Project 2: Lexer

CSC 413 Spring 2016 Professor Yoon

Developer's Guide

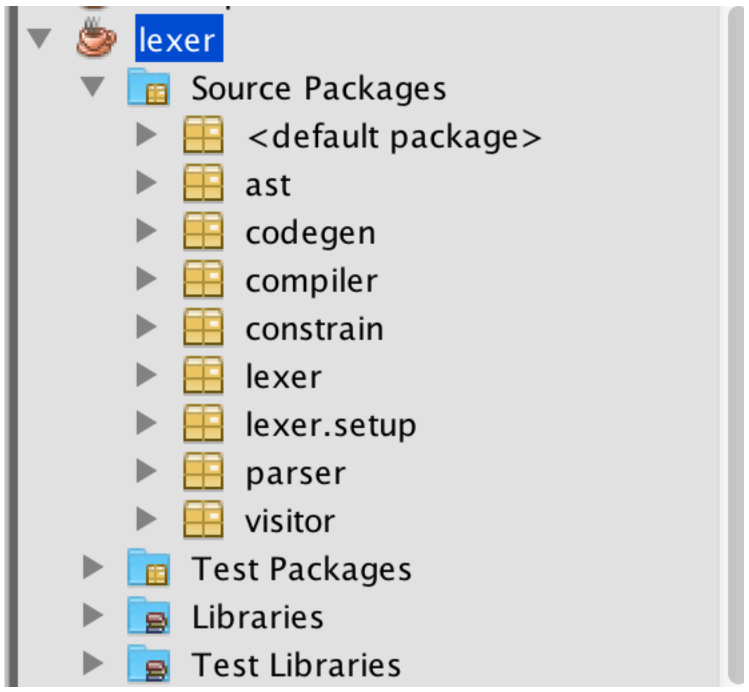
February 27, 2016

Qihong Kuang

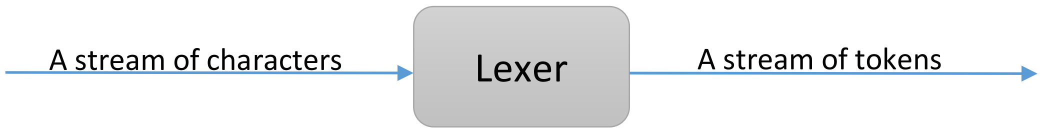
Table of Contents

1. Introduction

This document is a guide to explain how developers handle and design the lexer. This java project contains several packages. In project 2, we only need to revise the lexer package and the setup package.



The graph below is to show how Lexer work:



Lexer will first read the content of a file character by character from the command line. After running lexer project, Lexer will print out the results to indicate the types of token which Lexer have read from file. In order to trace the errors and clarify the tokens, Lexer needs to print out the line number, left position and right position of the token.

In Project 2, users needs to add more tokens into the token file, such >, >=, float, scientific notation, char and String. After adding these token into the token file, users must run tokenSetup.java so that the new tokens are added to the hash table. Users can check the updates in TokenType.java and Tokens.java

Lexer.java and SourceReader.java are the two main files that the developer needs to revise to recognize char<char>, String<string>, float<float> and scientificN<scientificN>.

2. Compile information.

Developer already added the extra tokens into the token file and run TokenSetup.java. There is no need the users compile the TokenSetup.java again.

Below is the compile information:

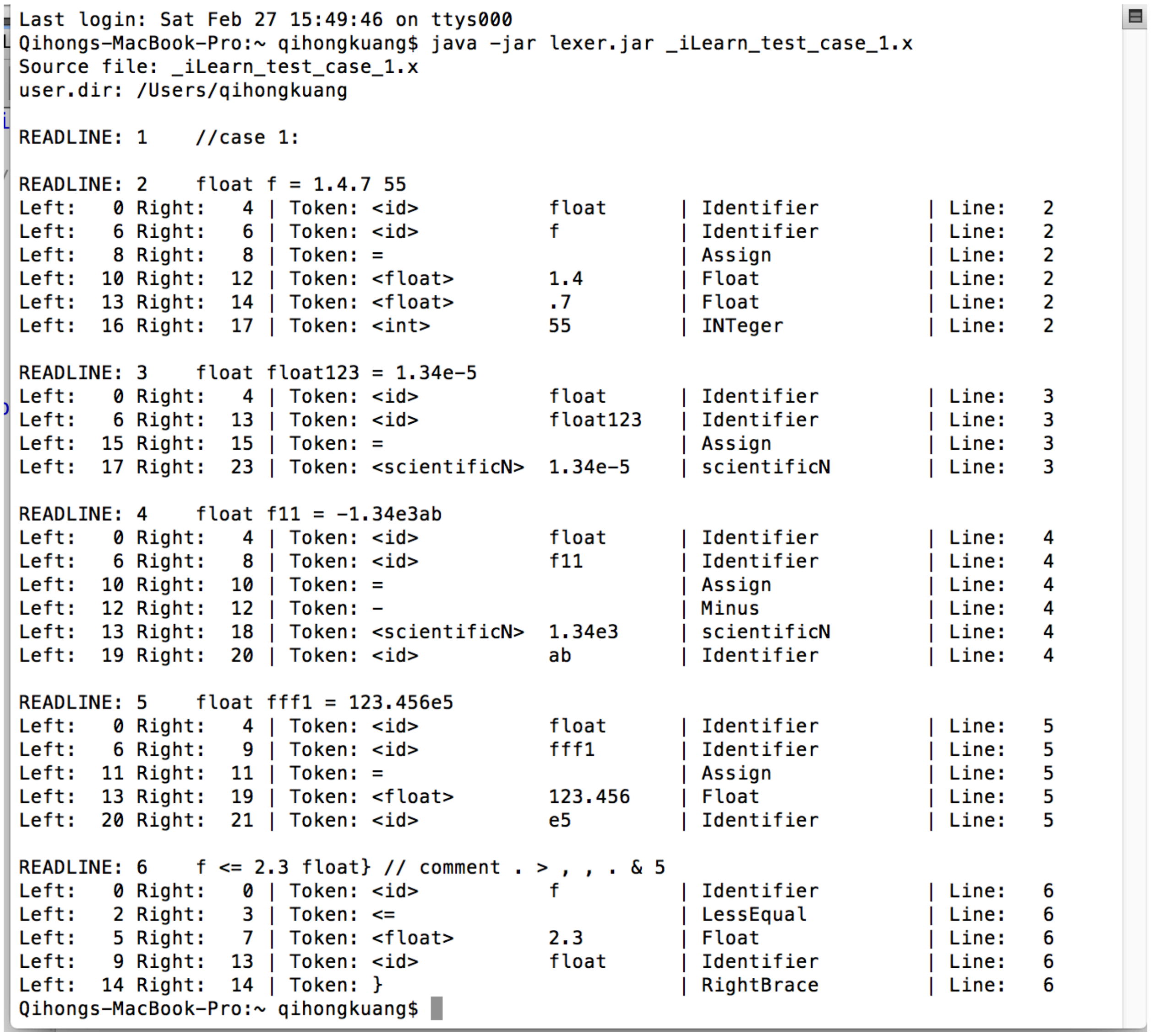
Use cd command navigate to the directory of Lexer source codes. Be sure all the test files and java files are in the same directory. Then, type:

