```java
package LexicalAnalyzer;
/**
 * Language
 *
 * This is the class construction for all possible token
 * including regular expressions for the package
 *
 * @author  Derek Trom
 * @author Elena Corpus
 * @version 1.0
 * @since   2020-09-26
 */
public final class Language {
    /**
     * Pascal reserved word token names
     */
    public static final String TOK_RW_AND       = "AND";
    public static final String TOK_RW_ARRAY     = "ARRAY";
    public static final String TOK_RW_BEGIN     = "BEGIN";
    public static final String TOK_RW_DIV       = "DIVIDE";
    public static final String TOK_RW_DO        = "DO";
    public static final String TOK_RW_DOWNTO    = "DOWNTO";
    public static final String TOK_RW_ELSE      = "ELSE";
    public static final String TOK_RW_END       = "END";
    public static final String TOK_RW_FOR       = "FOR";
    public static final String TOK_RW_FUNCTION  = "FUNCTION
";
    public static final String TOK_RW_IF        = "IF";
    public static final String TOK_RW_MOD       = "MOD";
    public static final String TOK_RW_NOT       = "NOT";
    public static final String TOK_RW_OF        = "OF";
    public static final String TOK_RW_OR        = "OR";
    public static final String TOK_RW_PROCEDURE = "
PROCEDURE";
    public static final String TOK_RW_PROGRAM   = "PROGRAM"
;
    public static final String TOK_RW_THEN      = "THEN";
    public static final String TOK_RW_TO        = "TO";
    public static final String TOK_RW_VAR       = "VAR";
    public static final String TOK_RW_WHILE     = "WHILE";

    /**
     * Pascal reserved type specifier names
     */
    public static final String TOK_TS_INT       = "INTTYPE"
;
    public static final String TOK_TS_REAL      = "REALTYPE
";
    public static final String TOK_TS_BOOL      = "BOOLTYPE
";
```

```
45      public static final String TOK_TS_STRING    = "STRTYPE"
    ;
46      public static final String TOK_TS_CHAR      = "CHARTYPE
    ";
47
48      /**
49       * Pascal reserved symbol token names
50       */
51      public static final String TOK_RS_PLUS       = "PLUS";
52      public static final String TOK_RS_MINUS      = "MINUS";
53      public static final String TOK_RS_MULT       = "MULT";
54      public static final String TOK_RS_LT         = "LT";
55      public static final String TOK_RS_LTE        = "LTE";
56      public static final String TOK_RS_GT         = "GT";
57      public static final String TOK_RS_GTE        = "GTE";
58      public static final String TOK_RS_EQU        = "EQU";
59      public static final String TOK_RS_NE         = "NE";
60      public static final String TOK_RS_ASSIGN     = "ASSIGN"
    ;
61      public static final String TOK_RS_COLON      = "COLON";
62      public static final String TOK_RS_SEMICOLON  = "
    SEMICOLON";
63      public static final String TOK_RS_COMMA      = "COMMA";
64      public static final String TOK_RS_LPAREN     = "LAPREN"
    ;
65      public static final String TOK_RS_RPAREN     = "RPAREN"
    ;
66      public static final String TOK_RS_LSQBRACKET = "
    LSQBRACKET";
67      public static final String TOK_RS_RSQBRACKET = "
    RSQBRACKET";
68      public static final String TOK_RS_PERIOD     = "PERIOD"
    ;
69      public static final String TOK_RS_RANGE      = "RANGE";
70
71      /**
72       * Pascal language pattern token names
73       */
74      public static final String TOK_LP_ID      = "ID";
75      public static final String TOK_LP_NUMBER  = "NUMBER";
76      public static final String TOK_LP_STRING  = "STRING";
77      public static final String TOK_LP_CHAR    = "CHAR";
78      public static final String TOK_LP_EOF     = "EOF";
79      public static final String TOK_LP_READ    = "READ";
80      public static final String TOK_LP_READLN  = "READLN";
81      public static final String TOK_LP_WRITE   = "WRITE";
82      public static final String TOK_LP_WRITELN = "WRITELN";
83      public static final String TOK_LP_COMMENT = "COMMENT";
84      public static final String TOK_LP_ERROR   = "ERROR";
85
```

```
 86     /**
 87      * Pascal literal datatype token names
 88      */
 89     public static final String TOK_LIT_INT  = "INTLIT";
 90     public static final String TOK_LIT_REAL = "REALLIT";
 91     public static final String TOK_LIT_BOOL = "BOOLLIT";
 92     public static final String TOK_LIT_CHAR = "CHRLIT";
 93     public static final String TOK_LIT_STR  = "STRLIT";
