| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/StreamTokenizer.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/io/StreamCorruptedException.html)   [**NEXT CLASS**](http://docs.google.com/java/io/StringBufferInputStream.html) | [**FRAMES**](http://docs.google.com/index.html?java/io/StreamTokenizer.html)    [**NO FRAMES**](http://docs.google.com/StreamTokenizer.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#1ksv4uv) | [METHOD](#z337ya) |

## **java.io**

Class StreamTokenizer

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **java.io.StreamTokenizer**

public class **StreamTokenizer**extends [Object](http://docs.google.com/java/lang/Object.html)

The StreamTokenizer class takes an input stream and parses it into "tokens", allowing the tokens to be read one at a time. The parsing process is controlled by a table and a number of flags that can be set to various states. The stream tokenizer can recognize identifiers, numbers, quoted strings, and various comment styles.

Each byte read from the input stream is regarded as a character in the range '\u0000' through '\u00FF'. The character value is used to look up five possible attributes of the character: *white space*, *alphabetic*, *numeric*, *string quote*, and *comment character*. Each character can have zero or more of these attributes.

In addition, an instance has four flags. These flags indicate:

* Whether line terminators are to be returned as tokens or treated as white space that merely separates tokens.
* Whether C-style comments are to be recognized and skipped.
* Whether C++-style comments are to be recognized and skipped.
* Whether the characters of identifiers are converted to lowercase.

A typical application first constructs an instance of this class, sets up the syntax tables, and then repeatedly loops calling the nextToken method in each iteration of the loop until it returns the value TT\_EOF.

**Since:** JDK1.0 **See Also:**[nextToken()](http://docs.google.com/java/io/StreamTokenizer.html#nextToken()), [TT\_EOF](http://docs.google.com/java/io/StreamTokenizer.html#TT_EOF)

| **Field Summary** | |
| --- | --- |
| double | [**nval**](http://docs.google.com/java/io/StreamTokenizer.html#nval)            If the current token is a number, this field contains the value of that number. |
| [String](http://docs.google.com/java/lang/String.html) | [**sval**](http://docs.google.com/java/io/StreamTokenizer.html#sval)            If the current token is a word token, this field contains a string giving the characters of the word token. |
| static int | [**TT\_EOF**](http://docs.google.com/java/io/StreamTokenizer.html#TT_EOF)            A constant indicating that the end of the stream has been read. |
| static int | [**TT\_EOL**](http://docs.google.com/java/io/StreamTokenizer.html#TT_EOL)            A constant indicating that the end of the line has been read. |
| static int | [**TT\_NUMBER**](http://docs.google.com/java/io/StreamTokenizer.html#TT_NUMBER)            A constant indicating that a number token has been read. |
| static int | [**TT\_WORD**](http://docs.google.com/java/io/StreamTokenizer.html#TT_WORD)            A constant indicating that a word token has been read. |
| int | [**ttype**](http://docs.google.com/java/io/StreamTokenizer.html#ttype)            After a call to the nextToken method, this field contains the type of the token just read. |

| **Constructor Summary** | |
| --- | --- |
| [**StreamTokenizer**](http://docs.google.com/java/io/StreamTokenizer.html#StreamTokenizer(java.io.InputStream))([InputStream](http://docs.google.com/java/io/InputStream.html) is)  **Deprecated.** *As of JDK version 1.1, the preferred way to tokenize an input stream is to convert it into a character stream, for example:*  *Reader r = new BufferedReader(new InputStreamReader(is));  StreamTokenizer st = new StreamTokenizer(r);* |
| [**StreamTokenizer**](http://docs.google.com/java/io/StreamTokenizer.html#StreamTokenizer(java.io.Reader))([Reader](http://docs.google.com/java/io/Reader.html) r)            Create a tokenizer that parses the given character stream. |