java –jar lexer.java (follow by one argument which is the test file)

2.1. Test Result

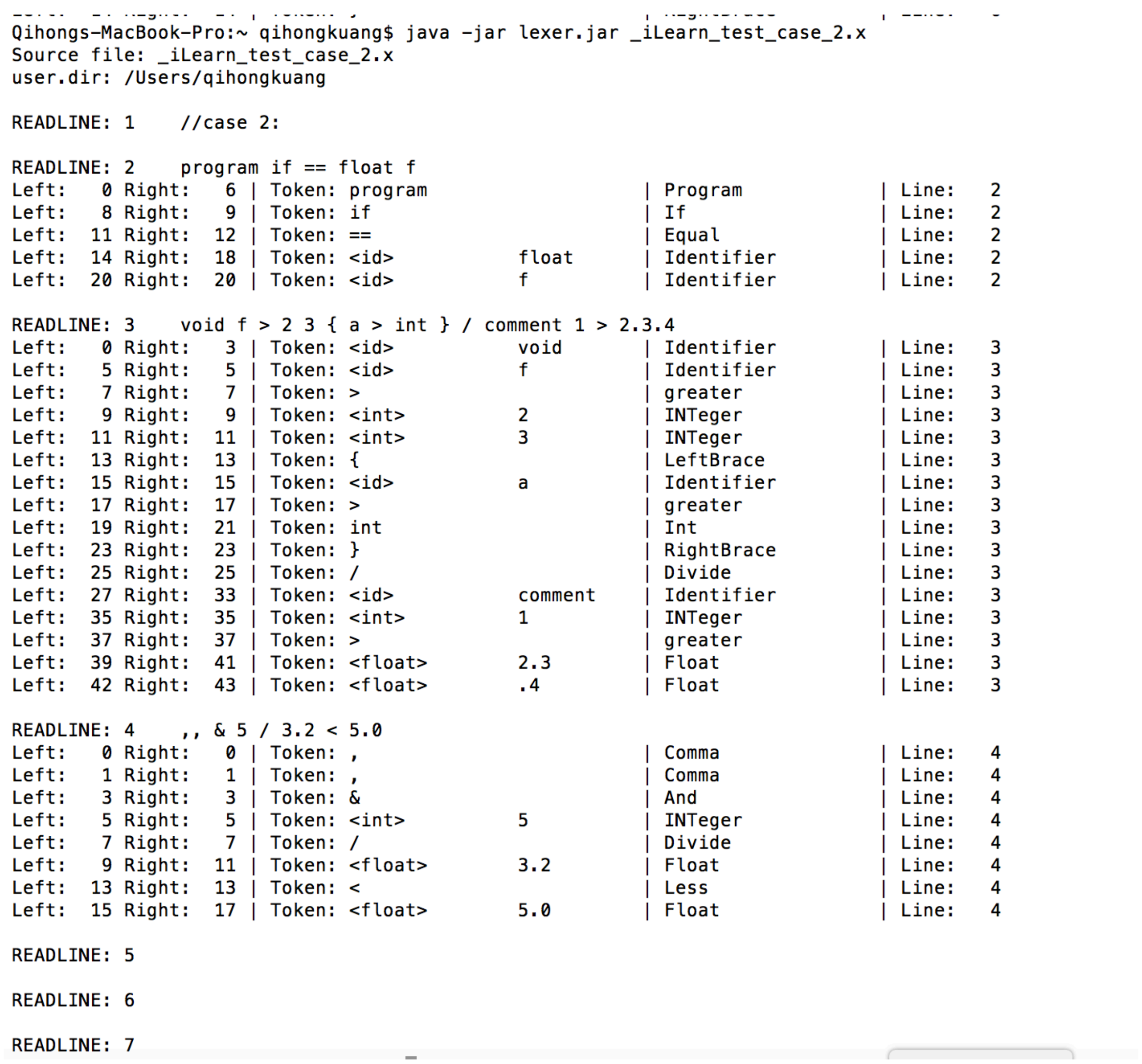
Below are a series of test cases print out result:

a. _iLearn_test_case_1.x:



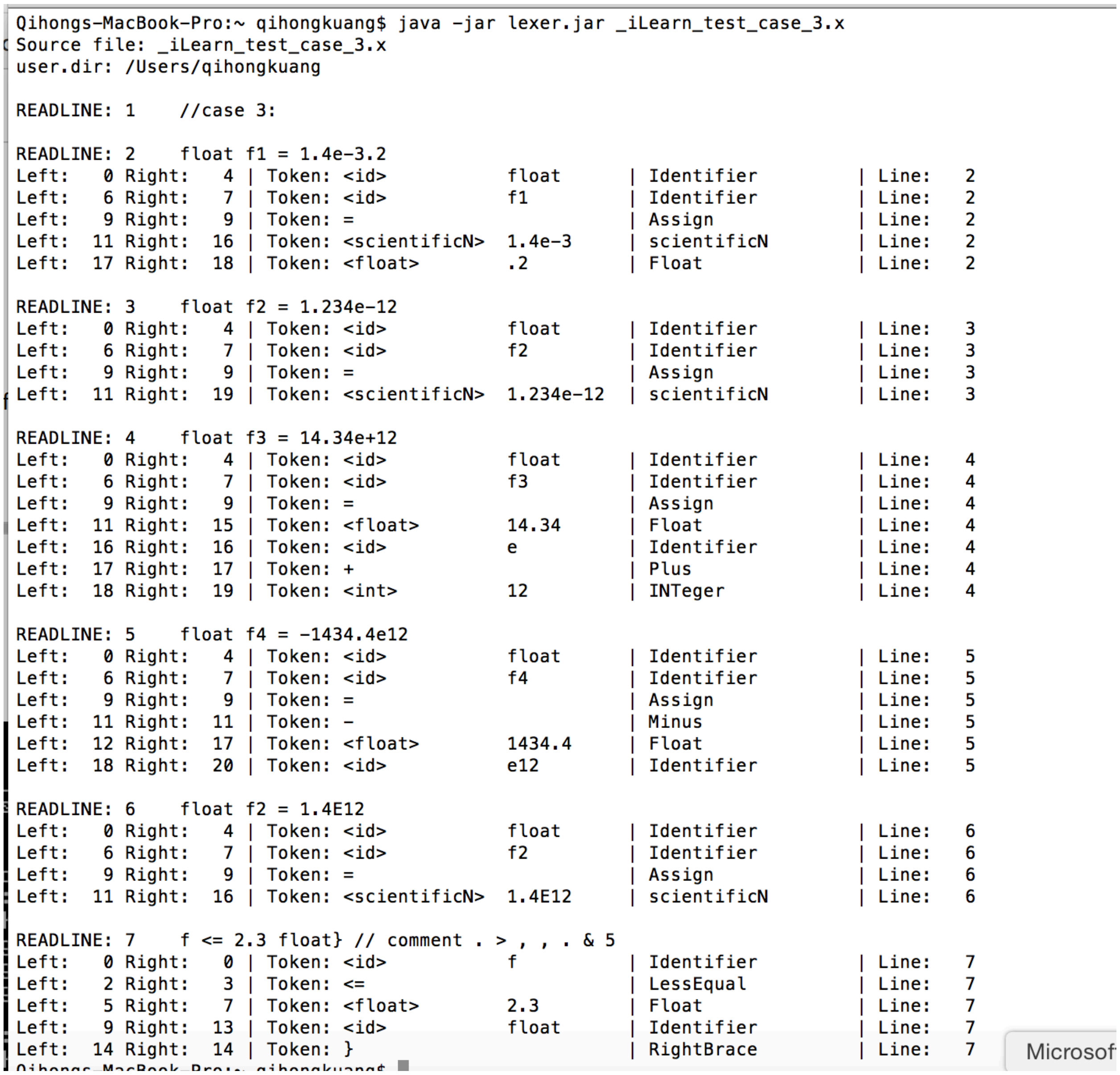
```
Last login: Sat Feb 27 15:49:46 on ttys000
Qihongs-MacBook-Pro:~ qihongkuang$ java -jar lexer.jar _iLearn_test_case_1.x
Source file: _iLearn_test_case_1.x
user.dir: /Users/qihongkuang

READLINE: 1     //case 1:

READLINE: 2     float f = 1.4.7 55
Left:   0 Right:   4 | Token: <id>          float      | Identifier          | Line:   2
Left:   6 Right:   6 | Token: <id>          f          | Identifier          | Line:   2
Left:   8 Right:   8 | Token: =                        | Assign              | Line:   2
Left:  10 Right:  12 | Token: <float>       1.4        | Float               | Line:   2
Left:  13 Right:  14 | Token: <float>       .7         | Float               | Line:   2
Left:  16 Right:  17 | Token: <int>         55         | INTeger             | Line:   2

READLINE: 3     float float123 = 1.34e-5
Left:   0 Right:   4 | Token: <id>          float      | Identifier          | Line:   3
Left:   6 Right:  13 | Token: <id>          float123   | Identifier          | Line:   3
Left:  15 Right:  15 | Token: =                        | Assign              | Line:   3
Left:  17 Right:  23 | Token: <scientificN> 1.34e-5    | scientificN         | Line:   3

READLINE: 4     float f11 = -1.34e3ab
Left:   0 Right:   4 | Token: <id>          float      | Identifier          | Line:   4
Left:   6 Right:   8 | Token: <id>          f11        | Identifier          | Line:   4
Left:  10 Right:  10 | Token: =                        | Assign              | Line:   4
Left:  12 Right:  12 | Token: -                        | Minus               | Line:   4
Left:  13 Right:  18 | Token: <scientificN> 1.34e3     | scientificN         | Line:   4
Left:  19 Right:  20 | Token: <id>          ab         | Identifier          | Line:   4

READLINE: 5     float fff1 = 123.456e5
Left:   0 Right:   4 | Token: <id>          float      | Identifier          | Line:   5
Left:   6 Right:   9 | Token: <id>          fff1       | Identifier          | Line:   5
Left:  11 Right:  11 | Token: =                        | Assign              | Line:   5
Left:  13 Right:  19 | Token: <float>       123.456    | Float               | Line:   5
Left:  20 Right:  21 | Token: <id>          e5         | Identifier          | Line:   5

READLINE: 6     f <= 2.3 float} // comment . > , , . & 5
Left:   0 Right:   0 | Token: <id>          f          | Identifier          | Line:   6
Left:   2 Right:   3 | Token: <=                       | LessEqual           | Line:   6
Left:   5 Right:   7 | Token: <float>       2.3        | Float               | Line:   6
Left:   9 Right:  13 | Token: <id>          float      | Identifier          | Line:   6
Left:  14 Right:  14 | Token: }                        | RightBrace          | Line:   6
Qihongs-MacBook-Pro:~ qihongkuang$
```