 94 
 95     /**
 96      * Pascal data type names
 97      */
 98     public static final String TOK_TYPE_INT  = "INTTYPE";
 99     public static final String TOK_TYPE_REAL = "REALTYPE";
100     public static final String TOK_TYPE_BOOL = "BOOLTYPE";
101     public static final String TOK_TYPE_CHAR = "CHRTYPE";
102     public static final String TOK_TYPE_STR  = "STRTYPE";
103 
104     /**
105      * Pascal regex for language pattern constructions
106      */
107     public static final String REGEX_LETTER     = "[a-zA-Z
    ]";
108     public static final String REGEX_DIGIT      = "\\d";
109     public static final String REGEX_TRUE       = "true";
110     public static final String REGEX_FALSE      = "false";
111     public static final String REGEX_NEWLINE    = "\n";
112     public static final String REGEX_SINGLEQT   = "'";
113     public static final String REGEX_ANYTHING   = ".*";
114     public static final String REGEX_WHITESPACE = "\\s";
115 
116     /**
117      * Pascal regex for reserved words
118      */
119     public static final String REGEX_RW_AND       = "and";
120     public static final String REGEX_RW_ARRAY     = "array
    ";
121     public static final String REGEX_RW_BEGIN     = "begin
    ";
122     public static final String REGEX_RW_BOOL      = "
    boolean";
123     public static final String REGEX_RW_CHAR      = "char"
    ;
124     public static final String REGEX_RW_DIV       = "div";
125     public static final String REGEX_RW_DO        = "do";
126     public static final String REGEX_RW_DOWNTO    = "
    downto";
127     public static final String REGEX_RW_ELSE      = "else"
    ;
128     public static final String REGEX_RW_END       = "end";
```

```
129     public static final String REGEX_RW_FOR       = "for";
130     public static final String REGEX_RW_FUNCTION  = "
    function";
131     public static final String REGEX_RW_IF        = "if";
132     public static final String REGEX_RW_INT       = "
    integer";
133     public static final String REGEX_RW_MOD       = "mod";
134     public static final String REGEX_RW_NOT       = "not";
135     public static final String REGEX_RW_OF        = "of";
136     public static final String REGEX_RW_OR        = "or";
137     public static final String REGEX_RW_PROCEDURE = "
    procedure";
138     public static final String REGEX_RW_PROGRAM   = "
    program";
139     public static final String REGEX_RW_REAL      = "real"
    ;
140     public static final String REGEX_RW_STR       = "
    string";
141     public static final String REGEX_RW_THEN      = "then"
    ;
142     public static final String REGEX_RW_TO        = "to";
143     public static final String REGEX_RW_VAR       = "var";
144     public static final String REGEX_RW_WHILE     = "while
    ";
145
146     /**
147      * Pascal regex for reserved symbols
148      */
149     public static final String REGEX_RS_PLUS       = "\\+"
    ;
150     public static final String REGEX_RS_MINUS      = "-";
151     public static final String REGEX_RS_MULT       = "\\*"
    ;
152     public static final String REGEX_RS_DIVIDE     = "/";
153     public static final String REGEX_RS_LT         = "<";
154     public static final String REGEX_RS_LTE        = "^<=$
    ";
155     public static final String REGEX_RS_GT         = ">";
156     public static final String REGEX_RS_GTE        = "^>=$
    ";
157     public static final String REGEX_RS_EQU        = "=";
158     public static final String REGEX_RS_NE         = "^<>$
    ";
159     public static final String REGEX_RS_ASSIGN     = "^:=$
    ";
160     public static final String REGEX_RS_COLON      = ":";
161     public static final String REGEX_RS_SEMICOLON  = ";";
162     public static final String REGEX_RS_COMMA      = ",";
163     public static final String REGEX_RS_LPAREN     = "\\("
    ;
```

```
164     public static final String REGEX_RS_RPAREN     = "\\)"
    ;
165     public static final String REGEX_RS_LSQBRACKET = "\\["
    ;
166     public static final String REGEX_RS_RSQBRACKET = "\\]"
    ;
167     public static final String REGEX_RS_LCRLYBRACK = "\\{"
    ;
168     public static final String REGEX_RS_RCRLYBRACK = "}";
169     public static final String REGEX_RS_LBIGRAM    = "\\(
    \\*";
170     public static final String REGEX_RS_RBIGRAM    = "\\*
    \\)";
171     public static final String REGEX_RS_PERIOD     = "\\."
