Assignment 1: Java Lexer

Ayush Kumar (170195)

26 January 2020

I pledge on my honor that I have not given or received any unauthorized assistance.

This is the solution for the 1^{st} assignment of this course. We have built a Lexical Analyser for Java [2]. It parses and tokenizes a given .java file and classifies each token as one of **Identifier**, **Keyword**, **Literal**, **Separator** or **Operator** and records its count. This metadata is stored in the form of a HashTable. After the parsing is done, the contents of the HashTable are written in CSV format on the Standard Output (**stdout**) if no output file is specified and into the output file otherwise. The CSV file contains 3 columns one each for 'Lexeme', 'Token' and 'Count' respectively. The Lexical Analyzer also reports simple lexical errors on the Standard Error (**stderr**).

We have used the Flex tool [1] for tokenizing the input file using Regular Expressions. The Flex tool internally creates a DFA for each regex, combines all the DFAs and minimizes it to create the tokenizer. For this, we wrote a 'javalexer.l' file which is transpiled by **lex** into 'lex.yy.c' file. This is then compiled with the **g++** toolchain to obtain the final binary. The following solution files are present along with this PDF.

1. javalexer.l
2. lex.yy.c
3. run.sh
4. test_1.java
5. test_2.java
6. test_3.java
7. test_4.java

Even though the 'lex.yy.c' file can be obtained from 'javalexer.l', we have still provided a copy of it. There are also four test .java files present which are a few of the many testcases this lexer has been tested upon.

To compile and run the lexer, use the commands given below. The output in written in the file 'test_1.out'.

```
$ sh run.sh test_1.java
```

As an alternative, the lexer can also be compiled and executed simply by using the **flex** and **g++** toolchains,

```
$ lex --yylineno javalexer.l
$ g++ -std=c++11 -o lexer lex.yy.c -ll
$ ./lexer <input_file> <output_file>
```

If the <output_file> is omitted, the output will be written on the Standard Output (**stdout**). If the <input_file> is omitted, the input will be read from the Standard Input (**stdin**).

References

[1] Flex 2.5.35 http://dinosaur.compilertools.net/

[2] Java Language Specification: Lexical Structure https://docs.oracle.com/javase/specs/jls/se8/html/jls-3.html