b. _iLearn_test_case_2.x



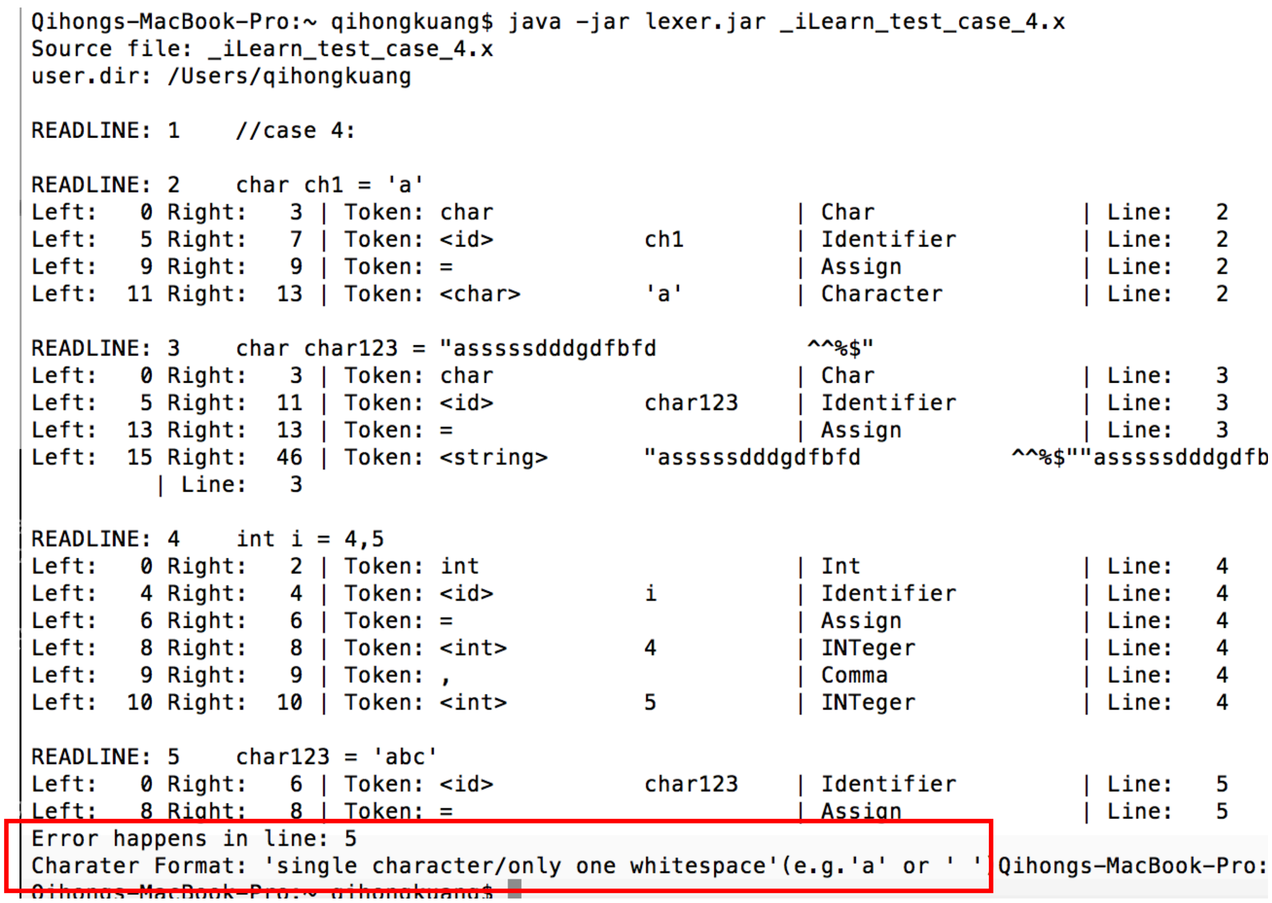
```
Qihongs-MacBook-Pro:~ qihongkuang$ java -jar lexer.jar _iLearn_test_case_2.x
Source file: _iLearn_test_case_2.x
user.dir: /Users/qihongkuang

READLINE: 1    //case 2:

READLINE: 2    program if == float f
Left:   0 Right:   6 | Token: program               | Program            | Line:   2
Left:   8 Right:   9 | Token: if                    | If                 | Line:   2
Left:  11 Right:  12 | Token: ==                    | Equal              | Line:   2
Left:  14 Right:  18 | Token: <id>          float   | Identifier         | Line:   2
Left:  20 Right:  20 | Token: <id>          f       | Identifier         | Line:   2

READLINE: 3    void f > 2 3 { a > int } / comment 1 > 2.3.4
Left:   0 Right:   3 | Token: <id>          void    | Identifier         | Line:   3
Left:   5 Right:   5 | Token: <id>          f       | Identifier         | Line:   3
Left:   7 Right:   7 | Token: >                     | greater            | Line:   3
Left:   9 Right:   9 | Token: <int>         2       | INTeger            | Line:   3
Left:  11 Right:  11 | Token: <int>         3       | INTeger            | Line:   3
Left:  13 Right:  13 | Token: {                     | LeftBrace          | Line:   3
Left:  15 Right:  15 | Token: <id>          a       | Identifier         | Line:   3
Left:  17 Right:  17 | Token: >                     | greater            | Line:   3
Left:  19 Right:  21 | Token: int                   | Int                | Line:   3
Left:  23 Right:  23 | Token: }                     | RightBrace         | Line:   3
Left:  25 Right:  25 | Token: /                     | Divide             | Line:   3
Left:  27 Right:  33 | Token: <id>          comment | Identifier         | Line:   3
Left:  35 Right:  35 | Token: <int>         1       | INTeger            | Line:   3
Left:  37 Right:  37 | Token: >                     | greater            | Line:   3
Left:  39 Right:  41 | Token: <float>       2.3     | Float              | Line:   3
Left:  42 Right:  43 | Token: <float>       .4      | Float              | Line:   3

READLINE: 4    ,, & 5 / 3.2 < 5.0
Left:   0 Right:   0 | Token: ,                     | Comma              | Line:   4
Left:   1 Right:   1 | Token: ,                     | Comma              | Line:   4
Left:   3 Right:   3 | Token: &                     | And                | Line:   4
Left:   5 Right:   5 | Token: <int>         5       | INTeger            | Line:   4
Left:   7 Right:   7 | Token: /                     | Divide             | Line:   4
Left:   9 Right:  11 | Token: <float>       3.2     | Float              | Line:   4
Left:  13 Right:  13 | Token: <                     | Less               | Line:   4
Left:  15 Right:  17 | Token: <float>       5.0     | Float              | Line:   4

READLINE: 5

READLINE: 6

READLINE: 7
```

