# StringTokenizer in Java

The **java.util.StringTokenizer** class allows you to break a string into tokens. It is simple way to break string.

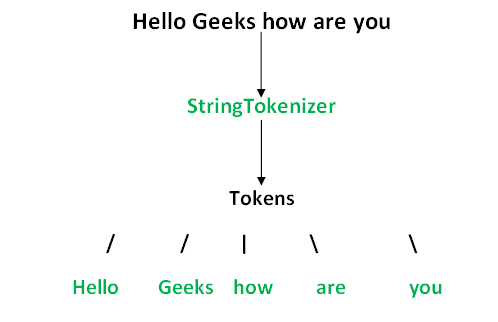
It doesn't provide the facility to differentiate numbers, quoted strings, identifiers etc. like StreamTokenizer class. We will discuss about the StreamTokenizer class in I/O chapter.

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| StringTokenizer(String str) | creates StringTokenizer with specified string. |
| StringTokenizer(String str, String delim) | creates StringTokenizer with specified string and delimeter. |
| StringTokenizer(String str, String delim, boolean returnValue) | creates StringTokenizer with specified string, delimeter 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**

|  |  |
| --- | --- |
| **Public method** | **Description** |
| boolean hasMoreTokens() | checks if there is more tokens available. |
| String nextToken() | returns the next token from the StringTokenizer object. |
| String nextToken(String delim) | returns the next token based on the delimeter. |
| boolean hasMoreElements() | same as hasMoreTokens() method. |
| Object nextElement() | same as nextToken() but its return type is Object. |
| int countTokens() | returns the total number of tokens. |

StringTokenizer class in Java is used to break a string into tokens.

**Example:**  


**Constructors:**

**StringTokenizer(String str) :**

**str** is string to be tokenized.

Considers default delimiters like new line, space, tab, carriage return and form feed.

**StringTokenizer(String str, String delim) :**

**delim** is set of delimiters that are used to tokenizethe given string.

**StringTokenizer(String str, String delim, boolean flag):**

The first two parameters have same meaning. The flag serves following purpose.

If the **flag** is **false**, delimiter characters serve to

separate tokens. For example, if string is "hello geeks"and delimiter is " ", then tokens are "hello" and "geeks".

If the **flag** is **true**, delimiter characters are

considered to be tokens. For example, if string is "hello geeks" and delimiter is " ", then tokens are "hello", " and "geeks".

/\* A Java program to illustrate working of StringTokenizer    class:\*/

import java.util.\*;

public class NewClass

{

    public static void main(String args[])

    {

        System.out.println("Using Constructor 1 - ");

        StringTokenizer st1 =  new StringTokenizer("Hello Geeks How are you", " ");

        while (st1.hasMoreTokens())

            System.out.println(st1.nextToken());

        System.out.println("Using Constructor 2 - ");

        StringTokenizer st2 =   new StringTokenizer("JAVA : Code : String", " :");

        while (st2.hasMoreTokens())

            System.out.println(st2.nextToken());

        System.out.println("Using Constructor 3 - ");

        StringTokenizer st3 =   new StringTokenizer("JAVA : Code : String", " :",  true);

        while (st3.hasMoreTokens())

            System.out.println(st3.nextToken());

    }

}

**StringTokenizer methods in Java with example :**

/ / | \ \

hasMoreToken nextToken countTokens nextElement hasMoreElements

.  
**Following are the StringTokenizer class methods :**

* **hasMoreTokens():**The method **java.util.StringTokenizer.hasMoreTokens()** plays role in testing, if tokens are present for the StringTokenizer’s string.  
  Those characters that are considered to be delimiters by the StringTokenizer object are changed to characters in the string delimiter. Then the next token to the current position in the string is returned.  
  **Syntax:**

public boolean hasMoreTokens()

**Returns:** True if and only if next token to the current position in the string exists, else false.

* **nextToken():**The method **java.util.StringTokenizer.nextToken()** returns next token from the given StringTokenizer.  
  **Syntax:**

public String nextToken()

