# An example of using Java, JLex, Java Cup, and JUnit for developing lexers

1. Get Java
I'm not going to cover that here. Download `http://dlc.sun.com/jdk/j2sdk-1_4_2_07-windows-i586-p.exe` for windows. However, to insure that you have the java compiler in your path, type "`javac`" at a command prompt. You should see something like:

```
Usage: javac <options> <source files>
where possible options include:
  -g                         Generate all debugging info
  -g:none                    Generate no debugging info
…
```

If not, make sure that the directory holding the "`javac`" binary is in your path. On windows, see the instructions at:`http://java.sun.com/j2se/1.4.2/install-windows.html` for how to install Java on windows.

Also: the following instructions are based on unix. To translate to windows command line, substitute the ":" in class paths to ";", and the "/" in directory paths with "\".

2. Prepare directories
Create a directory for holding the tools you're going to acquire. You will probably want to create this directory as a subdirectory of your home directory. For example, make a directory named "`src`" in your home directory.

3. Get JLex
Create a directory for holding JLex in your src directory, "`JLex`". JLex can be downloaded from the JLex web page: `http://www.cs.princeton.edu/~appel/modern/java/JLex/` In particular, the file to download is at: `http://www.cs.princeton.edu/~appel/modern/java/JLex/current/Main.java`
Download this file to your "`src/JLex`" directory.

4. Get JLex/Java Cup example code
Cretate a directory for holding the example code in your src directory, "`minimal`". Download the example source code referenced from the JLex home web page,
`http://www.cs.princeton.edu/~appel/modern/java/CUP/minimal.tar.gz`
Unpack this file. This will create a directory "`MinimalExample`". Rename this to "`Example`".

5. Get Java cup
Make a "`JavaCup`" directory in your src directory for the java cup source from the Java Cup home page, `http://www.cs.princeton.edu/~appel/modern/java/CUP/` Download to the JavaCup directory the source in the file
`http://www.cs.princeton.edu/~appel/modern/java/CUP/java_cup_v10k.tar.gz` Java Cup is now almost ready for use.

6. Build JLex
Now build JLex by compiling it. From your "`src`" directory, use "`javac JLex/Main.java`" to compile JLex.

7. Test Java Cup availablility
Java Cup can be run without changing any environment variables. Try the following:

```
java -cp /Users/jones/src/JavaCup java_cup.Main blah.cup

Unable to open "blah.cup" for input
```

```
Usage: java_cup [options] [filename]
  and expects a specification file on standard input if no filename is given.
…
```

The "`-cp ...`" option tells java to look in the "`.../JavaCup`" directory to find class files to execute. The "`java_cup.Main`" argument tells java which class to find the beginning point of the program in.

8. Test JLex availablility
JLex can be run without changing any environment variables. Try the following:

```
java -cp /Users/jones/src JLex.Main blah.lex
Exception in thread "main" java.io.FileNotFoundException: blah.lex (No such
file or directory)
        at java.io.FileInputStream.open(Native Method)
…
```

8. (optional, for Unix) Make JLex and Java Cup commands
I find it very useful to make Java Cup and JLex into commands. On unix, I have a "bin" directory in my home directory for holding commands:

```
$ cat ~/bin/java_cup
java -cp ~/src/java_cup_v10k java_cup.Main $*
$ cat ~/bin/jlex
java -cp ~/src JLex.Main $*
```

Make sure that "`~/bin`" is in your `PATH` environment variable and that the files are user executable.