c. _iLearn_test_case_3.x



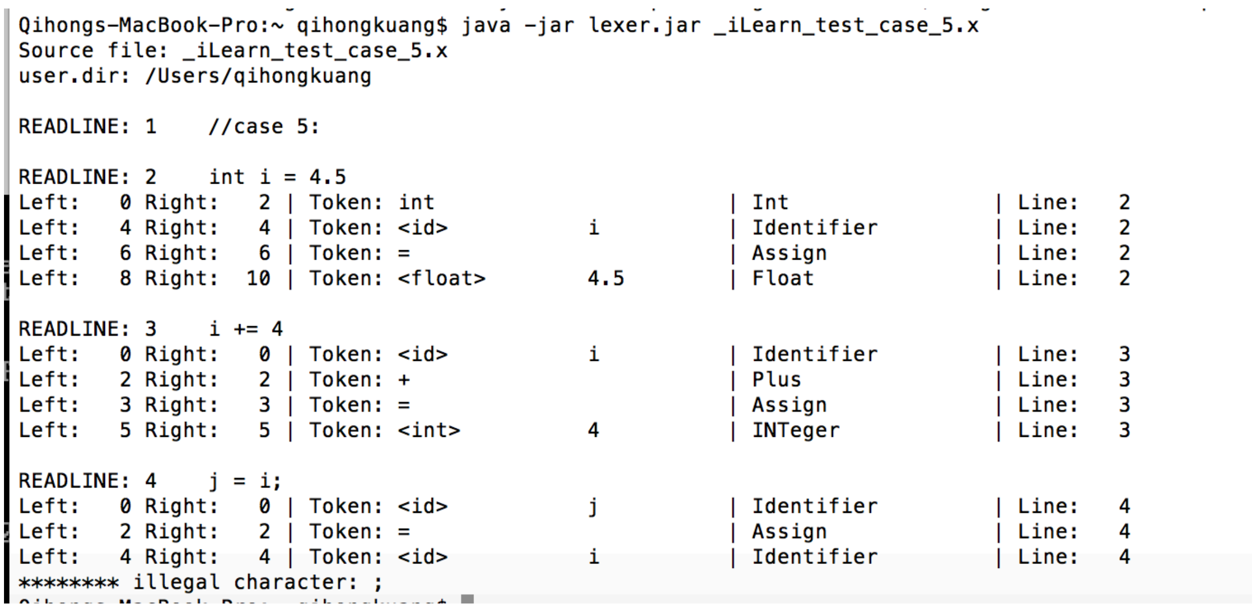
```
Qihongs-MacBook-Pro:~ qihongkuang$ java -jar lexer.jar _iLearn_test_case_3.x
Source file: _iLearn_test_case_3.x
user.dir: /Users/qihongkuang

READLINE: 1    //case 3:

READLINE: 2    float f1 = 1.4e-3.2
Left:   0 Right:   4 | Token: <id>           float      | Identifier         | Line:   2
Left:   6 Right:   7 | Token: <id>           f1         | Identifier         | Line:   2
Left:   9 Right:   9 | Token: =                         | Assign             | Line:   2
Left:  11 Right:  16 | Token: <scientificN>  1.4e-3     | scientificN        | Line:   2
Left:  17 Right:  18 | Token: <float>        .2         | Float              | Line:   2

READLINE: 3    float f2 = 1.234e-12
Left:   0 Right:   4 | Token: <id>           float      | Identifier         | Line:   3
Left:   6 Right:   7 | Token: <id>           f2         | Identifier         | Line:   3
Left:   9 Right:   9 | Token: =                         | Assign             | Line:   3
Left:  11 Right:  19 | Token: <scientificN>  1.234e-12  | scientificN        | Line:   3

READLINE: 4    float f3 = 14.34e+12
Left:   0 Right:   4 | Token: <id>           float      | Identifier         | Line:   4
Left:   6 Right:   7 | Token: <id>           f3         | Identifier         | Line:   4
Left:   9 Right:   9 | Token: =                         | Assign             | Line:   4
Left:  11 Right:  15 | Token: <float>        14.34      | Float              | Line:   4
Left:  16 Right:  16 | Token: <id>           e          | Identifier         | Line:   4
Left:  17 Right:  17 | Token: +                         | Plus               | Line:   4
Left:  18 Right:  19 | Token: <int>          12         | INTeger            | Line:   4

READLINE: 5    float f4 = -1434.4e12
Left:   0 Right:   4 | Token: <id>           float      | Identifier         | Line:   5
Left:   6 Right:   7 | Token: <id>           f4         | Identifier         | Line:   5
Left:   9 Right:   9 | Token: =                         | Assign             | Line:   5
Left:  11 Right:  11 | Token: -                         | Minus              | Line:   5
Left:  12 Right:  17 | Token: <float>        1434.4     | Float              | Line:   5
Left:  18 Right:  20 | Token: <id>           e12        | Identifier         | Line:   5

READLINE: 6    float f2 = 1.4E12
Left:   0 Right:   4 | Token: <id>           float      | Identifier         | Line:   6
Left:   6 Right:   7 | Token: <id>           f2         | Identifier         | Line:   6
Left:   9 Right:   9 | Token: =                         | Assign             | Line:   6
Left:  11 Right:  16 | Token: <scientificN>  1.4E12     | scientificN        | Line:   6

READLINE: 7    f <= 2.3 float} // comment . > , , . & 5
Left:   0 Right:   0 | Token: <id>           f          | Identifier         | Line:   7
Left:   2 Right:   3 | Token: <=                        | LessEqual          | Line:   7
Left:   5 Right:   7 | Token: <float>        2.3        | Float              | Line:   7
Left:   9 Right:  13 | Token: <id>           float      | Identifier         | Line:   7
Left:  14 Right:  14 | Token: }                         | RightBrace         | Line:   7
```

