**Java.io.StreamTokenizer Class in 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*

**Constructor :**

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

**Methods :**

***commentChar***: java.io.StreamTokenizer.commentChar(int arg) ignores all characters from the single-line comment character to the end of the line by this StreamTokenizer.

**Syntax :**

public void commentChar(int arg)

Parameters :

arg : the character after which all characters are ignored in the line.

Return :

No value is returned.

**Implementation :**

**// Java Program illustrating use of commentChar() method**

import java.io.\*;

public class NewClass

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

FileNotFoundException, IOException

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

BufferedReader bufferread = new BufferedReader(reader);

StreamTokenizer token = new StreamTokenizer(bufferread);

// Use of commentChar() method

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; } } } }

**Note :**

This program won’t run here as no ‘ABC’ file exists. You can check this code on Java compiler on your system.

To check this code, create a file ‘ABC’ on your system.

**‘ABC’ file contains :**

Programmers

1

2

3

Geeks

Hello

a Program is explained here my friends.

**Output:**

Word : Progr

Number : 1.0

Number : 2.0

Number : 3.0

Word : Geeks

Word : Hello

***lineno() :*** java.io.StreamTokenizer.lineno() returns the current line number of this StreamTokenizer.

**Syntax :**

public int lineno()

Parameters :

arg : the character after which all characters are ignored in the line.

Return :

returns the current line number of this StreamTokenizer.

**Implementation :**

**// Java Program illustrating use of lineno() method**

import java.io.\*;

public class NewClass

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

FileNotFoundException, IOException

{ FileReader reader = new FileReader("ABC.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());

token.commentChar('a');

int t;

while ((t = token.nextToken()) != StreamTokenizer.TT\_EOF)

{ switch (t)

{ case StreamTokenizer.TT\_EOL:

System.out.println("");

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; } } } }

**Output:**

Line Number:1

Word : Progr

Line No. : 2

Number : 1.0

Line No. : 3

Number : 2.0

Line No. : 4

Number : 3.0

Line No. : 5

Word : Geeks

Line No. : 6

Word : Hello

Line No. : 7

Word : This

Word : is

***toString() :*** java.io.StreamTokenizer.toString() represents current Stream token as a string along with it’s line no.

**Syntax :**

public String toString()

Return :

represents current Stream token as a string along with it's line no.

**Implementation :**

**// Java Program illustrating use of toString() method**

import java.io.\*;

public class NewClass

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

FileNotFoundException, IOException

{ FileReader reader = new FileReader("ABC.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;

// Value of ttype field returned by nextToken()

case StreamTokenizer.TT\_WORD:

// Use of toStringn() method

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

break; } } } }

**Output:**

Word : Token[Programmers], line 1

Number : 1.0

Number : 2.0

Number : 3.0

Word : Token[Geeks], line 5

Word : Token[Hello], line 6

Word : Token[a], line 7

Word : Token[Program], line 7

Word : Token[is], line 7

Word : Token[explained], line 7

Word : Token[here], line 7

Word : Token[my], line 7

Word : Token[friends.], line 7

***eolIsSignificant()*** : java.io.StreamTokenizer.eolIsSignificant(boolean arg) tells whether to treat end of line as a token or not. If ‘arg’ is true, then it End Of Line is treated as a token. If true, then the method returns TT\_EOL and set the ttype field when End Of Line is reached.

If ‘arg’ is false then the End Of Line is treated simply as a white space.

**Syntax :**

public void eolIsSignificant(boolean arg)

Parameters :

arg : boolean which tells whether to take EOL as a token or white space

Return :

No value is returned.

**Implementation :**

**// Java Program illustrating use of eolIsSignificant() method**

import java.io.\*;

public class NewClass

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

FileNotFoundException, IOException

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

BufferedReader bufferread = new BufferedReader(reader);

StreamTokenizer token = new StreamTokenizer(bufferread);

boolean arg = true;

// Use of eolIsSignificant() method

token.eolIsSignificant(arg);

// Here the 'arg' is set true, so EOL is treated as a token

int t;

while ((t = token.nextToken()) != StreamTokenizer.TT\_EOF)

{ switch (t)

{ case StreamTokenizer.TT\_EOL:

System.out.println("End of Line encountered.");

break;

case StreamTokenizer.TT\_NUMBER:

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

break;

case StreamTokenizer.TT\_WORD:

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

break; } } }}

**Note :**

This program won’t run here as no ‘ABC’ file exists. You can check this code on Java compiler on your system.

To check this code, create a file ‘ABC’ on your system.

**‘ABC’ file contains :**

1

Geeks

2

For

3

Geeks

**Output :**

Number : 1.0

End of Line encountered.

Word : Geeks

End of Line encountered.

Number : 2.0

End of Line encountered.

Word : For

End of Line encountered.

Number : 3.0

End of Line encountered.

Word : Geeks

***nextToken() :*** java.io.StreamTokenizer.nextToken() parses the next token from the Input Stream and returns it’s value to the ttype field along with the additional fields like ‘sval’, ‘nval’.

**Syntax :**

public int nextToken()

Parameters :

------

Return :

value to the ttype field

**Implementation :**

**// Java Program illustrating use of nextToken() method**

import java.io.\*;

public class NewClass

{

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

FileNotFoundException, IOException

{

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

BufferedReader bufferread = new BufferedReader(reader);

StreamTokenizer token = new StreamTokenizer(bufferread);

// Use of nextToken() method to parse Next Token from the Input Stream

int t = token.nextToken();

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;

}

}

}

}

**Note :**

This program won’t run here as no ‘ABC’ file exists. You can check this code on Java compiler on your system.

To check this code, create a file ‘ABC’ on your system.

**‘ABC’ file contains :**

1

This program tells

2

about use of

3

next token() method

**Output :**

Word : This

Word : program

Word : tells

Number : 2.0

Word : about

Word : use

Word : of

Number : 3.0

Word : next

Word : token

Word : method

***lowerCaseMode() :*** java.io.StreamTokenizer.lowerCaseMode(boolean arg) tells whether to lowercase the word tokens automatically or not. If the flag – ‘arg’ is set true, then the value of ‘sval’ field is lowered.

**Syntax :**

public void lowerCaseMode(boolean arg)

Parameters :

arg : indicates whether to lowercase the word tokens automatically or not

Return :

void

**Implementation :**

**// Java Program illustrating use of lowerCaseMode() method**

import java.io.\*;

public class NewClass

{

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

FileNotFoundException, IOException

{

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

BufferedReader bufferread = new BufferedReader(reader);

StreamTokenizer token = new StreamTokenizer(bufferread);

/\* Use of lowerCaseMode() method to

Here, the we have set the Lower Case Mode ON

\*/

boolean arg = true;

token.lowerCaseMode(arg);

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;

}

}

}

}

**Note :**

This program won’t run here as no ‘ABC’ file exists. You can check this code on Java compiler on your system.

To check this code, create a file ‘ABC’ on your system.

**‘ABC’ file contains :**

Hello Geeks

This Is About

LowerCaseMode()

**Output :**

Word : hello

Word : geeks

Word : this

Word : is

Word : about

Word : lowercasemode

***ordinaryChar() :*** java.io.StreamTokenizer.ordinaryChar(int arg) sets ‘arg’ character as an ordinary character. It will remove the arg character, if it has any significance as word, number, white space or comment Character.

Syntax :

public void ordinaryChar(int arg)

Parameters :

arg : the character which is to be set as an Ordinary Character

Return :

void

**Implementation :**

**// Java Program illustrating use of ordinaryChar() method**

import java.io.\*;

public class NewClass

{

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

FileNotFoundException, IOException

{

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

BufferedReader bufferread = new BufferedReader(reader);

StreamTokenizer token = new StreamTokenizer(bufferread);

// Use of ordinaryChar() method

// Here we have taken 's' as an ordinary character

token.ordinaryChar('s');

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;

}

}

}

}

**Note :**

This program won’t run here as no ‘ABC’ file exists. You can check this code on Java compiler on your system.

To check this code, create a file ‘ABC’ on your system.

**‘ABC’ file contains :**

Hello Geeks

Thissss Issszz About

ordinaryChar()

**This method has remove ‘s’ from the entire Stream**

**Output :**

Word : Hello

Word : Geek

Word : Thi

Word : I

Word : zz

Word : About

Word : ordinaryChar

**ordinaryChars() :** java.io.StreamTokenizer.ordinaryChars(int low, int high) sets character in the range – ‘a to b’ to Ordinary character in the StreamTokenizer

**Syntax :**

public void ordinaryChars(int low, int high)

Parameters :

low : lower limit of range

high : higher limit of range

Return :

void

**Implementation :**

**// Java Program illustrating use of ordinaryChars() method**

import java.io.\*;

public class NewClass

{

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

FileNotFoundException, IOException

{

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

BufferedReader bufferread = new BufferedReader(reader);

StreamTokenizer token = new StreamTokenizer(bufferread);

// Use of ordinaryChars() method

// Here we have taken low = 'a' and high = 'c'

token.ordinaryChars('a','c');

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;

}

}

}

}

**Note :**

This program won’t run here as no ‘ABC’ file exists. You can check this code on Java compiler on your system.

To check this code, create a file ‘ABC’ on your system.

**‘ABC’ file contains :**

Hello Geeks

This

is

about

ordinaryChars()

**Output :**

Word : Hello

Word : Geeks

Word : This

Word : is

Word : out

Word : ordin

Word : ryCh

Word : rs