JAVA Strings

Java String

In [Java](https://www.javatpoint.com/java-tutorial), string is basically an object that represents sequence of char values. An [array](https://www.javatpoint.com/array-in-java) of characters works same as Java string. For example:

char[] ch={'j','a','v','a','t','p','o','i','n','t'};

String s=new String(ch);

Is same as

String s="javatpoint";

**Java String** class provides a lot of methods to perform operations on strings such as compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() etc.

The java.lang.String class implements Serializable, Comparable and CharSequence [interfaces](https://www.javatpoint.com/interface-in-java).



## **CharSequence Interface**

The CharSequence interface is used to represent the sequence of characters. String, [StringBuffer](https://www.javatpoint.com/StringBuffer-class) and [StringBuilder](https://www.javatpoint.com/StringBuilder-class) classes implement it. It means, we can create strings in Java by using these three classes.



The Java String is immutable which means it cannot be changed. Whenever we change any string, a new instance is created. For mutable strings, you can use StringBuffer and StringBuilder classes.

### **What is String in Java?**

Generally, String is a sequence of characters. But in Java, string is an object that represents a sequence of characters. The java.lang.String class is used to create a string object.

### **How to create a string object?**

There are two ways to create String object:

1. By string literal
2. By new keyword

Java String literal is created by using double quotes. For Example:

* String s="welcome";

Each time you create a string literal, the JVM checks the "string constant pool" first. If the string already exists in the pool, a reference to the pooled instance is returned. If the string doesn't exist in the pool, a new string instance is created and placed in the pool. For example:

* String s1="Welcome";
* String s2="Welcome";//It doesn't create a new instance



In the above example, only one object will be created. Firstly, JVM will not find any string object with the value "Welcome" in string constant pool that is why it will create a new object. After that it will find the string with the value "Welcome" in the pool, it will not create a new object but will return the reference to the same instance.

### **Why Java uses the concept of String literal?**

To make Java more memory efficient (because no new objects are created if it exists already in the string constant pool).

### **By new keyword**

* String s=**new** String("Welcome");//creates two objects and one reference variable

In such case, [JVM](https://www.javatpoint.com/jvm-java-virtual-machine) will create a new string object in normal (non-pool) heap memory, and the literal "Welcome" will be placed in the string constant pool. The variable s will refer to the object in a heap (non-pool).

## **All String Methods**

The String class has a set of built-in methods that you can use on strings.