d. _iLearn_test_case_4.x

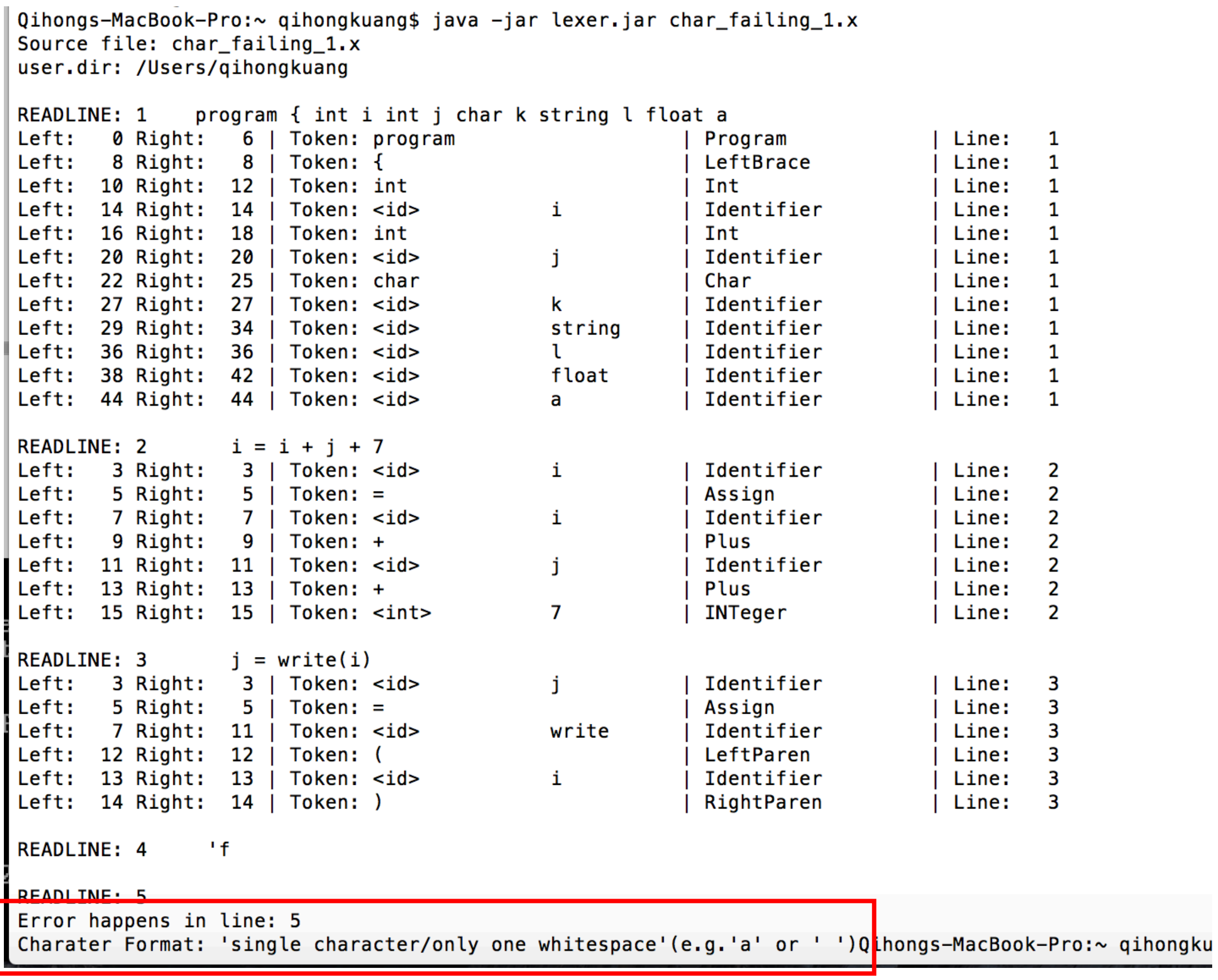


```
Qihongs-MacBook-Pro:~ qihongkuang$ java -jar lexer.jar _iLearn_test_case_4.x
Source file: _iLearn_test_case_4.x
user.dir: /Users/qihongkuang

READLINE: 1    //case 4:

READLINE: 2    char ch1 = 'a'
Left:   0 Right:   3 | Token: char                       | Char                 | Line:   2
Left:   5 Right:   7 | Token: <id>           ch1         | Identifier           | Line:   2
Left:   9 Right:   9 | Token: =                          | Assign               | Line:   2
Left:  11 Right:  13 | Token: <char>         'a'         | Character            | Line:   2

READLINE: 3    char char123 = "asssssdddgdfbfd           ^^%$"
Left:   0 Right:   3 | Token: char                       | Char                 | Line:   3
Left:   5 Right:  11 | Token: <id>           char123     | Identifier           | Line:   3
Left:  13 Right:  13 | Token: =                          | Assign               | Line:   3
Left:  15 Right:  46 | Token: <string>       "asssssdddgdfbfd           ^^%$""asssssdddgdfb
         | Line:   3

READLINE: 4    int i = 4,5
Left:   0 Right:   2 | Token: int                        | Int                  | Line:   4
Left:   4 Right:   4 | Token: <id>           i           | Identifier           | Line:   4
Left:   6 Right:   6 | Token: =                          | Assign               | Line:   4
Left:   8 Right:   8 | Token: <int>          4           | INTeger              | Line:   4
Left:   9 Right:   9 | Token: ,                          | Comma                | Line:   4
Left:  10 Right:  10 | Token: <int>          5           | INTeger              | Line:   4

READLINE: 5    char123 = 'abc'
Left:   0 Right:   6 | Token: <id>           char123     | Identifier           | Line:   5
Left:   8 Right:   8 | Token: =                          | Assign               | Line:   5
Error happens in line: 5
Charater Format: 'single character/only one whitespace'(e.g.'a' or ' ')Qihongs-MacBook-Pro:
Qihongs-MacBook-Pro:~ qihongkuang$
```

