Homework 3

Due: 4 May 2009 (Source code and report)
Demo: 7 May 2009

Write an interpreter for the language defined below. The main technical challenge of this project is to write a parser for the language by using the lexical analyzer you have already developed. After obtaining the parse tree it should be rather straightforward to run it.
The program should accept a text file (source file) as input and compute an integer value as output. The text file is meant to contain a script that describes how a certain arithmetic calculation is to be made. The name and path of the source file should be entered by the user, and then the program should parse the file, run the script and display the result. (or an error message instead, in case of any error)

Language Definition:

- A program is composed of multiple lines of code each of which is terminated by a semicolon.
- Every line of code is an assignment statement with a variable name on the left hand side and an arithmetic expression on the right hand side. The assignment symbol is ":=".
- The arithmetic expression is composed of integer constants (including negative numbers), integer variables (written with alphabetical characters only and case sensitive), the operators +, -, * (for addition, subtraction and multiplication respectively) and left and right parentheses '(' ')'.
- Parentheses can be nested and the operator precedence of conventional arithmetics is supported.
- Variable usage does not requite any declaration as the variables are assumed to be declared the first time they are used.
-The script returns the current value of a predefined variable called "result".

Code Example:

```
aVar :=11+ 5;
myVar:= 32;
result := (aVar-myVar)*3 ;
```

Your interpreter should catch all the necessary syntactic and semantic errors and issue suitable error messages. These include lexical errors, parser errors and run time errors. The errors to be caught include (but are not limited to) the following:

Lexical Errors

- Characters that cannot appear in any token in our source language, such as < or #.
- Integer constants out of bounds (integer constants bigger than Java maxInt or smaller than Java minInt).
- Identifier names that are too long (maximum length is 32 characters).

Parser Error Examples:

`c:=(a+b)*2)*24;`	(paranthesis mismatch at line 1 column 11)
`22:=32;`	(identifier expected on the left of the assignment operator)
`myVar+=1+b;`	(assignment operator expected at line 3 column 6)
`b:=b+1`	(Missing end of line mark ; at line 4)

Run-time Errors

- Variable used before it was initialized.
- Program exits with no assignment made to "result"
- Integer value out of bounds.

Errors should be reported with the corresponding line and column position as good compilers do.

- This project can be carried out in teams of 2 students from the same lab group. It may also be done individually.
- The program should be written in Java by using no other library or framework than the standard Java libraries.
- After the completion of this project a brief demonstration (demo) is going to be made to your lab instructor. (one of Önder Gürcan, Bekir Afşar or Ahmet Egesoy according to your lab group)
- The source files will be e-mailed and handed as hard copy to your lab instructor. The source files should be in conventional Java source file formats (not .doc and definitely not in .docx format) and a brief (1 or 2 page) report in .pdf format should be included in order to explain how your program works, mentioning some test data (sample scripts) that it has been proved to work with.
- Late submissions will be subject to a 10% decrease penalty in grade for each day.

Good luck