| **Method Summary** | |
| --- | --- |
| void | [**commentChar**](http://docs.google.com/java/io/StreamTokenizer.html#commentChar(int))(int ch)            Specified that the character argument starts a single-line comment. |
| void | [**eolIsSignificant**](http://docs.google.com/java/io/StreamTokenizer.html#eolIsSignificant(boolean))(boolean flag)            Determines whether or not ends of line are treated as tokens. |
| int | [**lineno**](http://docs.google.com/java/io/StreamTokenizer.html#lineno())()            Return the current line number. |
| void | [**lowerCaseMode**](http://docs.google.com/java/io/StreamTokenizer.html#lowerCaseMode(boolean))(boolean fl)            Determines whether or not word token are automatically lowercased. |
| int | [**nextToken**](http://docs.google.com/java/io/StreamTokenizer.html#nextToken())()            Parses the next token from the input stream of this tokenizer. |
| void | [**ordinaryChar**](http://docs.google.com/java/io/StreamTokenizer.html#ordinaryChar(int))(int ch)            Specifies that the character argument is "ordinary" in this tokenizer. |
| void | [**ordinaryChars**](http://docs.google.com/java/io/StreamTokenizer.html#ordinaryChars(int,%20int))(int low, int hi)            Specifies that all characters *c* in the range low <= *c* <= high are "ordinary" in this tokenizer. |
| void | [**parseNumbers**](http://docs.google.com/java/io/StreamTokenizer.html#parseNumbers())()            Specifies that numbers should be parsed by this tokenizer. |
| void | [**pushBack**](http://docs.google.com/java/io/StreamTokenizer.html#pushBack())()            Causes the next call to the nextToken method of this tokenizer to return the current value in the ttype field, and not to modify the value in the nval or sval field. |
| void | [**quoteChar**](http://docs.google.com/java/io/StreamTokenizer.html#quoteChar(int))(int ch)            Specifies that matching pairs of this character delimit string constants in this tokenizer. |
| void | [**resetSyntax**](http://docs.google.com/java/io/StreamTokenizer.html#resetSyntax())()            Resets this tokenizer's syntax table so that all characters are "ordinary." See the ordinaryChar method for more information on a character being ordinary. |
| void | [**slashSlashComments**](http://docs.google.com/java/io/StreamTokenizer.html#slashSlashComments(boolean))(boolean flag)            Determines whether or not the tokenizer recognizes C++-style comments. |
| void | [**slashStarComments**](http://docs.google.com/java/io/StreamTokenizer.html#slashStarComments(boolean))(boolean flag)            Determines whether or not the tokenizer recognizes C-style comments. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/io/StreamTokenizer.html#toString())()            Returns the string representation of the current stream token and the line number it occurs on. |
| void | [**whitespaceChars**](http://docs.google.com/java/io/StreamTokenizer.html#whitespaceChars(int,%20int))(int low, int hi)            Specifies that all characters *c* in the range low <= *c* <= high are white space characters. |
| void | [**wordChars**](http://docs.google.com/java/io/StreamTokenizer.html#wordChars(int,%20int))(int low, int hi)            Specifies that all characters *c* in the range low <= *c* <= high are word constituents. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Field Detail** |
| --- |

### ttype

public int **ttype**

After a call to the nextToken method, this field contains the type of the token just read. For a single character token, its value is the single character, converted to an integer. For a quoted string token, its value is the quote character. Otherwise, its value is one of the following:

* TT\_WORD indicates that the token is a word.
* TT\_NUMBER indicates that the token is a number.
* TT\_EOL indicates that the end of line has been read. The field can only have this value if the eolIsSignificant method has been called with the argument true.
* TT\_EOF indicates that the end of the input stream has been reached.

The initial value of this field is -4.

