ICSI 311 Assignment 1 – The Lexer

This assignment is extremely important – (nearly) every assignment after this one uses this one!

If you have bugs or missing features in this, you will need to fix them before you can continue on to new assignments. This is very typical in software development outside of school.

You must submit .java files. Any other file type will be ignored. Especially ".class" files.

You must not zip or otherwise compress your assignment. Blackboard will allow you to submit multiple files.

<u>You must submit buildable .java files for credit.</u>

This assignment **must** have three different source code files.

One file **must** be called Basic.java.

Basic.java **must** contain main. Your main **must** ensure that there is one and only one argument (args). If there are none or more than 1, it **must** print an appropriate error message and exit. That one argument will be considered as a filename. Your main **must** then use File.ReadAllLines to read all of the lines from the file denoted by filename. Your main **must** instantiate one instance of your Lexer class (to be defined below) for each line from ReadAllLines. You **must** parse each line using the lex method of the Lexer class. You **must** take the list of tokens from lex and concatenate it to a single list. If lex throws an exception, you **must** catch the exception, print that there was an exception and lex the next line. You **must** then print each token out (this is a temporary step to show that it works) once the lexing is complete.

One file must be called Token.java. This file **must** contain a Token class. The token class is made up of an instance of an enum and a value string. There **must** be a public accessor for both the enum and the value string; the underlying variables **must** be private. You may create whatever constructors you choose. The enum **must** be defined as containing values appropriate to what we will be processing. The definition of the enum should be public, but the instance inside Token must be private. We will add to this enum in the next several assignments. You will find it helpful to create an appropriate "ToString" overload.

The final file **must** be called Lexer.java. The Lexer class **must** contain a lex method that accepts a single string and returns a collection (array or list) of Tokens. The lex method **must** use one or more state machine(s) to iterate over the input string and create appropriate Tokens. Any character not allowed by your state machine(s) should throw an exception.

For this assignment, the allowable input consists of basic mathematical symbols and numbers such as you might find in Java. +, -, *, / and both positive and negative numbers, including decimals. Allow but ignore spaces/tabs. At the end of each line, add an "EndOfLine" token.

Some examples of valid input and the result output are:

Input	***Output***
an empty line	EndOfLine
5	NUMBER (5) EndOfLine
5.23 – 8.5 + 3	NUMBER(5.23) MINUS NUMBER(8.5) PLUS NUMBER(3) EndOfLine
*8 * -4 + 99999*	NUMBER (8) TIMES NUMBER (-4) PLUS NUMBER (99999) EndOfLine
7 4 3 1	NUMBER(7) NUMBER(4) NUMBER(3) NUMBER(1) EndOfLine
*+ - * /*	PLUS MINUS TIMES DIVIDE EndOfLine

HINTS

Do not wait until the assignment is nearly due to begin. Start early so that you can ask questions.

Write out your state machine(s) before you start coding.

Create some examples and walk through them with your state machine(s)

Test your work thoroughly before handing it in. Trade test cases with your friends to torture each other's code. Do not hand in your test files.

Rubric	Poor	OK	Good	Great
Comments	None/Excessive (0)	"What" not "Why", few (5)	Some "what" comments or missing some (7)	Anything not obvious has reasoning (10)
Variable/Function naming	Single letters everywhere (0)	Lots of abbreviations (5)	Full words most of the time (8)	Full words, descriptive (10)
Basic.java/main	Doesn't exist or named wrong (0)			Exists and named correctly (5)
File reading	Non-Existent (0)	Uses some other mechanism for reading lines from a file(2)		Uses RealAllLines(5)
Calling lex	Non-Existent (0)	Reuses the lexer or doesn't call lex appropriately(3)		Instantiates Lexer and calls lex for each line (5)
Handling exceptions	Don't handle exception (0)			Handles exception and prints message(5)
Printing results	Doesn't print results			Prints results appropriately(5)
Token.java	Doesn't exist(0)	One of: private enum, private value, has accessors (5)	Two of: Exists, private enum, private value, has accessors (7)	Exists, private enum, private value, has accessors (10)
Lexer.java	Doesn't exist or no public lex method(0)			Exists, has public lex method(5)
State machine(s) – symbols handled	Nonexistent or never correct (0)	Some cases handled(7)	Most cases handled(13)	All cases handled(20)
State machine(s) – numbers handled	Nonexistent or never correct (0)	Some cases handled(7)	Most cases handled(13)	All cases handled(20)