**Return:** the next token from the given StringTokenizer if present.

**Throws:** NoSuchElementException - if no more token are left.

* **countTokens():**The method **java.util.StringTokenizer.countTokens()** returns total number of tokens present so that we can use nextToken() method before it gives an exception..  
  **Syntax:**

public int countTokens()

**Return :** the number of tokens remaining in the string using the current delimiter set.

|  |
| --- |
| // Pragrom in Java illustrating the methods of StringTokenizer class:  // hasMoreToken     nextToken   countTokens  import java.util.\*;  public class NewClass  {      public static void main(String args[])      {          String mydelim = " : ";          String mystr = "JAVA : Code : String : Tokenizer : Geeks";            // Use of Constructor 2          // Here we are passing Delimiter - "mydelim"          StringTokenizer geeks3 = new StringTokenizer(mystr, mydelim);            // Printing count of tokens and tokens          int count = geeks3.countTokens();          System.out.println("Number of tokens : " + count + "\n");          for (int i = 0; i <count; i++)              System.out.println("token at [" + i + "] : " + geeks3.nextToken());            // .hasMoreTokens() method checks for more Tokens.          // Here not working as no Tokens left          while (geeks3.hasMoreTokens())                // .nextToken is method is returning next token.              System.out.println(geeks3.nextToken());      }  } |

Output:

Number of tokens : 5

token at [0] : JAVA

token at [1] : Code

token at [2] : String

token at [3] : Tokenizer

token at [4] : Geeks

* **nextElement():**The method **java.util.StringTokenizer.nextElements()** works similar to nextToken except that it returns Object rather than String.  
  Exists so that this class can implement the Enumeration interface.  
  **Syntax:**

public Object nextElement()

**Return:** the next token from the given StringTokenizer.

**Throws:**NoSuchElementException - if there are no more tokens left.

* **hasMoreElements():**This method**java.util.StringTokenizer.hasMoreElements()** returns same value as hasMoreToken. It exists so that the class can implement the Enumeration interface.  
  **Syntax:**

public boolean hasMoreElements()

**Return:** true if tokens are present in the string, else false

|  |
| --- |
| // Program in Java illustrating the methods of StringTokenizer  // class: hasMoreElements, nextElement and nextElement  import java.util.\*;  public class NewClass  {      public static void main(String args[])      {          String mydelim = " : ";          String mystr = "JAVA : Code : String : Tokenizer : Geeks";            // Use of Constructor 2          // Here we are passing Delimiter - "mydelim"          StringTokenizer geeks =  new StringTokenizer(mystr, mydelim);            //  .countTokens() method counts no. of tokens present.          int count = geeks.countTokens();          System.out.println("Number of tokens : " + count);            //  use of hasMoreElements() - true if tokens are present          while (geeks.hasMoreElements())                //  use of nextElement() - returns the next token              System.out.println(geeks.nextElement());      }  } |

Output:

Number of tokens : 5

JAVA

Code

String

Tokenizer

Geeks

**Important Points:**

* counttoken() method is a good alternative in using the combination hasMoreTokens and nextToken().
* The combination of countTokens and nextToken is used if you are interested in the number of tokens also.

# StringTokenizer nextElement() Method in Java with Examples

The **nextElement()** method of **StringTokenizer class** is also used to return the next token one after another from this StringTokenizer. It is similar to the nextToken() method, except that the return type is Object rather than the String.