**See Also:**[eolIsSignificant(boolean)](http://docs.google.com/java/io/StreamTokenizer.html#eolIsSignificant(boolean)), [nextToken()](http://docs.google.com/java/io/StreamTokenizer.html#nextToken()), [quoteChar(int)](http://docs.google.com/java/io/StreamTokenizer.html#quoteChar(int)), [TT\_EOF](http://docs.google.com/java/io/StreamTokenizer.html#TT_EOF), [TT\_EOL](http://docs.google.com/java/io/StreamTokenizer.html#TT_EOL), [TT\_NUMBER](http://docs.google.com/java/io/StreamTokenizer.html#TT_NUMBER), [TT\_WORD](http://docs.google.com/java/io/StreamTokenizer.html#TT_WORD)

### TT\_EOF

public static final int **TT\_EOF**

A constant indicating that the end of the stream has been read.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.io.StreamTokenizer.TT_EOF)

### TT\_EOL

public static final int **TT\_EOL**

A constant indicating that the end of the line has been read.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.io.StreamTokenizer.TT_EOL)

### TT\_NUMBER

public static final int **TT\_NUMBER**

A constant indicating that a number token has been read.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.io.StreamTokenizer.TT_NUMBER)

### TT\_WORD

public static final int **TT\_WORD**

A constant indicating that a word token has been read.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.io.StreamTokenizer.TT_WORD)

### sval

public [String](http://docs.google.com/java/lang/String.html) **sval**

If the current token is a word token, this field contains a string giving the characters of the word token. When the current token is a quoted string token, this field contains the body of the string.

The current token is a word when the value of the ttype field is TT\_WORD. The current token is a quoted string token when the value of the ttype field is a quote character.

The initial value of this field is null.

**See Also:**[quoteChar(int)](http://docs.google.com/java/io/StreamTokenizer.html#quoteChar(int)), [TT\_WORD](http://docs.google.com/java/io/StreamTokenizer.html#TT_WORD), [ttype](http://docs.google.com/java/io/StreamTokenizer.html#ttype)

### nval

public double **nval**

If the current token is a number, this field contains the value of that number. The current token is a number when the value of the ttype field is TT\_NUMBER.

The initial value of this field is 0.0.

**See Also:**[TT\_NUMBER](http://docs.google.com/java/io/StreamTokenizer.html#TT_NUMBER), [ttype](http://docs.google.com/java/io/StreamTokenizer.html#ttype)

| **Constructor Detail** |
| --- |

### StreamTokenizer

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public **StreamTokenizer**([InputStream](http://docs.google.com/java/io/InputStream.html) is)

**Deprecated.** *As of JDK version 1.1, the preferred way to tokenize an input stream is to convert it into a character stream, for example:*

*Reader r = new BufferedReader(new InputStreamReader(is));  
 StreamTokenizer st = new StreamTokenizer(r);*

Creates a stream tokenizer that parses the specified input stream. The stream tokenizer is initialized to the following default state:

* All byte values 'A' through 'Z', 'a' through 'z', and '\u00A0' through '\u00FF' are considered to be alphabetic.
* All byte values '\u0000' through '\u0020' are considered to be white space.
* '/' is a comment character.
* Single quote '\'' and double quote '"' are string quote characters.
* Numbers are parsed.
* Ends of lines are treated as white space, not as separate tokens.
* C-style and C++-style comments are not recognized.

**Parameters:**is - an input stream.**See Also:**[BufferedReader](http://docs.google.com/java/io/BufferedReader.html), [InputStreamReader](http://docs.google.com/java/io/InputStreamReader.html), [StreamTokenizer(java.io.Reader)](http://docs.google.com/java/io/StreamTokenizer.html#StreamTokenizer(java.io.Reader))

### StreamTokenizer

public **StreamTokenizer**([Reader](http://docs.google.com/java/io/Reader.html) r)

Create a tokenizer that parses the given character stream.

**Parameters:**r - a Reader object providing the input stream.**Since:** JDK1.1

| **Method Detail** |
| --- |

### resetSyntax

public void **resetSyntax**()

Resets this tokenizer's syntax table so that all characters are "ordinary." See the ordinaryChar method for more information on a character being ordinary.

**See Also:**[ordinaryChar(int)](http://docs.google.com/java/io/StreamTokenizer.html#ordinaryChar(int))

### wordChars

public void **wordChars**(int low,  
 int hi)

Specifies that all characters *c* in the range low <= *c* <= high are word constituents. A word token consists of a word constituent followed by zero or more word constituents or number constituents.