e. _iLearn_test_case_5.x

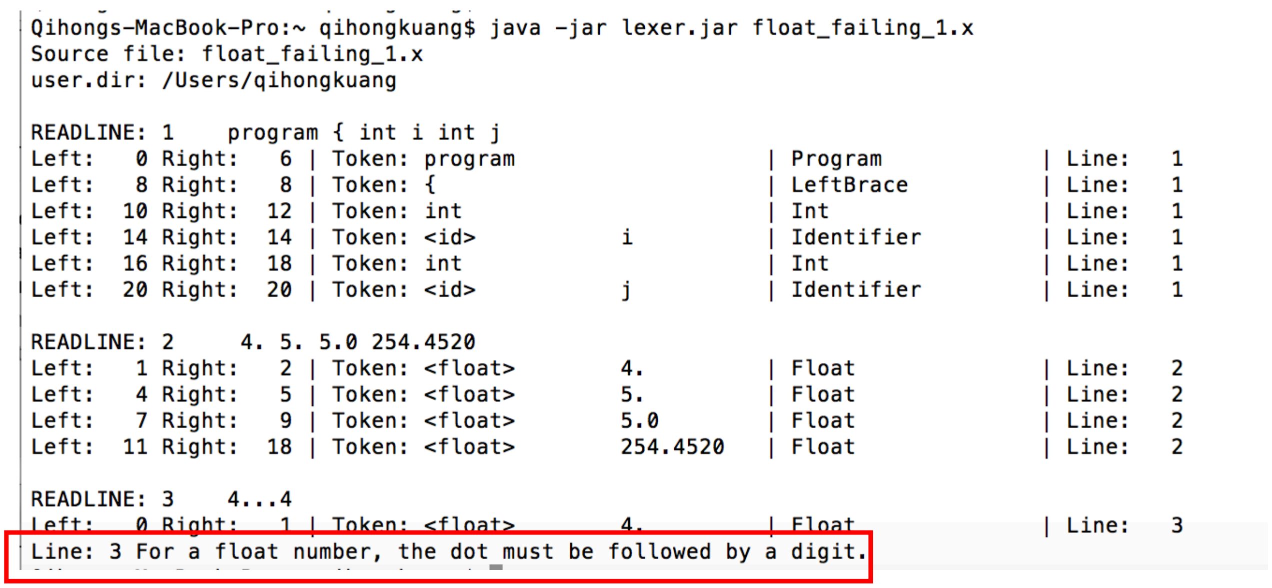

```
Qihongs-MacBook-Pro:~ qihongkuang$ java -jar lexer.jar _iLearn_test_case_5.x
Source file: _iLearn_test_case_5.x
user.dir: /Users/qihongkuang

READLINE: 1    //case 5:

READLINE: 2    int i = 4.5
Left:   0 Right:   2 | Token: int                     | Int                 | Line:   2
Left:   4 Right:   4 | Token: <id>         i          | Identifier          | Line:   2
Left:   6 Right:   6 | Token: =                       | Assign              | Line:   2
Left:   8 Right:  10 | Token: <float>      4.5        | Float               | Line:   2

READLINE: 3    i += 4
Left:   0 Right:   0 | Token: <id>         i          | Identifier          | Line:   3
Left:   2 Right:   2 | Token: +                       | Plus                | Line:   3
Left:   3 Right:   3 | Token: =                       | Assign              | Line:   3
Left:   5 Right:   5 | Token: <int>        4          | INTeger             | Line:   3

READLINE: 4    j = i;
Left:   0 Right:   0 | Token: <id>         j          | Identifier          | Line:   4
Left:   2 Right:   2 | Token: =                       | Assign              | Line:   4
Left:   4 Right:   4 | Token: <id>         i          | Identifier          | Line:   4
******** illegal character: ;
```

f. char_failing_1.x



```
Qihongs-MacBook-Pro:~ qihongkuang$ java -jar lexer.jar char_failing_1.x
Source file: char_failing_1.x
user.dir: /Users/qihongkuang

READLINE: 1    program { int i int j char k string l float a
Left:    0 Right:   6 | Token: program              | Program           | Line:   1
Left:    8 Right:   8 | Token: {                    | LeftBrace         | Line:   1
Left:   10 Right:  12 | Token: int                  | Int               | Line:   1
Left:   14 Right:  14 | Token: <id>         i       | Identifier        | Line:   1
Left:   16 Right:  18 | Token: int                  | Int               | Line:   1
Left:   20 Right:  20 | Token: <id>         j       | Identifier        | Line:   1
Left:   22 Right:  25 | Token: char                 | Char              | Line:   1
Left:   27 Right:  27 | Token: <id>         k       | Identifier        | Line:   1
Left:   29 Right:  34 | Token: <id>         string  | Identifier        | Line:   1
Left:   36 Right:  36 | Token: <id>         l       | Identifier        | Line:   1
Left:   38 Right:  42 | Token: <id>         float   | Identifier        | Line:   1
Left:   44 Right:  44 | Token: <id>         a       | Identifier        | Line:   1

READLINE: 2       i = i + j + 7
Left:    3 Right:   3 | Token: <id>         i       | Identifier        | Line:   2
Left:    5 Right:   5 | Token: =                    | Assign            | Line:   2
Left:    7 Right:   7 | Token: <id>         i       | Identifier        | Line:   2
Left:    9 Right:   9 | Token: +                    | Plus              | Line:   2
Left:   11 Right:  11 | Token: <id>         j       | Identifier        | Line:   2
Left:   13 Right:  13 | Token: +                    | Plus              | Line:   2
Left:   15 Right:  15 | Token: <int>        7       | INTeger           | Line:   2

READLINE: 3       j = write(i)
Left:    3 Right:   3 | Token: <id>         j       | Identifier        | Line:   3
Left:    5 Right:   5 | Token: =                    | Assign            | Line:   3
Left:    7 Right:  11 | Token: <id>         write   | Identifier        | Line:   3
Left:   12 Right:  12 | Token: (                    | LeftParen         | Line:   3
Left:   13 Right:  13 | Token: <id>         i       | Identifier        | Line:   3
Left:   14 Right:  14 | Token: )                    | RightParen        | Line:   3

READLINE: 4     'f

READLINE: 5
Error happens in line: 5
Charater Format: 'single character/only one whitespace'(e.g.'a' or ' ')Qihongs-MacBook-Pro:~ qihongku
```

