Computer Science Department - San Francisco State University

CSC 413 – Fall 2017

**Gabriel Equitz – id: 915254839**

Assignment 2 – Lexer

Repository at: <https://github.com/CSC-413-SFSU-02/csc413-p2-gequitz.git>

**Introduction**: The project consisted of writing a computer code that works as a lexer, that is, it a computer program that performs lexical analysis. A lexer breaks down an expression of a computer program in components, known as tokens, which are recognized by the compiler. Example of tokens are “{“, “+”, “>”, “write”, etc. The program was written in Java in the NetBeans environment.

The tokens accepted by the compiler are hardcoded in a file called “tokens”. That file contains two columns: the first column is the name of the token and the second is the token associated with it. Examples: Grater >, Equal =, etc.

The packages and their respective classes of the project are the following:

**Package name**: lexer.setup

Classes:

***TokenSetup.java***: Reads the tokens from the file “tokens” and automatically creates the Tokens.java and TokenTypes.java classes. If any change is made in the file “tokens”, this file has to be run again to update those two classes.

**Package name**: lexer

Classes:

***Lexer.java***: This class is responsible for scanning the source file and displaying the each line, the tokens and their beginning and ending position, the line number and in the last step, .java displays the entire source file. If a token is not listed in the file “tokens” an error message is displayed.

***SourceReader.java***: This class gets from class Lexer.java the source file and reads each character of the source file and also separates that file in the individual lines, storing information on the line number, the position of the characters and a list of all the lines of the file. This class is called by Lexer.java and sends back the information to it.

***Token.java***: This class records all the information of a token: its symbol, the starting column in the source file and, ending column in the source file and the line number. This class is called by Lexer .java and sends back information to it.

***Symbol.java***: This class is used to store the token names and their respective kinds. For example, the name “Greater” will have associated with it the kind “>”.This class is called by Lexer.java and sends back information to it.

***TokenType.java***: This class is automatically generated by the class TokenSetup.java. It contains a HashMap table of the mappings from token names to their respective symbols. This class is called by Lexer.java and sends back information to it.

***Tokens.java***: This class is automatically generated by the class TokenSetup.java. It contains the enumerations of all the tokens. This class is called by Lexer.java and sends back information to it.

To compile the project go to the directory where the files are and type the following:

(NOTE: TO COMPILE USING THE COMMAND LINE, IT IS ADVISED TO COMMENT OUT IN THE CLASSES THE LINE THAT SAYS: “package lexer” and package “lexer.setup” OTHERWISE COMPILATIONS USING THE COMMAND LINE MIGHT GIVE AN ERROR. HOWEVER IF THE COMPILATION IS DONE IN NETBEANS, THOSE LINES ARE NECESSARY.)

javac \*.java setup\\*.java

to run:

java Lexer SourceReader Symbol Token Tokens TokenType

to create a jar file:

jar –cvfe Lexer.jar Lexer \*.class

To run the jar file:

java –jar Lexer.jar

The diagram below shows how the classes interact with each other:

SourceReader

Token

Symbol

Lexer

Tokens

TokenType

Setup

List of Tokens accepted by the program (name and symbol or word):

Program program

Int int

BOOLean boolean

If if

Then then

Else else

While while

Function function

Return return

Identifier <id>

INTeger <int>

LeftBrace {

RightBrace }

LeftParen (

RightParen )

Comma ,

Assign =

Equal ==

NotEqual !=

Less <

LessEqual <=

Plus +

Minus -

Or |

And &

Multiply \*

Divide /

Comment //

Greater >

GreaterEqual >=

Void void

Float float

Double double

BooleanAnd &&

BooleanOr ||

**Example of input and output**:

**Input**:

program { int i int j

i = i + j + 7

j = write(i)

}

**Output (column: symbol,starting position, ending position, line number)**:

Source file: simple.x

user.dir: C:\Users\Gabriel\Documents\NetBeansProjects\Lexer

READLINE: program { int i int j

program left: 0 right: 6 line: 1

{ left: 8 right: 8 line: 1

int left: 10 right: 12 line: 1

i left: 14 right: 14 line: 1

int left: 16 right: 18 line: 1

j left: 20 right: 20 line: 1

READLINE: i = i + j + 7

i left: 3 right: 3 line: 2

= left: 5 right: 5 line: 2

i left: 7 right: 7 line: 2

+ left: 9 right: 9 line: 2

j left: 11 right: 11 line: 2

+ left: 13 right: 13 line: 2

7 left: 15 right: 15 line: 2

READLINE: j = write(i)

j left: 3 right: 3 line: 3

= left: 5 right: 5 line: 3

write left: 7 right: 11 line: 3

( left: 12 right: 12 line: 3

i left: 13 right: 13 line: 3

) left: 14 right: 14 line: 3

READLINE: }

} left: 0 right: 0 line: 4

1. program { int i int j

2. i = i + j + 7

3. j = write(i)

4. }

The requirements for the project were:

1. To modify the program to allow input via a filename provided as the command line.
2. To update the compiler to accept seven additional tokens: <, >=, ||, &&, void, float, double..
3. To update the Token.java class to include the line number that the token was found (for subsequent error reporting, etc.)
4. To update the Lexer.java class output for readability, and to include the line number from the Token.
5. To update the Lexer.java class to include a printout, with line number, of each line read in the source file.

Challenges: The most difficult part of the assignment was to set up display of the entire source file the end of the program run which was achieved using a java arrayList. All the requirements were done.

This project gave me the background on how compilers work.

The methods of each class are given below:

Classes:

***TokenSetup.java***:

Methods:

public static void main(String args[])

TokenSetup()

public void getNextToken() throws IOException

public void initTokenClasses()

***Lexer.java***:

Methods:

public static void main(String args[])

***SourceReader.java***:

Methods:

public SourceReader(String sourceFile) throws IOException

void close()

public char read() throws IOException

public int getPosition()

public int getLineno()

public String toString()

public ArrayList getArrayList()

***Symbol.java***:

Methods:

private Symbol(String n, Tokens kind)

public String toString()

public Tokens getKind()

***Token.java:***

Methods:

public Token(int leftPosition, int rightPosition, Symbol sym, int line\_numb)

public Symbol getSymbol()

public String toString()

public int getLeftPosition()

public int getRightPosition()

public int getLineNumber()

public Tokens getKind()

***Tokens.java***:

Methods:

Constructor

***TokenType.java***:

Methods:

public TokenType()

public Token newIdToken(String id, int startPosition, int endPosition, int lineNumber)

public Token newNumberToken(String number, int startPosition, int endPosition, int lineNumber)

public Token makeToken(String s, int startPosition, int endPosition)

public Token nextToken()

public String toString() {