TokenMgrError.java

```java
/* Generated By:JavaCC: Do not edit this line. TokenMgrError.java Version 3.0 */
package jminusminus;

public class TokenMgrError extends Error
{
   /*
    * Ordinals for various reasons why an Error of this type can be thrown.
    */

   /**
    * Lexical error occured.
    */
   static final int LEXICAL_ERROR = 0;

   /**
    * An attempt wass made to create a second instance of a static token manager.
    */
   static final int STATIC_LEXER_ERROR = 1;

   /**
    * Tried to change to an invalid lexical state.
    */
   static final int INVALID_LEXICAL_STATE = 2;

   /**
    * Detected (and bailed out of) an infinite loop in the token manager.
    */
   static final int LOOP_DETECTED = 3;

   /**
    * Indicates the reason why the exception is thrown. It will have
    * one of the above 4 values.
    */
   int errorCode;

   /**
    * Replaces unprintable characters by their espaced (or unicode escaped)
    * equivalents in the given string
    */
   protected static final String addEscapes(String str) {
      StringBuffer retval = new StringBuffer();
      char ch;
      for (int i = 0; i < str.length(); i++) {
        switch (str.charAt(i))
        {
           case 0 :
              continue;
           case '\b':
              retval.append("\\b");
              continue;
           case '\t':
              retval.append("\\t");
              continue;
           case '\n':
              retval.append("\\n");
              continue;
           case '\f':
              retval.append("\\f");
              continue;
           case '\r':
              retval.append("\\r");
              continue;
           case '\"':
              retval.append("\\\"");
              continue;
           case '\'':
              retval.append("\\\'");
              continue;
           case '\\':
              retval.append("\\\\");
              continue;
           default:
              if ((ch = str.charAt(i)) < 0x20 || ch > 0x7e) {
                 String s = "0000" + Integer.toString(ch, 16);
                 retval.append("\\u" + s.substring(s.length() - 4, s.length()));
              } else {
                 retval.append(ch);
              }
              continue;
        }
      }
      return retval.toString();
   }

   /**
    * Returns a detailed message for the Error when it is thrown by the
    * token manager to indicate a lexical error.
    * Parameters :
    *    EOFSeen     : indicates if EOF caused the lexicl error
    *    curLexState : lexical state in which this error occured
    *    errorLine   : line number when the error occured
    *    errorColumn : column number when the error occured
    *    errorAfter  : prefix that was seen before this error occured
    *    curchar     : the offending character
    * Note: You can customize the lexical error message by modifying this method.
    */
   protected static String LexicalError(boolean EOFSeen, int lexState, int
errorLine, int errorColumn, String errorAfter, char curChar) {
      return("Lexical error at line " +
           errorLine + ", column " +
           errorColumn + ".  Encountered: " +
           (EOFSeen ? "<EOF> " : ("\"" + addEscapes(String.valueOf(curChar)) +
"\"") + " (" + (int)curChar + "), ") +
           "after : \"" + addEscapes(errorAfter) + "\"");
   }

   /**
    * You can also modify the body of this method to customize your error
messages.
    * For example, cases like LOOP_DETECTED and INVALID_LEXICAL_STATE are not
    * of end-users concern, so you can return something like :
    *
    *     "Internal Error : Please file a bug report .... "
    *
    * from this method for such cases in the release version of your parser.
    */
   public String getMessage() {
      return super.getMessage();
   }

   /*
    * Constructors of various flavors follow.
    */

   public TokenMgrError() {
   }

   public TokenMgrError(String message, int reason) {
      super(message);
      errorCode = reason;
   }

   public TokenMgrError(boolean EOFSeen, int lexState, int errorLine, int
errorColumn, String errorAfter, char curChar, int reason) {
      this(LexicalError(EOFSeen, lexState, errorLine, errorColumn, errorAfter,
curChar), reason);
    }
}

```