

TokenMgrError.java

```
1  /* Generated By:JavaCC: Do not edit this line. TokenMgrError.java Version 3.0 */
2  package jminusminus;
3
4  public class TokenMgrError extends Error
5  {
6      /*
7       * Ordinals for various reasons why an Error of this type can be thrown.
8       */
9
10     /**
11      * Lexical error occurred.
12      */
13     static final int LEXICAL_ERROR = 0;
14
15     /**
16      * An attempt was made to create a second instance of a static token manager.
17      */
18     static final int STATIC_LEXER_ERROR = 1;
19
20     /**
21      * Tried to change to an invalid lexical state.
22      */
23     static final int INVALID_LEXICAL_STATE = 2;
24
25     /**
26      * Detected (and bailed out of) an infinite loop in the token manager.
27      */
28     static final int LOOP_DETECTED = 3;
29
30     /**
31      * Indicates the reason why the exception is thrown. It will have
32      * one of the above 4 values
33      */
34     int errorCode;
35
36     /**
37      * Replaces unprintable characters by their escaped (or unicode escaped)
38      * equivalents in the given string
39      */
40     protected static final String addEscapes(String str) {
41         StringBuffer retval = new StringBuffer();
42         char ch;
43         for (int i = 0; i < str.length(); i++) {
44             switch (str.charAt(i))
45             {
46                 case 0 :
47                     continue;
48                 case '\b':
49                     retval.append("\\b");
50                     continue;
51                 case '\t':
52                     retval.append("\\t");
53                     continue;
54                 case '\n':
55                     retval.append("\\n");
56                     continue;
57                 case '\f':
58                     retval.append("\\f");
59                     continue;
60                 case '\r':
61                     retval.append("\\r");
62                     continue;
63                 case '\"':
64                     retval.append("\\\"");
65                     continue;
66                 case '\\':
```

```

67         retval.append("\\\\");
68         continue;
69     case '\\':
70         retval.append("\\\\");
71         continue;
72     default:
73         if ((ch = str.charAt(i)) < 0x20 || ch > 0x7e) {
74             String s = "0000" + Integer.toString(ch, 16);
75             retval.append("\\u" + s.substring(s.length() - 4, s.length()));
76         } else {
77             retval.append(ch);
78         }
79         continue;
80     }
81 }
82 return retval.toString();
83 }
84
85 /**
86  * Returns a detailed message for the Error when it is thrown by the
87  * token manager to indicate a lexical error.
88  * Parameters :
89  *     EOFSeen      : indicates if EOF caused the lexical error
90  *     curLexState   : lexical state in which this error occurred
91  *     errorLine     : line number when the error occurred
92  *     errorColumn   : column number when the error occurred
93  *     errorAfter    : prefix that was seen before this error occurred
94  *     curchar       : the offending character
95  * Note: You can customize the lexical error message by modifying this method.
96  */
97 protected static String LexicalError(boolean EOFSeen, int lexState, int
errorLine, int errorColumn, String errorAfter, char curChar) {
98     return("Lexical error at line " +
99         errorLine + " / column " +
100         errorColumn + " : " + errorAfter + curChar + "\n");
101     (EOFSeen ? "<EOF> " : ("\" + addEscapes(String.valueOf(curChar)) +
102         "\"") + " (" + (int)curChar + "), ") +
103         "after : \"" + addEscapes(errorAfter) + "\"");
104 }
105
106 /**
107  * You can also modify the body of this method to customize your error
108  * messages.
109  * For example, cases like LOOP_DETECTED and INVALID_LEXICAL_STATE are not
110  * of end-users concern, so you can return something like :
111  *
112  *     "Internal Error : Please file a bug report .... "
113  *
114  * from this method for such cases in the release version of your parser.
115  */
116 public String getMessage() {
117     return super.getMessage();
118 }
119
120 /**
121  * Constructors of various flavors follow.
122  */
123 public TokenMgrError() {
124 }
125
126 public TokenMgrError(String message, int reason) {
127     super(message);
128     errorCode = reason;
129 }
130
131 public TokenMgrError(boolean EOFSeen, int lexState, int errorLine, int
errorColumn, String errorAfter, char curChar, int reason) {
132     this(LexicalError(EOFSeen, lexState, errorLine, errorColumn, errorAfter,

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```
curChar), reason);  
132     }  
133 }  
134
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder