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.

Constructors of StringTokenizer class

There are 3 constructors defined in the StringTokenizer class.

|  |  |
| --- | --- |
| **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 of StringTokenizer class

The 6 useful methods of StringTokenizer class are as follows:

|  |  |
| --- | --- |
| **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. |

Simple example of StringTokenizer class

Let's see the simple example of StringTokenizer class that tokenizes a string "Methodist college of Engineering and Technology" on the basis of whitespace.

**import java.util.StringTokenizer;**

**public class Simple{**

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

**StringTokenizer st = new StringTokenizer("Methodist college of Engineering and Technology "," ");**

 **while (st.hasMoreTokens()) {**

 **System.out.println(st.nextToken());**

 **}**

 **}**

**}**

Example of nextToken(String delim) method of StringTokenizer class

**import java.util.\*;**

**public class Test {**

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

 **StringTokenizer st = new StringTokenizer("Methodist,college,of,Engineering,and,Technology ");**

 **// printing next token**

 **System.out.println("Next token is : " + st.nextToken(","));**

 **}**

**}**

**Output:Next token is : Methodist**

Java Scanner

Scanner class in Java is found in the java.util package. Java provides various ways to read input from the keyboard, the java.util.Scanner class is one of them.

The Java Scanner class breaks the input into tokens using a delimiter which is whitespace by default. It provides many methods to read and parse various primitive values.

The Java Scanner class is widely used to parse text for strings and primitive types using a regular expression. It is the simplest way to get input in Java. By the help of Scanner in Java, we can get input from the user in primitive types such as int, long, double, byte, float, short, etc.

The Java Scanner class extends Object class and implements Iterator and Closeable interfaces.

The Java Scanner class provides nextXXX() methods to return the type of value such as nextInt(), nextByte(), nextShort(), next(), nextLine(), nextDouble(), nextFloat(), nextBoolean(), etc. To get a single character from the scanner, you can call next().charAt(0) method which returns a single character.

## How to get Java Scanner

To get the instance of Java Scanner which reads input from the user, we need to pass the input stream (System.in) in the constructor of Scanner class. For Example:

Scanner in = **new** Scanner(System.in);

To get the instance of Java Scanner which parses the strings, we need to pass the strings in the constructor of Scanner class. For Example:

Scanner in = **new** Scanner("Hello Java");

## Java Scanner Class Methods

The following are the list of Scanner methods:

|  |  |  |  |
| --- | --- | --- | --- |
| SN | Modifier & Type | Method | Description |
| 1) | void | [close()](https://www.javatpoint.com/post/java-scanner-close-method) | It is used to close this scanner. |
| 2) | pattern | [delimiter()](https://www.javatpoint.com/post/java-scanner-delimiter-method) | It is used to get the Pattern which the Scanner class is currently using to match delimiters. |
| 3) | Stream<MatchResult> | findAll() | It is used to find a stream of match results that match the provided pattern string. |
| 4) | String | [findInLine()](https://www.javatpoint.com/post/java-scanner-findinline-method) | It is used to find the next occurrence of a pattern constructed from the specified string, ignoring delimiters. |
| 5) | string | [findWithinHorizon()](https://www.javatpoint.com/post/java-scanner-findwithinhorizon-method) | It is used to find the next occurrence of a pattern constructed from the specified string, ignoring delimiters. |
| 6) | boolean | [hasNext()](https://www.javatpoint.com/post/java-scanner-hasnext-method) | It returns true if this scanner has another token in its input. |
| 7) | boolean | [hasNextBigDecimal()](https://www.javatpoint.com/post/java-scanner-hasnextbigdecimal-method) | It is used to check if the next token in this scanner's input can be interpreted as a BigDecimal using the nextBigDecimal() method or not. |
| 8) | boolean | [hasNextBigInteger()](https://www.javatpoint.com/post/java-scanner-hasnextbiginteger-method) | It is used to check if the next token in this scanner's input can be interpreted as a BigDecimal using the nextBigDecimal() method or not. |
| 9) | boolean | [hasNextBoolean()](https://www.javatpoint.com/post/java-scanner-hasnextboolean-method) | It is used to check if the next token in this scanner's input can be interpreted as a Boolean using the nextBoolean() method or not. |
| 10) | boolean | [hasNextByte()](https://www.javatpoint.com/post/java-scanner-hasnextbyte-method) | It is used to check if the next token in this scanner's input can be interpreted as a Byte using the nextBigDecimal() method or not. |
| 11) | boolean | [hasNextDouble()](https://www.javatpoint.com/post/java-scanner-hasnextdouble-method) | It is used to check if the next token in this scanner's input can be interpreted as a BigDecimal using the nextByte() method or not. |
| 12) | boolean | [hasNextFloat()](https://www.javatpoint.com/post/java-scanner-hasnextfloat-method) | It is used to check if the next token in this scanner's input can be interpreted as a Float using the nextFloat() method or not. |
| 13) | boolean | [hasNextInt()](https://www.javatpoint.com/post/java-scanner-hasnextint-method) | It is used to check if the next token in this scanner's input can be interpreted as an int using the nextInt() method or not. |
| 14) | boolean | [hasNextLine()](https://www.javatpoint.com/post/java-scanner-hasnextline-method) | It is used to check if there is another line in the input of this scanner or not. |
| 15) | boolean | [hasNextLong()](https://www.javatpoint.com/post/java-scanner-hasnextlong-method) | It is used to check if the next token in this scanner's input can be interpreted as a Long using the nextLong() method or not. |
| 16) | boolean | [hasNextShort()](https://www.javatpoint.com/post/java-scanner-hasnextshort-method) | It is used to check if the next token in this scanner's input can be interpreted as a Short using the nextShort() method or not. |
| 17) | IOException | [ioException()](https://www.javatpoint.com/post/java-scanner-ioexception-method) | It is used to get the IOException last thrown by this Scanner's readable. |
| 18) | Locale | [locale()](https://www.javatpoint.com/post/java-scanner-locale-method) | It is used to get a Locale of the Scanner class. |
| 19) | MatchResult | [match()](https://www.javatpoint.com/post/java-scanner-match-method) | It is used to get the match result of the last scanning operation performed by this scanner. |
| 20) | String | [next()](https://www.javatpoint.com/post/java-scanner-next-method) | It is used to get the next complete token from the scanner which is in use. |
| 21) | BigDecimal | [nextBigDecimal()](https://www.javatpoint.com/post/java-scanner-nextbigdecimal-method) | It scans the next token of the input as a BigDecimal. |
| 22) | BigInteger | [nextBigInteger()](https://www.javatpoint.com/post/java-scanner-nextbiginteger-method) | It scans the next token of the input as a BigInteger. |
| 23) | boolean | [nextBoolean()](https://www.javatpoint.com/post/java-scanner-nextboolean-method) | It scans the next token of the input into a boolean value and returns that value. |
| 24) | byte | [nextByte()](https://www.javatpoint.com/post/java-scanner-nextbyte-method) | It scans the next token of the input as a byte. |
| 25) | double | [nextDouble()](https://www.javatpoint.com/post/java-scanner-nextdouble-method) | It scans the next token of the input as a double. |
| 26) | float | [nextFloat()](https://www.javatpoint.com/post/java-scanner-nextfloat-method) | It scans the next token of the input as a float. |
| 27) | int | [nextInt()](https://www.javatpoint.com/post/java-scanner-nextint-method) | It scans the next token of the input as an Int. |
| 28) | String | [nextLine()](https://www.javatpoint.com/post/java-scanner-nextline-method) | It is used to get the input string that was skipped of the Scanner object. |
| 29) | long | [nextLong()](https://www.javatpoint.com/post/java-scanner-nextlong-method) | It scans the next token of the input as a long. |
| 30) | short | [nextShort()](https://www.javatpoint.com/post/java-scanner-nextshort-method) | It scans the next token of the input as a short. |
| 31) | int | [radix()](https://www.javatpoint.com/post/java-scanner-radix-method) | It is used to get the default radix of the Scanner use. |
| 32) | void | [remove()](https://www.javatpoint.com/post/java-scanner-remove-method) | It is used when remove operation is not supported by this implementation of Iterator. |
| 33) | Scanner | [reset()](https://www.javatpoint.com/post/java-scanner-reset-method) | It is used to reset the Scanner which is in use. |
| 34) | Scanner | [skip()](https://www.javatpoint.com/post/java-scanner-skip-method) | It skips input that matches the specified pattern, ignoring delimiters |
| 35) | Stream<String> | [tokens()](https://www.javatpoint.com/post/java-scanner-tokens-method) | It is used to get a stream of delimiter-separated tokens from the Scanner object which is in use. |
| 36) | String | [toString()](https://www.javatpoint.com/post/java-scanner-tostring-method) | It is used to get the string representation of Scanner using. |
| 37) | Scanner | [useDelimiter()](https://www.javatpoint.com/post/java-scanner-usedelimiter-method) | It is used to set the delimiting pattern of the Scanner which is in use to the specified pattern. |
| 38) | Scanner | [useLocale()](https://www.javatpoint.com/post/java-scanner-uselocale-method) | It is used to sets this scanner's locale object to the specified locale. |
| 39) | Scanner | [useRadix()](https://www.javatpoint.com/post/java-scanner-useradix-method) | It is used to set the default radix of the Scanner which is in use to the specified radix. |

