```java
package LexicalAnalyzer;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.File;
import java.util.List;
import java.util.ArrayList;
/**
 * This is the IO module that receives input file
 * as well as receives errors from package
 * classes for the Lexical analyzer package
 *
 * @author  Derek Trom
 * @author Elena Corpus
 * @version 1.0
 * @since   2020-09-26
 */


public class IOModule {
    /**
     * Create variables for the input text
     * as well as an ArrayList structure for
     * the matches generated
     */
    private String programText = "";
    private List<Match> matches = new ArrayList<Match>();
    /**
     * <p>
     * This method is used to retrieve file and create a buffer
     * to be analyzed
     * @throws Exception error reading the file contents
     * @param fileName This is the input filename from the command line
     *
     */
    public IOModule (String fileName) throws Exception{

        /**
         * Try to construct and read from the input file
         * and throw an exception if it fails.
         */
        try {
            /**
             * Create File structure and a BufferedReader
             */
            File pascalInput = new File(fileName);
            BufferedReader rdr = new BufferedReader(new FileReader(pascalInput));
            String currentLine = "";
```

```
48
49              while (rdr.ready()) {
50                  currentLine = rdr.readLine();
51                  programText += currentLine+"\n";
52              }
53              programText = programText.trim();
54              rdr.close();
55          } catch (Exception e) {
56              System.err.println("There was a problem reading
    the file.");
57              System.err.println(e.getMessage());
58              System.exit(1);
59          }
60
61          /**
62           * Adds headers for the printable table in list
63           */
64          this.addMatch(new LexicalAnalyzer.Match("LEXEME","
   SPELLING"));
65          this.addMatch(new LexicalAnalyzer.Match("",""));
66      }
67      /**
68       * Used to return the text
69       * @return Returns the programText read from the input
   file
70       */
71      public String getProgramText() {
72          return programText;
73      }
74      /**
75       * Adds a symbol match to the matches list
76       * @param newMatch match to be added to the list
77       */
78      public void addMatch(Match newMatch) {
79          matches.add(newMatch);
80      }
81
82      /**
83       * Prints out the matches in the list
84       * generated from the program
85       */
86      public void printMatches() {
87          System.out.println();
88          for (Match m : matches) {
89              System.out.println(m);
90          }
91      }
92  }
93
```