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Return Type** |
| [charAt()](https://www.w3schools.com/java/ref_string_charat.asp) | Returns the character at the specified index (position) | char |
| [codePointAt()](https://www.w3schools.com/java/ref_string_codepointat.asp) | Returns the Unicode of the character at the specified index | int |
| [codePointBefore()](https://www.w3schools.com/java/ref_string_codepointbefore.asp) | Returns the Unicode of the character before the specified index | int |
| [codePointCount()](https://www.w3schools.com/java/ref_string_codepointcount.asp) | Returns the number of Unicode values found in a string. | int |
| [compareTo()](https://www.w3schools.com/java/ref_string_compareto.asp) | Compares two strings lexicographically | int |
| [compareToIgnoreCase()](https://www.w3schools.com/java/ref_string_comparetoignorecase.asp) | Compares two strings lexicographically, ignoring case differences | int |
| [concat()](https://www.w3schools.com/java/ref_string_concat.asp) | Appends a string to the end of another string | String |
| [contains()](https://www.w3schools.com/java/ref_string_contains.asp) | Checks whether a string contains a sequence of characters | boolean |
| [contentEquals()](https://www.w3schools.com/java/ref_string_contentequals.asp) | Checks whether a string contains the exact same sequence of characters of the specified CharSequence or StringBuffer | boolean |
| [copyValueOf()](https://www.w3schools.com/java/ref_string_copyvalueof.asp) | Returns a String that represents the characters of the character array | String |
| [endsWith()](https://www.w3schools.com/java/ref_string_endswith.asp) | Checks whether a string ends with the specified character(s) | boolean |
| [equals()](https://www.w3schools.com/java/ref_string_equals.asp) | Compares two strings. Returns true if the strings are equal, and false if not | boolean |
| [equalsIgnoreCase()](https://www.w3schools.com/java/ref_string_equalsignorecase.asp) | Compares two strings, ignoring case considerations | boolean |
| format() | Returns a formatted string using the specified locale, format string, and arguments | String |
| getBytes() | Encodes this String into a sequence of bytes using the named charset, storing the result into a new byte array | byte[] |
| getChars() | Copies characters from a string to an array of chars | void |
| [hashCode()](https://www.w3schools.com/java/ref_string_hashcode.asp) | Returns the hash code of a string | int |
| [indexOf()](https://www.w3schools.com/java/ref_string_indexof.asp) | Returns the position of the first found occurrence of specified characters in a string | int |
| intern() | Returns the canonical representation for the string object | String |
| [isEmpty()](https://www.w3schools.com/java/ref_string_isempty.asp) | Checks whether a string is empty or not | boolean |
| [lastIndexOf()](https://www.w3schools.com/java/ref_string_lastindexof.asp) | Returns the position of the last found occurrence of specified characters in a string | int |
| [length()](https://www.w3schools.com/java/ref_string_length.asp) | Returns the length of a specified string | int |
| matches() | Searches a string for a match against a regular expression, and returns the matches | boolean |
| offsetByCodePoints() | Returns the index within this String that is offset from the given index by codePointOffset code points | int |
| regionMatches() | Tests if two string regions are equal | boolean |
| [replace()](https://www.w3schools.com/java/ref_string_replace.asp) | Searches a string for a specified value, and returns a new string where the specified values are replaced | String |
| replaceFirst() | Replaces the first occurrence of a substring that matches the given regular expression with the given replacement | String |
| replaceAll() | Replaces each substring of this string that matches the given regular expression with the given replacement | String |
| split() | Splits a string into an array of substrings | String[] |
| [startsWith()](https://www.w3schools.com/java/ref_string_startswith.asp) | Checks whether a string starts with specified characters | boolean |
| subSequence() | Returns a new character sequence that is a subsequence of this sequence | CharSequence |
| substring() | Returns a new string which is the substring of a specified string | String |
| toCharArray() | Converts this string to a new character array | char[] |
| [toLowerCase()](https://www.w3schools.com/java/ref_string_tolowercase.asp) | Converts a string to lower case letters | String |
| toString() | Returns the value of a String object | String |
| [toUpperCase()](https://www.w3schools.com/java/ref_string_touppercase.asp) | Converts a string to upper case letters | String |
| [trim()](https://www.w3schools.com/java/ref_string_trim.asp) | Removes whitespace from both ends of a string | String |
| valueOf() | Returns the string representation of the specified value | String |

String is immutable for several reasons, here is a summary:

* **Security**: parameters are typically represented as String in network connections, database connection urls, usernames/passwords etc. If it were mutable, these parameters could be easily changed.
* **Synchronization** and concurrency: making String immutable automatically makes them thread safe thereby solving the synchronization issues.
* **Caching**: when compiler optimizes your String objects, it sees that if two objects have same value (a="test", and b="test") and thus you need only one string object (for both a and b, these two will point to the same object).
* **Class loading**: String is used as arguments for class loading. If mutable, it could result in wrong class being loaded (because mutable objects change their state).

# Java StringBuffer Class

Java StringBuffer class is used to create mutable (modifiable) String objects. The StringBuffer class in Java is the same as String class except it is mutable i.e. it can be changed.

# Java StringBuffer Class

Java StringBuffer class is used to create mutable (modifiable) String objects. The StringBuffer class in Java is the same as String class except it is mutable i.e. it can be changed.

### **Important Constructors of StringBuffer Class**

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| StringBuffer() | It creates an empty String buffer with the initial capacity of 16. |
| StringBuffer(String str) | It creates a String buffer with the specified string.. |
| StringBuffer(int capacity) | It creates an empty String buffer with the specified capacity as length. |

### **What is a mutable String?**

A String that can be modified or changed is known as mutable String. StringBuffer and StringBuilder classes are used for creating mutable strings.

### **Important methods of 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) etc. |
| 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) etc. |
| public synchronized StringBuffer | replace(int startIndex, int endIndex, String str) | It is used to replace the string from specified startIndex and endIndex. |
| public synchronized StringBuffer | delete(int startIndex, int endIndex) | It is used to delete the string from specified startIndex and endIndex. |
| 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. |

# Java StringBuilder Class

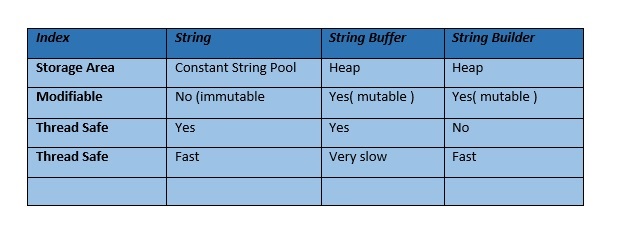
Java StringBuilder class is used to create mutable (modifiable) String. The Java StringBuilder class is same as StringBuffer class except that it is non-synchronized. It is available since JDK 1.5.

## **Important Constructors of StringBuilder class**

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| StringBuilder() | It creates an empty String Builder with the initial capacity of 16. |
| StringBuilder(String str) | It creates a String Builder with the specified string. |
| StringBuilder(int length) | It creates an empty String Builder with the specified capacity as length. |

## **Important methods of StringBuilder class**

|  |  |
| --- | --- |
| **Method** | **Description** |
| public StringBuilder 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) etc. |
| public StringBuilder 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) etc. |
| public StringBuilder replace(int startIndex, int endIndex, String str) | It is used to replace the string from specified startIndex and endIndex. |
| public StringBuilder delete(int startIndex, int endIndex) | It is used to delete the string from specified startIndex and endIndex. |
| public StringBuilder reverse() | It 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. |



StringBuilder and StringBuffer are almost the same. The difference is that StringBuffer is synchronized and StringBuilder is not. Although, StringBuilder is faster than StringBuffer, the difference in performance is very little. StringBuilder is a SUN's replacement of StringBuffer. It just avoids synchronization from all the public methods. Rather than that, their functionality is the same.

Example of good usage:

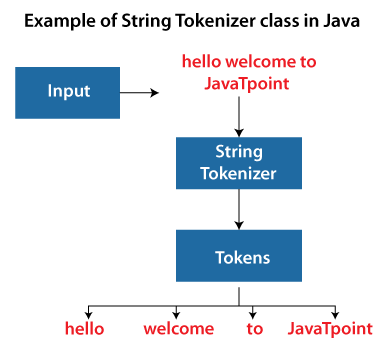
If your text is going to change and is used by multiple threads, then it is better to use StringBuffer. If your text is going to change but is used by a single thread, then use StringBuilder.

# StringTokenizer in Java

The **java.util.StringTokenizer** class allows you to break a String into tokens. It is simple way to break a String. It is a legacy class of Java.

It doesn't provide the facility to differentiate numbers, quoted strings, identifiers etc. like StreamTokenizer class. We will discuss about the StreamTokenizer class in I/O chapter.

In the StringTokenizer class, the delimiters can be provided at the time of creation or one by one to the tokens.



|  |  |
| --- | --- |
| **Constructor** | **Description** |
| StringTokenizer(String str) | It creates StringTokenizer with specified string. |
| StringTokenizer(String str, String delim) | It creates StringTokenizer with specified string and delimiter. |
| StringTokenizer(String str, String delim, boolean returnValue) | It creates StringTokenizer with specified string, delimiter and returnValue. If return value is true, delimiter characters are considered to be tokens. If it is false, delimiter characters serve to separate tokens. |
| Methods | Description |
| boolean hasMoreTokens() | It checks if there is more tokens available. |
| String nextToken() | It returns the next token from the StringTokenizer object. |
| String nextToken(String delim) | It returns the next token based on the delimiter. |
| boolean hasMoreElements() | It is the same as hasMoreTokens() method. |
| Object nextElement() | It is the same as nextToken() but its return type is Object. |
| int countTokens() | It returns the total number of tokens. |

1. **import** java.util.StringTokenizer;
2. **public** **class** Simple{
3. **public** **static** **void** main(String args[]){
4. StringTokenizer st = **new** StringTokenizer("my name is khan"," ");
5. **while** (st.hasMoreTokens()) {
6. System.out.println(st.nextToken());
7. }
8. }
9. }
10. **import** java.util.\*;
12. **public** **class** Test {
13. **public** **static** **void** main(String[] args) {
14. StringTokenizer st = **new** StringTokenizer("my,name,is,khan");
16. // printing next token
17. System.out.println("Next token is : " + st.nextToken(","));
18. }
19. }