Syntax Analyzer Assignment

Project objective:

Implement a recursive descent parser for the Ada Language given on page 3.

Design:

1. The code for the lexical analyzer is the following three files provided in the assignment in Blackboard:
   a. Download the file LexicalAnalyzer.java containing the lexical analyzer class and include it in your project. You cannot modify any of the code in this file. You must use this enumeration class in this assignment.
   b. Download the file Symbol.java containing the enumerated type which contains the token symbols for all the lexemes in ADALS1 and include it in your project. You cannot modify any of the code in this file. You must use this enumeration class in this assignment.
   c. Download the file Token.java containing the token class and include it in your project. You cannot modify any of the code in this file. You must use this class in this assignment.

2. The ADA source program is either syntactically correct or contains one or more syntax errors.
   a. If the ADA source program is syntactically correct, then the parser will reach the token EOI. In this case, have the parser print "The program is syntactically correct." on the line after the last line of the program.
   b. If the ADA source program contains syntax errors then the parser (in this assignment) will only find the first syntax error, print the appropriate error message for that syntax error, and then terminate. Print the error message on the two lines below the program line containing the syntax error. Also, use the **^** character to point to the location in the line where the syntax error possibly occurred. See below for an example. The parser terminates right away by using System,exit().

      Error message example:

```
x, y z : integer;
    ^
Error: expecting a comma.
```

3. Create an Errors class to contain all the possible error messages for the parser of this assignment. Create a method called error. The method error has one parameter, an integer, which indicates the error message to print. The public method getCurrCharPosNum() in the LexicalAnalyzer class returns the position number (within the current line) where the lexical analyzer begins searching for the next token the next time its method getToken executes. The error method uses the position number to print the appropriate number of blanks before the **^** character to place it at the location in the line where the syntax error possibly occurred. The last statement the error method executes is System,exit().

4. Parser is the main class of ADALS1's syntax analyzer. You must declare private all the data members of the Parser class. You must declare private all the methods in the Parser class, except for the constructor and the method **Start**. The constructor has one parameter, the lexical analyzer object. You will create one method for each nonterminal symbol in the grammar except for <IDENTLIST>. You can create any additional methods for your parser, as needed. You can create any additional classes for your parser, as needed.

5. Create a driver class and make the name of the driver class **Assignment2** containing only one method:
   `public static void main(String args[]).`
   The main method receives, via the command line arguments, the name of the Ada program source code file that your parser processes. The main method itself is fairly short containing only the code to 1) create the lexical analyzer object, passing into it the filename of the Ada program source code file, 2) create the parser object, passing into it the lexical analyzer object, and 3) calling the method **Start** of the parser object to start the parsing process.

   The command to launch your program, assuming example1.ada contains an Ada program, is:
   `java Assignment2 example1.ada`

6. You must declare public each class you create which means you define each class in its own file.

7. You must declare private all the data members in every class you create.

8. You cannot use extends in this assignment to extend any class.

9. **Tip:** Make your program as modular as possible, not placing all your code in one .java file. You can create as many classes as you need in addition to the class described above. Methods being reasonably small follow the guidance that "A function does one thing and does it well." You will lose a lot of points for code readability if you do not make your program as modular as possible. But do not go overboard on creating classes and methods. Your common sense guides your creation of classes and methods.

10. Do **NOT** use your own packages in your program. If you see the keyword **package** on the top line of any of your .java files, then you created a package. Create every .java file in the **src** folder of your Eclipse project, if you're using Eclipse.

11. Do **NOT** use any graphical user interface code in your program!

12. Do **NOT** type any comments in your program. If you do a good job of programming by following the advice in number 9 above, then it will be easy for me to determine the task of your code.

