Lexical Analyzer

For this milestone you will implement a program to perform lexical analysis for MiniJava. Your lexer will take the name of a text file on disk as an argument. The program will read the file and output the tokens recognized in order – one per line – according to the lexical structure given in the Term Project Description.

There are five *sorts* of tokens in the lexical syntax: *ID, Integer, ReservedWord, Operator, and Delimiter*. When your lexer recognizes a token, it should output the *sort* of token, followed by a comma, a space, then the value of the token. Case, spacing, and punctuation count.

For example, if the input file were:

```
/** This is a test. */
class Test {
	public static void main (String[] args) {
		System.out.println(2 + 13); // cool
	}
}
```

Your lexer should output *exactly* the following lines:

```
ReservedWord, class
ID, Test
Delimiter, {
ReservedWord, public
ReservedWord, static
ReservedWord, void
ReservedWord, main
Delimiter, (
ReservedWord, String
Delimiter, [
Delimiter, ]
ID, args
Delimiter, )
Delimiter, {
ReservedWord, System.out.println
Delimiter, (
```

Integer, 2

Operator, +

Integer, 13

Delimiter,)

Delimiter, ;

Delimiter, }

Delimiter, }

As shown in the example, your program should silently skip whitespace and comments.

DELIVERABLES

Sample Input

You must submit 4 sample input files, along with the expected output, **by Midnight at least three days before** the milestone deadline. These sample input files should simply be placed in your Subversion repository. **Before the deadline, please email me with the location of your sample input files within your repository.** No single input file may be longer than 100 tokens.

Note that sample input does not have to be well-formed MiniJava code. For example,

+-CSSE404

is valid sample input consisting of three tokens.

I will give a set of sample inputs to every team at least two days before the milestone deadline. I will give a full set of sample inputs to every team at class time on the milestone deadline. (The sample inputs will be posted on MOODLE.)

Sample Output and Code

By midnight on the milestone deadline you must commit to your repository:

- the output of your program for each sample input (well labeled so I can match input to output)
- the source code for your program
- a plain text file named Lexer.txt in the MilestonePointers subdirectory of your repository. This file should indicated where to find the source code and program output for your lexer.

Grading

Your grade will be based on whether you submit correct sample inputs and the percentage of the full set of sample input files that your program processes correctly. For each sample input, you'll either get a zero or a one, i.e., no partial credit for processing a portion of a sample. I'll award bonus points for sample inputs that expose bugs in my implementation of the compiler.