Java StreamTokenizer

Java.io.StreamTokenizer class parses input stream into “tokens”.It allows to read one token at a time. Stream Tokenizer can recognize numbers, quoted strings, and various comment styles.

Declaration:

public class StreamTokenizer extends Object

Field:

Following are the fields for Java.io.StreamTokenizer class −

double nval − If the current token is a number, this field contains the value of that number.

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

static int TT_EOL − A constant indicating that the end of the line has been read.

static int TT_NUMBER − A constant indicating that a number token has been read.

static int TT_WORD − A constant indicating that a word token has been read.

int ttype − After a call to the nextToken method, this field contains the type of the token just read.

Constructor :

StreamTokenizer(Reader arg) : Creates a tokenizer that parses the given character stream.

Methods :

Sr.No.	Method & Description
1	void commentChar(int ch) Specified that the character argument starts a single-line comment.
2	void eolIsSignificant(boolean flag) This method determines whether or not ends of line are treated as tokens.
3	int lineno() This method returns the current line number.
4	void lowerCaseMode(boolean fl) This method determines whether or not word token are automatically lowercased.
5	int nextToken() This method parses the next token from the input stream of this tokenizer.
6	void ordinaryChar(int ch) This method specifies that the character argument is "ordinary" in this tokenizer.
7	void ordinaryChars(int low, int hi) This method specifies that all characters c in the range low <= c <= high are "ordinary" in this tokenizer.
8	void parseNumbers() This method specifies that numbers should be parsed by this tokenizer.
9	void pushBack() This method 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.

10	void quoteChar(int ch) This method specifies that matching pairs of this character delimit string constants in this tokenizer.
11	void resetSyntax() This method resets this tokenizer's syntax table so that all characters are "ordinary." See the ordinaryChar method for more information on a character being ordinary.
12	void slashSlashComments(boolean flag) This method determines whether or not the tokenizer recognizes C++ style comments.
13	void slashStarComments(boolean flag) This method determines whether or not the tokenizer recognizes C style comments.
14	String toString() This method returns the string representation of the current stream token and the line number it occurs on.
15	void whitespaceChars(int low, int hi) This method specifies that all characters c in the range low <= c <= high are white space characters.
16	void wordChars(int low, int hi) This method specifies that all characters c in the range low <= c >= high are word constituents.

Example 1 commentChar():

```
import java.io.*;
class CmtChar
{
	public static void main(String[] args) throws InterruptedException,
FileNotFoundException, IOException
	{
		FileReader reader = new FileReader("new.txt");
		BufferedReader bufferread = new BufferedReader(reader);
		StreamTokenizer token = new StreamTokenizer(bufferread);

		token.commentChar('a');
		int t;
		while ((t = token.nextToken()) != StreamTokenizer.TT_EOF)
		{
			switch (t)
			{
			case StreamTokenizer.TT_NUMBER:
				System.out.println("Number : " + token.nval);
				break;
			case StreamTokenizer.TT_WORD:
				System.out.println("Word : " + token.sval);
				break;
			}
		}
	}
}
```

Example 2 lineno():

```
import java.io.*;
class LineNum
{

	public static void main(String[] args) throws InterruptedException,
FileNotFoundException, IOException
	{
		FileReader reader = new FileReader("new.txt");
		BufferedReader bufferread = new BufferedReader(reader);
		StreamTokenizer token = new StreamTokenizer(bufferread);

		token.eolIsSignificant(true);
		//Use of lineno() method to get current line no.
		System.out.println("Line Number:" + token.lineno());

		int t;
		while ((t = token.nextToken()) != StreamTokenizer.TT_EOF)
		{
			switch (t)
			{
			case StreamTokenizer.TT_EOL:
				System.out.println("Line No. : " + token.lineno());
				break;
			case StreamTokenizer.TT_NUMBER:
				System.out.println("Number : " + token.nval);
				break;
			case StreamTokenizer.TT_WORD:
				System.out.println("Word : " + token.sval);
				break;
			}
		}
	}
}
```

Example 3 toString():

```
import java.io.*;

class ToString

{

	public static void main(String[] args) throws InterruptedException,
FileNotFoundException, IOException

	{

		FileReader reader = new FileReader("new.txt");

		BufferedReader bufferread = new BufferedReader(reader);

		StreamTokenizer token = new StreamTokenizer(bufferread);

		int t;

		while ((t = token.nextToken()) != StreamTokenizer.TT_EOF)

		{

			switch (t)

			{

			case StreamTokenizer.TT_NUMBER:

				// Value of ttype field returned by nextToken()

				System.out.println("Number : " + token.nval);

				break;

			case StreamTokenizer.TT_WORD:

				// Use of toStringn() method

				System.out.println("Word : " + token.toString());

				break;

			}

		}

	}

}
```

Example 4 ordinaryChar():

```
import java.io.*;
class OrdinaryChar
{
	public static void main(String[] args) throws InterruptedException,
FileNotFoundException, IOException
	{
		FileReader reader = new FileReader("new.txt");
		BufferedReader bufferread = new BufferedReader(reader);
		StreamTokenizer token = new StreamTokenizer(bufferread);

		// Use of ordinaryChar() method Here we have taken 'm' as an
ordinary character
		token.ordinaryChar('m');

		int t;
		while ((t = token.nextToken()) != StreamTokenizer.TT_EOF)
		{
			switch (t)
			{
			case StreamTokenizer.TT_NUMBER:
				System.out.println("Number : " + token.nval);
				break;
			case StreamTokenizer.TT_WORD:
				System.out.println("Word : " + token.sval);
				break;
			}
		}
	}
}
```

Example 5 wordChars():

```
import java.io.*;
class WordChars
{
	public static void main(String[] args) throws InterruptedException,
FileNotFoundException, IOException
	{
		FileReader reader = new FileReader("new.txt");
		StreamTokenizer token = new StreamTokenizer(reader);

		// Use of wordChars() method Here we have taken '_' as an
wordChars, it will be considered as word
		token.wordChars('_','_');
		int t;

		while((t = token.nextToken()) != StreamTokenizer.TT_EOF)
		{
			switch (t)
			{
			case StreamTokenizer.TT_NUMBER:
				System.out.println("Double : " + token.nval);
				break;
			case StreamTokenizer.TT_WORD:
				System.out.println("Word : " + token.sval);
				break;
			}
		}
	}
}
```

Example 5:

```
import java.io.*;
class Main
{
	public static void main(String[] args) throws InterruptedException,
FileNotFoundException, IOException
	{
		FileReader reader = new FileReader("new.txt");
		StreamTokenizer token = new StreamTokenizer(reader);

		int t;
		while((t = token.nextToken()) != StreamTokenizer.TT_EOF)
		{
			switch (t)
			{
			case StreamTokenizer.TT_NUMBER:
				if (token.nval % 1 == 0 )
					System.out.println("Integer : " +
(int)token.nval);
				else
					System.out.println("Double : " + token.nval);
				break;
			case StreamTokenizer.TT_WORD:
				String word = token.sval;
				if (word.equalsIgnoreCase("true") ||
word.equalsIgnoreCase("false"))
					System.out.println("Boolean : " + token.sval);
				else
					System.out.println("Word : " + token.sval);
				break;
			default:
				System.out.println("Other : "+ (char) token.ttype);
			}
		}
	}
}
```