COP5555 Spring 2015
Assignment 1

Due: Wed. January 28 at 10am.

1. Exercise 2f in Scott, Chapter 2.
2. Implement a Scanner and Junit tests for the attached lexical specification. The last token should be an EOF token, thus every program, even an empty one, should contain at least one token. Errors should be handled by constructing a token with an appropriate error kind and then continuing processing. Your scanner should handle arbitrary input without crashing.

   The overall structure is similar to that used by eclipse IDEs. The basic idea is that TokenStream class is initialized with a char array containing the input characters. It also contains an initially empty Token lists tokens. Your scanner takes a TokenStream instance and scans its input characters, inserting an object for each recognized Token into the tokens list.

   The TokenStream class along with two inner class Token has been provided. You should not change anything in the TokenStream class for this assignment. For convenience, TokenStream has several constructors that get the input from different sources. (For example, if you want to read from a file, you can pass it a FileReader, or if the input is in a String, pass it the String). Using a String input is convenient for testing, but you must be careful about escape characters and String literarals in Java.

   Use the given Scanner class (which will not compile as is) as a starting point and add additional methods and fields as required. **Do not remove or change the signatures of anything that is given or change the package declaration. Do not use static variables.**

   Add additional tests to the provided TestScanner class to thoroughly test your Scanner. We may use your test cases in addition to our own to test everyone's submissions. Use of student test classes is experimental. Points MAY be deducted for incorrect tests. Extra credit MAY be given for tests that find errors others miss.

   The Token class is an inner class of the TokenStream class. Token is a non-static inner class, which means that its instances are connected to an instance of the enclosing class, in this case TokenStream. To create a Token instance, you need a TokenStream instance (call it stream) and then say something like Token t = stream.new Token(….).

Submit two files:

1. A pdf file containing your answers to question 1.

2. A jar file called COP5555.jar containing the Java source files Scanner.java, TestScanner.java, and TokenStream.java (which should be exactly as given).

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:

Here is how the pieces fit together:

Create a TokenStream instance, passing the input source to the constructor.

```
TokenStream stream = new TokenStream(input);
```

Instantiate a Scanner object, passing the TokenStream to the constructor

```
Scanner scanner = new Scanner(stream);
```

Call the scan method to scan the input and generate the token list

```
scanner.scan();
  i.  later the
```

Your test class, later your Parser, will get the tokens from the TokenStream by repeatedly calling the nextToken method.

```
Token t = stream.nextToken();
```

The Token class is an inner class of the TokenStream class. Token is a non-static inner class, which means that its instances are connected to an instance of the enclosing class, in this case TokenStream. To create a Token instance, you need a TokenStream instance (call it stream) and then say something like

```
Token t = stream.new Token(….);
```

The provided TestScanner class includes a few JUnit tests so that you can ensure that things are set up properly. It uses a variety of ways to formulate a test. Use your favorite one for your own test cases.

You can run TestScanner in eclipse by right clicking on the file name, selecting Run As in the menu, and then choose JUnit Test.

Work incrementally. First implement **and test** an empty program, then add functionality and tests for the single character tokens, then the two character tokens, etc.

When you get a chance, read up on test-driven development and practice as appropriate in this project. Be able to talk about it on job interviews.