COP5555 Spring 2015
Assignment 2

Due: Wed. Feb 11 at 10am.

1. Is the given Phrase Structure LL(1)? If not, give an example where the necessary and sufficient conditions are violated. Determine PREDICT(Factor) and PREDICT(Statement) for the language with the attached CFG.
2. Correct any errors in your Scanner from Assignment 1.
3. Using your Scanner from Assignment 1, implement a parser and Junit tests for the attached phrase structure (i.e. context-free grammar) specification. Your parser should be called SimpleParser and complete the given class to handle the entire grammar. Add additional tests to the provided TestSimpleParser class to thoroughly test your parser, including checking that erroneous input is detected. In this assignment, the parser should return silently if the input is allowed by the grammar. If an error is detected, your parser should throw a SyntaxException and terminate. If you use the given match methods to handle terminals, the details of this are already done for you. **Do not remove or change the signatures of anything that is given or change the package declaration. Do not use static variables (exception: constants, which are static final are OK.)**

Submit two files:

1. A pdf file with your answer to question 1,

2. A jar file called cop5555.jar with containing the Java source files Scanner.java, TokenStream.java, SimpleParser.java, TestSimpleParser. **Make sure that your jar file contains your sources**.

> *To ensure a successful submission, double check that your files were actually submitted. Also, double check the contents of your jar file. If you generate the jar file using the eclipse export function, be aware that by default, it only includes class files in the jar; you will need to explicitly tell it to include the source files. To grade, we will unjar, compile, and execute your code with your test cases, and with ours.* ***Make sure that you do not break the grading script.***Hints:

The parseCorrectInput and parseIncorrectInput methods in the provided TestSimpleParser class show how the scanner, parser, and TokenStream class are glued together.

The SimpleParser class as given should pass two of the provided test cases: almostEmpty, and almostEmptyIncorrect.

As with the scanner, it is best to work incrementally. You will probably find it easier to create your own test cases as you go along, than to try to use the given ones to drive development.