**Syntax:**

public Object nextElement()

**Parameters:** The method does not take any parameters.

**Return Value:** The method returns the next token present in the line of the string tokenizer.

Below programs illustrate the working of nextElement() Method of StringTokenizer:

**Example 1:**

|  |
| --- |
| // Java code to illustrate nextElement() method    import java.util.\*;    public class StringTokenizer\_Demo {      public static void main(String args[])      {          // Creating a StringTokenizer          StringTokenizer str\_arr = new StringTokenizer(  "Lets practice at GeeksforGeeks");            // Displaying the Tokens          while (str\_arr.hasMoreElements()) {              System.out.println("The Next token: " + str\_arr.nextElement());          }      }  } |

**Output:**

The Next token: Lets

The Next token: practice

The Next token: at

The Next token: GeeksforGeeks

**Example 2:**

|  |
| --- |
| // Java code to illustrate nextElement() method    import java.util.\*;    public class StringTokenizer\_Demo {      public static void main(String args[])      {          // Creating a StringTokenizer          StringTokenizer str\_arr  = new StringTokenizer(“Welcome to GeeksforGeeks");            // Displaying the Tokens          while (str\_arr.hasMoreElements()) {              System.out.println("The Next token: " + str\_arr.nextElement());          }      }  } |

**Output:**

The Next token: Welcome

The Next token: to

The Next token: GeeksforGeeks

# StringTokenizer nextToken() Method in Java

The **nextToken()** method of **StringTokenizer class** is used to return the next token one after another from this StringTokenizer.

**Syntax:**

public String nextToken()

**Parameters:** The method does not take any parameters.

**Return Value:** The method returns the next token present in the line of the string tokenizer.

Below programs illustrate the working of nextToken() Method of StringTokenizer:

**Example 1:**

|  |
| --- |
| // Java code to illustrate nextToken() method    import java.util.\*;    public class StringTokenizer\_Demo {      public static void main(String args[])      {          // Creating a StringTokenizer          StringTokenizer str\_arr  = new StringTokenizer( "Lets practice at GeeksforGeeks");            // Displaying the Tokens          while (str\_arr.hasMoreTokens()) {              System.out.println("The Next token: " + str\_arr.nextToken());          }      }  } |

**Output:**

The Next token: Lets

The Next token: practice

The Next token: at

The Next token: GeeksforGeeks

**Example 2:**

|  |
| --- |
| // Java code to illustrate nextToken() method    import java.util.\*;    public class StringTokenizer\_Demo {      public static void main(String args[])      {          // Creating a StringTokenizer          StringTokenizer str\_arr = new StringTokenizer( "Welcome to GeeksforGeeks");            // Displaying the Tokens          while (str\_arr.hasMoreTokens()) {  System.out.println("The Next token: " + str\_arr.nextToken());          }      }  } |

**Output:**

The Next token: Welcome

The Next token: to

The Next token: GeeksforGeeks

# StringTokenizer hasMoreTokens() Method in Java

The **hasMoreTokens()** method of **StringTokenizer class** checks whether there are any more tokens available with this StringTokenizer.

**Syntax:**

public boolean hasMoreTokens()

**Parameters:** The method does not take any parameters.

**Return Value:** The method returns boolean True if the availability of at least one more token is found in the string after the current position else false.

Below programs illustrate the working of hasMoreTokens() Method of StringTokenizer:  
**Example 1:**

|  |
| --- |
| // Java code to illustrate hasMoreTokens() method    import java.util.\*;    public class StringTokenizer\_Demo {      public static void main(String args[])      {          // Creating a StringTokenizer          StringTokenizer str\_arr = new StringTokenizer( "Lets practice at GeeksforGeeks");            // Counting the tokens          System.out.println("The number of Tokens are: "  + str\_arr.countTokens());            // Checking for any tokens          System.out.println(str\_arr.hasMoreTokens());            // Checking and displaying the Tokens          while (str\_arr.hasMoreTokens()) {  System.out.println("The Next token: " + str\_arr.nextToken());          }      }  } |

**Output:**

The number of Tokens are: 4

true

The Next token: Lets

The Next token: practice

The Next token: at

The Next token: GeeksforGeeks

**Example 2:**

|  |
| --- |
| // Java code to illustrate hasMoreTokens() method    import java.util.\*;    public class StringTokenizer\_Demo {      public static void main(String args[])      {          // Creating a StringTokenizer          StringTokenizer str\_arr  = new StringTokenizer("");            // Counting the tokens          System.out.println("The number of Tokens are: " + str\_arr.countTokens());            // Checking for any tokens          System.out.println(str\_arr.hasMoreTokens());      }  } |

**Output:**

The number of Tokens are: 0

false

# StringTokenizer countTokens() Method in Java

The **countTokens()** method of **StringTokenizer class** calculate the number of times that this tokenizer’s **nextToken** method can be called before the method generates any further exception.  
**Note:** The current position is not advanced during the process.