**Parameters:**low - the low end of the range.hi - the high end of the range.

### whitespaceChars

public void **whitespaceChars**(int low,  
 int hi)

Specifies that all characters *c* in the range low <= *c* <= high are white space characters. White space characters serve only to separate tokens in the input stream.

Any other attribute settings for the characters in the specified range are cleared.

**Parameters:**low - the low end of the range.hi - the high end of the range.

### ordinaryChars

public void **ordinaryChars**(int low,  
 int hi)

Specifies that all characters *c* in the range low <= *c* <= high are "ordinary" in this tokenizer. See the ordinaryChar method for more information on a character being ordinary.

**Parameters:**low - the low end of the range.hi - the high end of the range.**See Also:**[ordinaryChar(int)](http://docs.google.com/java/io/StreamTokenizer.html#ordinaryChar(int))

### ordinaryChar

public void **ordinaryChar**(int ch)

Specifies that the character argument is "ordinary" in this tokenizer. It removes any special significance the character has as a comment character, word component, string delimiter, white space, or number character. When such a character is encountered by the parser, the parser treats it as a single-character token and sets ttype field to the character value.

Making a line terminator character "ordinary" may interfere with the ability of a StreamTokenizer to count lines. The lineno method may no longer reflect the presence of such terminator characters in its line count.

**Parameters:**ch - the character.**See Also:**[ttype](http://docs.google.com/java/io/StreamTokenizer.html#ttype)

### commentChar

public void **commentChar**(int ch)

Specified that the character argument starts a single-line comment. All characters from the comment character to the end of the line are ignored by this stream tokenizer.

Any other attribute settings for the specified character are cleared.

**Parameters:**ch - the character.

### quoteChar

public void **quoteChar**(int ch)

Specifies that matching pairs of this character delimit string constants in this tokenizer.

When the nextToken method encounters a string constant, the ttype field is set to the string delimiter and the sval field is set to the body of the string.

If a string quote character is encountered, then a string is recognized, consisting of all characters after (but not including) the string quote character, up to (but not including) the next occurrence of that same string quote character, or a line terminator, or end of file. The usual escape sequences such as "\n" and "\t" are recognized and converted to single characters as the string is parsed.

Any other attribute settings for the specified character are cleared.

**Parameters:**ch - the character.**See Also:**[nextToken()](http://docs.google.com/java/io/StreamTokenizer.html#nextToken()), [sval](http://docs.google.com/java/io/StreamTokenizer.html#sval), [ttype](http://docs.google.com/java/io/StreamTokenizer.html#ttype)

### parseNumbers

public void **parseNumbers**()

Specifies that numbers should be parsed by this tokenizer. The syntax table of this tokenizer is modified so that each of the twelve characters:

0 1 2 3 4 5 6 7 8 9 . -

has the "numeric" attribute.

When the parser encounters a word token that has the format of a double precision floating-point number, it treats the token as a number rather than a word, by setting the ttype field to the value TT\_NUMBER and putting the numeric value of the token into the nval field.

**See Also:**[nval](http://docs.google.com/java/io/StreamTokenizer.html#nval), [TT\_NUMBER](http://docs.google.com/java/io/StreamTokenizer.html#TT_NUMBER), [ttype](http://docs.google.com/java/io/StreamTokenizer.html#ttype)

### eolIsSignificant

public void **eolIsSignificant**(boolean flag)

Determines whether or not ends of line are treated as tokens. If the flag argument is true, this tokenizer treats end of lines as tokens; the nextToken method returns TT\_EOL and also sets the ttype field to this value when an end of line is read.

A line is a sequence of characters ending with either a carriage-return character ('\r') or a newline character ('\n'). In addition, a carriage-return character followed immediately by a newline character is treated as a single end-of-line token.

If the flag is false, end-of-line characters are treated as white space and serve only to separate tokens.