g. float_failing_1.x


```
Qihongs-MacBook-Pro:~ qihongkuang$ java -jar lexer.jar float_failing_1.x
Source file: float_failing_1.x
user.dir: /Users/qihongkuang

READLINE: 1    program { int i int j
Left:   0 Right:   6 | Token: program                | Program          | Line:   1
Left:   8 Right:   8 | Token: {                      | LeftBrace        | Line:   1
Left:  10 Right:  12 | Token: int                    | Int              | Line:   1
Left:  14 Right:  14 | Token: <id>         i         | Identifier       | Line:   1
Left:  16 Right:  18 | Token: int                    | Int              | Line:   1
Left:  20 Right:  20 | Token: <id>         j         | Identifier       | Line:   1

READLINE: 2     4. 5. 5.0 254.4520
Left:   1 Right:   2 | Token: <float>      4.        | Float            | Line:   2
Left:   4 Right:   5 | Token: <float>      5.        | Float            | Line:   2
Left:   7 Right:   9 | Token: <float>      5.0       | Float            | Line:   2
Left:  11 Right:  18 | Token: <float>      254.4520  | Float            | Line:   2

READLINE: 3    4...4
Left:   0 Right:   1 | Token: <float>      4.        | Float            | Line:   3
Line: 3 For a float number, the dot must be followed by a digit.
```

3. Implementation of Lexer

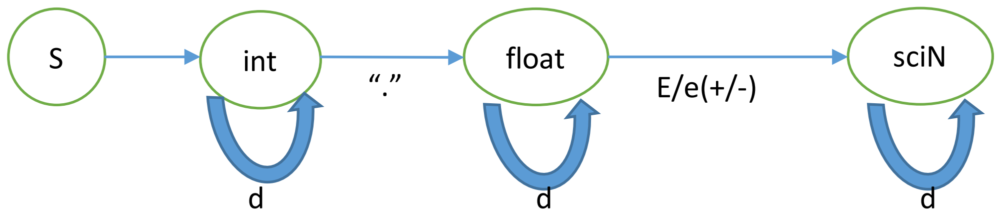
3.1. Introduction of Lexer.java

This java file is used to recognize and return different kinds of tokens. See UML of Lexer.java

Lexer
- atEOF: boolean - ch: char - source: SourceReader - startPosition: int - endPosition: int
+ Lexer(sourceFile: String) + main(args[]: String) static + newIdToken(id: String, startPosition: int, endPosition: int): Token + newNumberToken(number: String, startPosition: int, endPosition: int): Token + newFloatToken(floatNo: String, startPosition: int, endPosition: int): Token + newSciToken(sciNo: String, startPosition: int, endPosition: int): Token + newCharToken(ch: String, startPosition: int, endPosition: int): Token + newStringToken(st: String, startPosition: int, endPosition: int): Token + makeToken(s: String, startPosition: int, endPosition: int): Token + makeScientificToken(number: String): Token + nextToken(): Token

3.2. nextToken()

Developer revised the nextToken() to recognize the String, float, scientific notation, integer and Ascii characters. To handle the float, number and scientific, developer made use of the knowledge of deterministic finite automata in CSC520.



3.2.1. Integer Token

For an integer, as long as all the tokens read are digits and no dots or "E/e" or "+/-" are detected, then the new string will be recognized as integer.

3.2.2. Float Token

For a float, patterns like d+.d* or d*.d+ are all recognized as float as long as only one dot is detected. However, a single "." cannot be verified as a float. So the developer define an integer named dot (int dot = 0;). As long as one dot is detected, then integer dot will be incremented by 1. Next time, if another dot is detected, integer dot is 1 so an exception will be thrown since a float shouldn't include more than one ".".

3.2.3. Scientific Notation Token

For a scientific notation, -2.12E+3, 3.45e5, -3.12e-3 are all valid, but 123.6e+3, 0.3e-12 are not valid as this is not normalized. The scientific notation has two characteristics. First, the first digit must be greater than 0. Second, the second position should be a "." or "E" or "e". That is to say, by the time the Lexer read an "E" or "e", it is possible that the new string (Let's name it sciN.) can be a scientific notation. By this time, either the length of sciN is 1 or the second character of sciN is ".". Moreover, even though, the string satisfies the first two conditions, string like 3.3e-p still cannot be recognized as a scientific notation. If we don't pay attention to this situation, the print out result will be 3.3e- will be verified as scientific notation and p will be verified as <id>. Actually, this is a wrong result. To solve this problem, developer modified the SourceReader.java file. Developer added one new method – the readBack() method. This method change the variable position which is the position method read() is going to read. This method will solve the situation discussed before.

3.2.4. Character Token

For a character, it must be only single one character starts and ends with a single quote. Input like ‘’ is not a valid character. Input like ‘abc’ is also not valid because it only contains one character. Also, ‘a is not valid. For these situations, developer catch exception and return the line the errors happen.

3.2.5 String Token

For a string token, the console will eventually return a string like this – “something character”. That is to say, the return value will include the double quote. Here, developer added method getThisString() to include the first double quote because the type of token affect the increment of the new string. Method getThisString() will help the new string include the left character. Moreover, empty string “” is recognized as a valid string.

3.3 main() method

Developer revised the main method in Lexer.java to print the lexer analysis result. The result should include the line number, left position and right position. Also, it should contain the outcome of each character that lexer read. To make it more readable, developer modified the SourceReader() slightly.

4. Conclusion

1. After revising, Lexer can now recognize float, scientific notation, character and Strings.

2. The revised Lexer passed all the test cases.

3. nextToken() includes too many if statement. If possible, developer will rearrange the code to make it more readable.

4. More token will be added to token file and will be implemented to be recognized.