**Syntax:**

public int countTokens()

**Parameters:** The method does not take any parameters.

**Return Value:** The method is used to return the number of tokens remaining in the string using the current delimiter set.

Below programs illustrate the working of countTokens() Method of StringTokenizer:  
**Example 1:**

|  |
| --- |
| // Java code to illustrate countTokens() method    import java.util.\*;    public class StringTokenizer\_Demo1 {      public static void main(String args[])      {            // Creating a StringTokenizer          StringTokenizer str\_arr  = new StringTokenizer(  "Lets practice at GeeksforGeeks");            // Counting the tokens          int count = str\_arr.countTokens();          System.out.println("Total number of Tokens: "  + count);            // Print the tokens          for (int i = 0; i < count; i++)              System.out.println("token at [" + i + "] : " + str\_arr.nextToken());      }  } |

**Output:**

Total number of Tokens: 4

token at [0] : Lets

token at [1] : practice

token at [2] : at

token at [3] : GeeksforGeeks

**Example 2:**

|  |
| --- |
| // Java code to illustrate countTokens() method    import java.util.\*;    public class StringTokenizer\_Demo2 {      public static void main(String args[])      {            // Creating a StringTokenizer          StringTokenizer str\_arr = new StringTokenizer( "Welcome to GeeksforGeeks");            // Counting the tokens          int count = str\_arr.countTokens();          System.out.println("Total number of Tokens: "  + count);            // Print the tokens          for (int i = 0; i < count; i++)              System.out.println("token at [" + i + "] : "  + str\_arr.nextToken());      }  } |

**Output:**

Total number of Tokens: 3

token at [0] : Welcome

token at [1] : to

token at [2] : GeeksforGeeks

# StringTokenizer hasMoreElements() Method in Java

The **hasMoreElements()** method of **StringTokenizer class** also checks whether there are any more tokens available with this StringTokenizer. It is similar to the hasMoreTokens(). The method exists exclusively so that the Enumeration interface of this class can be implemented.

**Syntax:**

public boolean hasMoreElements()

**Parameters:** The method does not take any parameters.

**Return Value:** The method returns boolean True if the availability of at least one more token is found in the string after the current position else false.

Below programs illustrate the working of hasMoreElements() Method of StringTokenizer:  
**Example 1:**

|  |
| --- |
| // Java code to illustrate hasMoreElements() method    import java.util.\*;    public class StringTokenizer\_Demo {      public static void main(String args[])      {          // Creating a StringTokenizer          StringTokenizer str\_arr = new StringTokenizer(  "Lets practice at GeeksforGeeks");            // Counting the tokens          System.out.println("The number of Tokens are: " + str\_arr.countTokens());            // Checking for any tokens          System.out.println(str\_arr.hasMoreElements());            // Checking and displaying the Tokens          while (str\_arr.hasMoreElements()) {  System.out.println("The Next token: " + str\_arr.nextToken());          }      }  } |

**Output:**

The number of Tokens are: 4

true

The Next token: Lets

The Next token: practice

The Next token: at

The Next token: GeeksforGeeks

**Example 2:**

|  |
| --- |
| // Java code to illustrate hasMoreElements() method    import java.util.\*;    public class StringTokenizer\_Demo {      public static void main(String args[])      {          // Creating a StringTokenizer          StringTokenizer str\_arr  = new StringTokenizer("");            // Counting the tokens          System.out.println("The number of Tokens are: "  + str\_arr.countTokens());            // Checking for any tokens          System.out.println(str\_arr.hasMoreElements());      }  } |

**Output:**

The number of Tokens are: 0

false