The join() method of string in Java as the name suggests is used to join a group of strings using the joiner between them.

The joiner variable can be any character, string or a sequence of characters.

Syntax:

string.join(joiner, str1, str2, str3,..)

String str1Join = String.join("-","Have","a","Nice","day");

System.out.println(str1Join);

**String trim() Method**

The trim() method of string in Java is used to trim (or remove) the extra white spaces from the specified string from both the ends.

Syntax:

string.trim()

Example:

String str = " Coding is ";

System.out.println(str + " awesome");

str = str.trim();

System.out.println(str + " awesome");

**Output:**

Coding is awesome

Coding is awesome

String replace(char oldChar, char newChar)

The replace() method of string in Java as the name suggests is used to replace all the specified character of the string with another character.

Syntax:

string.replace(oldChar, newChar)

**Example:**

String str = "Hi, i will be replaced with a";

str = str.replace('i','a');

System.out.println(str);

// Java Program to Sort Names in an Alphabetical Order

**import** java.io.\*;

**class** StringOrdering {

**public** **static** **void** main(String[] args)

    {

        // storing input in variable

**int** n = 4;

        // create string array called names

        String names[]

            = { "Bombay", "Culcutta", "Ahemdabad", "Delhi" };

        String temp;

**for** (**int** i = 0; i < n; i++) {

**for** (**int** j = i + 1; j < n; j++) {

                // to compare one string with other strings

**if** (names[i].compareTo(names[j]) > 0) {

                    // swapping

                    temp = names[i];

                    names[i] = names[j];

                    names[j] = temp;

                }

            }

        }

        // print output array

        System.out.println(

            "The names in alphabetical order are: ");

**for** (**int** i = 0; i < n; i++) {

            System.out.println(names[i]);

        }

    }

}

**some important features and methods of the StringBuffer class:**

|  |  |  |
| --- | --- | --- |
| **Modifier and Type** | **Method** | **Description** |
| public synchronized StringBuffer | append(String s) | It is used to append the specified string with this string.  The append() method is overloaded like append(char),  append(boolean),  append(int),  append(float), append(double)  StringBuffer sb=new StringBuffer("Hello ");  sb.append("Java"); |
| public synchronized StringBuffer | insert(int offset, String s) | It is used to insert the specified string with this string at the specified position.  The insert() method is overloaded like insert(int, char),  insert(int, boolean),  insert(int, int),  insert(int, float),  insert(int, double)  StringBuffer sb=new StringBuffer("Hello ");  sb.insert(1,"Java"); |
| public synchronized StringBuffer | replace(int startIndex, int endIndex, String str) | The replace() method replaces the given string from the specified beginIndex and endIndex-1.  StringBuffer sb=new StringBuffer("Hello");  sb.replace(1,3,"Java"); |
| public synchronized StringBuffer | delete(int startIndex, int endIndex) | The delete() method of the StringBuffer class deletes the string from the specified beginIndex to endIndex-1.   StringBuffer sb = **new** StringBuffer("Hello");  sb.delete(1, 3); |
| public synchronized StringBuffer | reverse() | is used to reverse the string. |
| public int | capacity() | It is used to return the current capacity. |
| public void | ensureCapacity(int minimumCapacity) | It is used to ensure the capacity at least equal to the given minimum. |
| public char | charAt(int index) | It is used to return the character at the specified position. |
| public int | length() | It is used to return the length of the string i.e. total number of characters. |
| public String | substring(int beginIndex) | It is used to return the substring from the specified beginIndex. |
| public String | substring(int beginIndex, int endIndex) | It is used to return the substring from the specified beginIndex and endIndex. |

1. StringBuffer objects are mutable, meaning that you can change the contents of the buffer without creating a new object.
2. The initial capacity of a StringBuffer can be specified when it is created, or it can be set later with the ensureCapacity() method.
3. The append() method is used to add characters, strings, or other objects to the end of the buffer.
4. The insert() method is used to insert characters, strings, or other objects at a specified position in the buffer.
5. The delete() method is used to remove characters from the buffer.
6. The reverse() method is used to reverse the order of the characters in the buffer.