At this point, your src directory should look something like this:

```
$ ls -R src
JLex    JavaCup minimal

src/JLex:
CAccept.class                   CNfaPair.class
...

src/JavaCup:
CHANGELOG               README                  java_cup_v10k.tar.gz
...

src/JavaCup/gnuwin:
README          makefile

src/JavaCup/java_cup:
CUP$parser$actions.class        parse_action_table.class
Main.class                      parse_action_table.java
...

src/JavaCup/java_cup/runtime:
Scanner.class                   lr_parser.class
...

src/JavaCup/java_cup/simple_calc:
```

```
CUP$parser$actions.class        parser.java
Main.class                      scanner.class
...

src/minimal:
Example  minimal.tar.gz

src/minimal/Example:
README          minimal.cup     minimal.lex
```

9. Build Minimal
Make sure you in in the directory with the minimal example source files, `/Users/jones/School/src/minimal/Example` It will have the contents as given immediately above. Run JLex on the jlex specification file: `java -cp /Users/jones/src JLex.Main minimal.lex` This will produce the following output:

```
Processing first section -- user code.
Processing second section -- JLex declarations.
Processing third section -- lexical rules.
Creating NFA machine representation.
NFA comprised of 30 states.
Working on character classes.::::::::.
NFA has 10 distinct character classes.
Creating DFA transition table.
Working on DFA states............
Minimizing DFA transition table.
10 states after removal of redundant states.
Outputting lexical analyzer code.
```

The new contents of the directory now include a new file, `minimal.lex.java`. When we examine the `minimal.lex.java` file, we note that it declares that it is a member of the `Example` package. Therefore, we need to change to the `src/minimal` directory before compiling. If we compile this file using the command: `javac Example/minimal.lex.java` we get fifteen errors, as we haven't told the java compiler where to get the java_cup.runtime.Symbol file, as indicated by the error message:

```
minimal.lex.java:283: cannot resolve symbol
symbol  : class Symbol
location: class Example.Yylex
                                                { return new
Symbol(sym.NUMBER, new Integer(yytext())); }
```

Also, we don't have a class file for "`sym`", as indicated by the error message:

```
minimal.lex.java:283: cannot resolve symbol
symbol  : variable sym
location: class Example.Yylex
                                                { return new
Symbol(sym.NUMBER, new Integer(yytext())); }
```

Let's fix these problems one at a time. We can either add the java cup directory to the `CLASSPATH` environment variable, or just specify it on the command line. Taking the second approach, we issure the command:

```
javac -classpath /Users/jones/src/JavaCup minimal.lex.java
```

This reduces the number of errors to six and all are due to the missing "`sym`" class definition. Normally, the sym file is generated by Java cup, by using the "`terminal`" declarations. This file consists of a set of constant definitions, defining the integer the represents the token type. Since we aren't concerned with Java Cup parsing yet, simply create a `sym.java` file with the following contents:

```
package Example;

public class sym {
  /* terminals */
  public static final int RPAREN = 6;
  public static final int error = 1;
  public static final int PLUS = 3;
  public static final int NUMBER = 7;
  public static final int SEMI = 2;
  public static final int LPAREN = 5;
  public static final int TIMES = 4;
  public static final int EOF = 0;
}
```

Now, reissue the compilation command, but also include the current directory by adding a "." to search in the current directory:

```
javac -classpath /Users/jones/src/JavaCup:. Example/minimal.lex.java
```

This should compile with no errors.

10. Get JUnit
On the JUnit home page, `http://www.junit.org/index.htm` is a link to the download page of JUnit: `http://prdownloads.sourceforge.net/junit/junit3.8.1.zip?download` From that page, choose one of the download mirrors and download it to your src directory. When it is unzipped, you should have a directory `junit3.8.1` in your src directory.

12. Test minimal with JUnit test, `MinimalLexerTest.java`
Use the following command to compile the MinimalLexerTest.java file:

```
javac -classpath
/Users/jones/src/junit3.8.1/junit.jar:.:/Users/jones/src/JavaCup
Example/MinimalLexerTest.java
```

Use the following command to run the `MinimalLexerTest.java` file:

```
java -cp /Users/jones/src/junit3.8.1/junit.jar:.:/Users/jones/src/JavaCup
Example.MinimalLexerTest
```

You should see the following output, indicating that no errors were found:

```
..
Time: 0.075

OK (2 tests)
```