ADALS1 Grammar

```
<START> → procedure ident is <DECPART> begin <SEQOFSTMT> end ; EOI

<SEQOFSTMT> → <STATEMENT> { <STATEMENT> }

<DECPART> → { <OBJECTDEC> }

<OBJECTDEC> → <IDENTLIST> : ( boolean | integer ) ;

<IDENTLIST> → ident { , <IDENTLIST> }

<STATEMENT> → null ;
<STATEMENT> → ident := <EXPRESSION> ;
<STATEMENT> → if <CONDITION> then <SEQOFSTMT>
                          [ else <SEQOFSTMT> ] end if ;
<STATEMENT> → while <CONDITION> loop <SEQOFSTMT> end loop ;
<STATEMENT> → ( get | put ) ( <IDENTLIST> ) ;
<STATEMENT> → newline ;

<CONDITION> → <EXPRESSION>

<EXPRESSION> → <SIMPEXPR> [ ( = | /= | < | <= | > | >= ) <SIMPEXPR> ]

<SIMPEXPR> → <TERM> { ( + | - ) <TERM> }

<TERM> → <PRIMARY> { ( * | / | rem ) <PRIMARY> }

<PRIMARY> → ( <EXPRESSION> ) | ident | numlit | true | false
```

Grading Criteria:

The assignment is worth a total of 20 points, broken down as follows:

1. If your code does not implement the task described in this assignment then the grade for the assignment is zero.
2. If your program does not compile successfully then the grade for the assignment is zero.
3. If your program produces runtime errors which prevents the grader from determining if your code works properly then the grade for the assignment is zero.

Important: Make sure your project compiles and runs via the command line using the Java JDK because I will not use any other Java development tool to compile and run your project code.

If the program compiles successfully and executes without significant runtime errors then the grade computes as follows:
Followed proper submission instructions, 4 points:

1. Was the file submitted a zip file.
2. The zip file has the correct filename.
3. The contents of the zip file are in the correct format.
4. The keyword **package** does not appear at the top of any of the .java files.

Code implementation and Program execution, 12 points:

- The driver file has the correct filename, **Assignment2.java** and contains only the method **main** performing the exact tasks as described in the assignment description.
- The code performs all the tasks as described in the assignment description.
- The code is free from logical errors.
- Program input, the program properly processes the input.
- Program output, the program produces the proper results for the assignment.

Code readability, 4 points:

- Good variable, method, and class names.
- Variables, classes, and methods that have a single small purpose.
- Consistent indentation and formatting style.
- Reduction of the nesting level in code.

Late submission penalty: assignments submitted after the due date are subjected to a 2 point deduction for each day late.

Late submission policy: you **CAN** submit your assignment early, before the due date. You are given plenty of time to complete the assignment well before the due date. Therefore, I do **NOT** accept any reason for not counting late points if you decide to wait until the due date (and the last possible moment) to submit your assignment and something happens to cause you to submit your assignment late. I only use the date submitted, ignoring the time as well as Blackboard's late submission label.

Submission Instructions:

Go to the folder containing the .java files of your assignment and select all (and **ONLY**) the .java files which you created for the assignment in order to place them in a Zip file. The file can **NOT** be a **7z** or **rar** file! Then, follow the directions below for creating a zip file depending on the operating system running on the computer containing your assignment's .java files.

Creating a Zip file in Microsoft Windows (any version):

1. Right-click any of the selected .java files to display a pop-up menu.
2. Click on **Send to**.
3. Click on **Compressed (zipped) Folder**.
4. Rename your Zip file as described below.
5. Follow the directions below to submit your assignment.

Creating a Zip file in Mac OS X:

1. Click **File** on the menu bar.
2. Click on **Compress ? Items** where ? is the number of .java files you selected.
3. Mac OS X creates the file **Archive.zip**.
4. Rename **Archive** as described below.
5. Follow the directions below to submit your assignment.

Save the Zip file with the filename having the following format:

your last name,
followed by an underscore _,
followed by your first name,
followed by an underscore _,
followed by the word **Assignment2**.

For example, if your name is John Doe then the filename would be: **Doe_John_Assignment2**

Once you submit your assignment you will not be able to resubmit it!
Make absolutely sure the assignment you want to submit is the assignment you want graded.
There will be **NO** exceptions to this rule!

You will submit your Zip file via your CUNY Blackboard account.
The only accepted submission method!

Follow these instructions:

Log onto your CUNY Blackboard account.
Click on the CSCI 316 course link in the list of courses you are taking this semester.
Click on **Assignments** tab in the red area on the left side of the webpage.
You will see the **Syntax Analyzer Assignment**.
Click on the assignment.
Upload your Zip file and then click the submit button to submit your assignment.

Due Date: Submit this assignment on or before 11:59 p.m. Tuesday, November 16, 2020.