## Example 1

Let's see a simple example of Java Scanner where we are getting a single input from the user. Here, we are asking for a string through in.nextLine() method.

**import** java.util.\*;

**public** **class** ScannerExample {

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

          Scanner in = **new** Scanner(System.in);

          System.out.print("Enter your name: ");

          String name = in.nextLine();

          System.out.println("Name is: " + name);

          in.close();

          }

}

**Example 2**

**import** java.util.\*;

**public** **class** ScannerClassExample1 {

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

          String s = "Hello, This is JavaTpoint.";

          //Create scanner Object and pass string in it

          Scanner scan = **new** Scanner(s);

          //Check if the scanner has a token

          System.out.println("Boolean Result: " + scan.hasNext());

          //Print the string

          System.out.println("String: " +scan.nextLine());

          scan.close();

          System.out.println("--------Enter Your Details-------- ");

          Scanner in = **new** Scanner(System.in);

          System.out.print("Enter your name: ");

          String name = in.next();

          System.out.println("Name: " + name);

          System.out.print("Enter your age: ");

          **int** i = in.nextInt();

          System.out.println("Age: " + i);

          System.out.print("Enter your salary: ");

          **double** d = in.nextDouble();

          System.out.println("Salary: " + d);

          in.close();

          }

}

Example 3

**import** java.util.\*;

**public** **class** ScannerClassExample2 {

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

     String str = "Helloworld/Java is Programming Language/Java is Pure Object oriented language .";

          //Create scanner with the specified String Object

          Scanner scanner = **new** Scanner(str);

System.out.println("Boolean Result: "+scanner.hasNextBoolean());

          //Change the delimiter of this scanner

          scanner.useDelimiter("/");

          //Printing the tokenized Strings

          System.out.println("---Tokenizes String---");

        **while**(scanner.hasNext()){

            System.out.println(scanner.next());

        }

          //Display the new delimiter

  System.out.println("Delimiter used: " +scanner.delimiter());                      scanner.close();

          }

}

Lab Program

**Java program that reads a line of integers, and displays each integer, and the sum of all the integers ( uses StringTokenizer class of java.util)**

import java.util.\*;

class Sumofintergers {

    public static void main(String args[]) {

        int n;

        int sum = 0;

        Scanner sc = new Scanner(System.in);

System.out.println("Enter integers with one space gap:");

        String s = sc.nextLine();

   StringTokenizer st = new StringTokenizer(s," ");

        while (st.hasMoreTokens()) {

            String temp = st.nextToken();

            n = Integer.parseInt(temp);

            System.out.println(n);

            sum = sum + n;

        }

        System.out.println("sum of the integers is: " + sum);

        sc.close();

    }

}