
Class java.io.StreamTokenizer

```
java.lang.Object
   |
   +----java.io.StreamTokenizer
```

---

public class **StreamTokenizer**
extends Object

A class to turn an input stream into a stream of tokens. There are a number of methods that define the lexical syntax of tokens.

Version:
1.10, 08/15/95
Author:
James Gosling

Variable Index

o **TT_EOF**
The End–of–file token.
o **TT_EOL**
The End–of–line token.
o **TT_NUMBER**
The number token.
o **TT_WORD**
The word token.
o **nval**
The number value.
o **sval**
The Stream value.
o **ttype**
The type of the last token returned.

Constructor Index

o **StreamTokenizer**(InputStream)
    Creates a stream tokenizer that parses the specified input stream.

Method Index

o **commentChar**(int)
    Specifies that this character starts a single line comment.
o **eolIsSignificant**(boolean)
    If the flag is true, end–of–lines are significant (TT_EOL will be returned by nexttoken).
o **lineno**()
    Return the current line number.
o **lowerCaseMode**(boolean)
    Examines a boolean to decide whether TT_WORD tokens are forced to be lower case.
o **nextToken**()
    Parses a token from the input stream.
o **ordinaryChar**(int)
    Specifies that this character is 'ordinary': it removes any significance as a word, comment, string, whitespace or number character.
o **ordinaryChars**(int, int)
    Specifies that characters in this range are 'ordinary'.
o **parseNumbers**()
    Specifies that numbers should be parsed.
o **pushBack**()
    Pushes back a stream token.
o **quoteChar**(int)
    Specifies that matching pairs of this character delimit String constants.
o **resetSyntax**()
    Resets the syntax table so that all characters are special.
o **slashSlashComments**(boolean)
    If the flag is true, recognize C++ style(//) comments.
o **slashStarComments**(boolean)
    If the flag is true, recognize C style(/*) comments.
o **toString**()
    Returns the String representation of the stream token.
o **whitespaceChars**(int, int)
    Specifies that characters in this range are whitespace characters.
o **wordChars**(int, int)
    Specifies that characters in this range are word characters.

Variables

o **ttype**

```
public int ttype
```

The type of the last token returned. It's value will either be one of the following TT_* constants, or a single character. For example, if '+' is encountered and is not a valid word character, ttype will be '+'

- **TT_EOF**

```
public final static int TT_EOF
```

The End–of–file token.

- **TT_EOL**

```
public final static int TT_EOL
```

The End–of–line token.

- **TT_NUMBER**

```
public final static int TT_NUMBER
```

The number token. This value is in nval.

- **TT_WORD**

```
public final static int TT_WORD
```

The word token. This value is in sval.

- **sval**

```
public String sval
```

The Stream value.

- **nval**

```
public double nval
```

The number value.

Constructors

- **StreamTokenizer**

```
public StreamTokenizer(InputStream I)
```

Creates a stream tokenizer that parses the specified input stream. By default, it recognizes numbers, Strings quoted with single and double quotes, and all the

alphabetics.

Parameters:

I – the input stream

Methods

o **resetSyntax**

```
public void resetSyntax()
```

Resets the syntax table so that all characters are special.

o **wordChars**

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

Specifies that characters in this range are word characters.

Parameters:

low – the low end of the range

hi – the high end of the range

o **whitespaceChars**

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

Specifies that characters in this range are whitespace characters.

Parameters:

low – the low end of the range

hi – the high end of the range

o **ordinaryChars**

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

Specifies that characters in this range are 'ordinary'. Ordinary characters mean that any significance as words, comments, strings, whitespaces or number characters are removed. When these characters are encountered by the parser, they return a ttype equal to the character.

Parameters:

low – the low end of the range

hi – the high end of the range

o **ordinaryChar**

```
public void ordinaryChar(int ch)
```

Specifies that this character is 'ordinary': it removes any significance as a word, comment, string, whitespace or number character. When encountered by the parser, it returns a ttype equal to the character.

Parameters:

ch – the character

- **commentChar**

```
public void commentChar(int ch)
```

Specifies that this character starts a single line comment.

Parameters:

ch – the character

- **quoteChar**

```
public void quoteChar(int ch)
```

Specifies that matching pairs of this character delimit String constants. When a String constant is recognized, ttype will be the character that delimits the String, and sval will have the body of the String.

Parameters:

ch – the character

- **parseNumbers**

```
public void parseNumbers()
```

Specifies that numbers should be parsed. This method accepts double precision floating point numbers and returns a ttype of TT_NUMBER with the value in nval.

- **eolIsSignificant**

```
public void eolIsSignificant(boolean flag)
```

If the flag is true, end–of–lines are significant (TT_EOL will be returned by nexttoken). If false, they will be treated as whitespace.

- **slashStarComments**

```
public void slashStarComments(boolean flag)
```

If the flag is true, recognize C style(/*) comments.

- **slashSlashComments**

```
public void slashSlashComments(boolean flag)
```

If the flag is true, recognize C++ style(//) comments.

- **lowerCaseMode**

```
public void lowerCaseMode(boolean fl)
```

Examines a boolean to decide whether TT_WORD tokens are forced to be lower case.

Parameters:

fl – the boolean flag

- **nextToken**

```
public int nextToken() throws IOException
```

Parses a token from the input stream. The return value is the same as the value of ttype. 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.

- **pushBack**

```
public void pushBack()
```

Pushes back a stream token.

- **lineno**

```
public int lineno()
```

Return the current line number.

- **toString**

```
public String toString()
```

Returns the String representation of the stream token.

Overrides:

toString in class Object