**Parameters:**flag - true indicates that end-of-line characters are separate tokens; false indicates that end-of-line characters are white space.**See Also:**[nextToken()](http://docs.google.com/java/io/StreamTokenizer.html#nextToken()), [ttype](http://docs.google.com/java/io/StreamTokenizer.html#ttype), [TT\_EOL](http://docs.google.com/java/io/StreamTokenizer.html#TT_EOL)

### slashStarComments

public void **slashStarComments**(boolean flag)

Determines whether or not the tokenizer recognizes C-style comments. If the flag argument is true, this stream tokenizer recognizes C-style comments. All text between successive occurrences of /\* and \*/ are discarded.

If the flag argument is false, then C-style comments are not treated specially.

**Parameters:**flag - true indicates to recognize and ignore C-style comments.

### slashSlashComments

public void **slashSlashComments**(boolean flag)

Determines whether or not the tokenizer recognizes C++-style comments. If the flag argument is true, this stream tokenizer recognizes C++-style comments. Any occurrence of two consecutive slash characters ('/') is treated as the beginning of a comment that extends to the end of the line.

If the flag argument is false, then C++-style comments are not treated specially.

**Parameters:**flag - true indicates to recognize and ignore C++-style comments.

### lowerCaseMode

public void **lowerCaseMode**(boolean fl)

Determines whether or not word token are automatically lowercased. If the flag argument is true, then the value in the sval field is lowercased whenever a word token is returned (the ttype field has the value TT\_WORD by the nextToken method of this tokenizer.

If the flag argument is false, then the sval field is not modified.

**Parameters:**fl - true indicates that all word tokens should be lowercased.**See Also:**[nextToken()](http://docs.google.com/java/io/StreamTokenizer.html#nextToken()), [ttype](http://docs.google.com/java/io/StreamTokenizer.html#ttype), [TT\_WORD](http://docs.google.com/java/io/StreamTokenizer.html#TT_WORD)

### nextToken

public int **nextToken**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Parses the next token from the input stream of this tokenizer. The type of the next token is returned in the ttype field. Additional information about the token may be in the nval field or the sval field of this tokenizer.

Typical clients of this class first set up the syntax tables and then sit in a loop calling nextToken to parse successive tokens until TT\_EOF is returned.

**Returns:**the value of the ttype field. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.**See Also:**[nval](http://docs.google.com/java/io/StreamTokenizer.html#nval), [sval](http://docs.google.com/java/io/StreamTokenizer.html#sval), [ttype](http://docs.google.com/java/io/StreamTokenizer.html#ttype)

### pushBack

public void **pushBack**()

Causes the next call to the nextToken method of this tokenizer to return the current value in the ttype field, and not to modify the value in the nval or sval field.

**See Also:**[nextToken()](http://docs.google.com/java/io/StreamTokenizer.html#nextToken()), [nval](http://docs.google.com/java/io/StreamTokenizer.html#nval), [sval](http://docs.google.com/java/io/StreamTokenizer.html#sval), [ttype](http://docs.google.com/java/io/StreamTokenizer.html#ttype)

### lineno

public int **lineno**()

Return the current line number.

**Returns:**the current line number of this stream tokenizer.

### toString

public [String](http://docs.google.com/java/lang/String.html) **toString**()

Returns the string representation of the current stream token and the line number it occurs on.

The precise string returned is unspecified, although the following example can be considered typical:

Token['a'], line 10

**Overrides:**[toString](http://docs.google.com/java/lang/Object.html#toString()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a string representation of the token**See Also:**[nval](http://docs.google.com/java/io/StreamTokenizer.html#nval), [sval](http://docs.google.com/java/io/StreamTokenizer.html#sval), [ttype](http://docs.google.com/java/io/StreamTokenizer.html#ttype)

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/StreamTokenizer.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/io/StreamCorruptedException.html)   [**NEXT CLASS**](http://docs.google.com/java/io/StringBufferInputStream.html) | [**FRAMES**](http://docs.google.com/index.html?java/io/StreamTokenizer.html)    [**NO FRAMES**](http://docs.google.com/StreamTokenizer.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#1ksv4uv) | [METHOD](#z337ya) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).