    ;
172     public static final String REGEX_RS_DECIMAL    = ".";
173     public static final String REGEX_RS_RANGE      = "..";
174
175     /**
176      * Pascal regex for literal datatypes
177      */
178     public static final String REGEX_LIT_INT    = "(\\
    +|-)?"+REGEX_DIGIT + "+";
179     public static final String REGEX_LIT_REAL   = "(\\
    +|-)?"+REGEX_DIGIT + "+\\." +REGEX_DIGIT+"+";
180     public static final String REGEX_LIT_BOOL   =
    REGEX_TRUE + "|" + REGEX_FALSE;
181     public static final String REGEX_LIT_CHAR   = "\'.\'";
182     public static final String REGEX_LIT_STRING =
    REGEX_SINGLEQT + REGEX_ANYTHING + REGEX_SINGLEQT;
183
184     /**
185      * Pascal regex for complex language patterns
186      */
187     public static final String REGEX_PT_ID          =
    REGEX_LETTER+"("+REGEX_LETTER+"|"+REGEX_DIGIT+")*";
188     public static final String REGEX_PT_CRLYCOMMENT =
    REGEX_RS_LCRLYBRACK+REGEX_ANYTHING+REGEX_RS_RCRLYBRACK;
189     public static final String REGEX_PT_BGRMCOMMENT =
    REGEX_RS_LBIGRAM+REGEX_ANYTHING+REGEX_RS_RBIGRAM;
190     public static final String REGEX_PT_ADDOP       = "["+
    REGEX_RS_PLUS+REGEX_RS_MINUS+"]";
191     public static final String REGEX_PT_RELOP       =
    "^(^("+REGEX_RS_NE+")?|^("+REGEX_RS_LTE+")?|^("+
    REGEX_RS_GTE+")?|^("+REGEX_RS_LT+")?|^("+REGEX_RS_GT+
    ")?|^("+REGEX_RS_EQU+")?)?";
192
193     /**
194      * State values for lexical analysis
195      */
```

```
196     public static final int ST_START
                              = 0;
197     public static final int ST_COLON
                             = 1;
198     public static final int ST_COLON_EQUALS
                       = 2;
199     public static final int ST_LCRLYBRK
                         = 3;
200     public static final int ST_LCRLYBRK_IGNOREALL
              = 4;
201     public static final int ST_LCRLYBRK_IGNOREALL_RCRLYBRK
    = 5;
202     public static final int ST_LPAREN
                            = 6;
203     public static final int ST_LBIGRAM
                           = 7;
204     public static final int ST_LBIGRAM_IGNOREALL
                  = 8;
205     public static final int ST_LBIGRAM_IGNOREALL_RBIGRAM
      = 9;
206     public static final int ST_RPAREN
                            = 10;
207     public static final int ST_LETTER
                            = 11;
208     public static final int ST_ID
                              = 12;
209     public static final int ST_COMMA
                             = 13;
210     public static final int ST_SEMICOLON
                         = 14;
211     public static final int ST_EQU
                               = 15;
212     public static final int ST_LT
                              = 16;
213     public static final int ST_LT_EQU
                            = 17;
214     public static final int ST_NE
                              = 18;
215     public static final int ST_GT
                              = 19;
216     public static final int ST_GT_EQU
                            = 20;
217     public static final int ST_DIGIT
                             = 21;
218     public static final int ST_INTEGER
                           = 22;
219     public static final int ST_REAL
                             = 23;
220     public static final int ST_PERIOD
                            = 24;
```

```
221     public static final int ST_RANGE
                             = 25;
222     public static final int ST_PLUS
                              = 26;
223     public static final int ST_MINUS
                             = 27;
224     public static final int ST_MULT
                              = 28;
225     public static final int ST_DIVIDE
                            = 29;
226     public static final int ST_LSQBRACKET
                          = 30;
227     public static final int ST_RSQBRACKET
                          = 31;
228     public static final int ST_SINGLEQT
                            = 32;
229     public static final int ST_SINGLEQT_ACCEPTALL
              = 33;
230     public static final int ST_SINGLEQT_ACCEPTALL_SINGLEQT
    = 34;
231     public static final int ST_ERROR = 35;
232
233     /**
234      * empty constructor for reading
235      */
236     public Language() {}
237 }
238
```