Assignment 2

Due: 13 February at 11:59pm

Implement a recursive descent parser for the following context-free grammar:

program ::= IDENT block
program ::= IDENT param_dec (, param_dec)* block
paramDec ::= (KW_URL | KW_FILE | KW_INTEGER | KW_BOOLEAN) IDENT
block ::= { (dec | statement) * }
dec ::= (KW_INTEGER | KW_BOOLEAN | KW_IMAGE | KW_FRAME) IDENT
statement ::= OP_SLEEP expression ; | whileStatement | ifStatement | chain ; | assign ;
assign ::= IDENT ASSIGN expression
chain ::= chainElem arrowOp chainElem (arrowOp chainElem)*
whileStatement ::= KW_WHILE (expression) block
ifStatement ::= KW_IF (expression) block
arrowOp ::= ARROW | BARARROW
chainElem ::= IDENT | filterOp arg | frameOp arg | imageOp arg
filterOp ::= OP_BLUR |OP_GRAY | OP_CONVOLVE
frameOp ::= KW_SHOW | KW_HIDE | KW_MOVE | KW_XLOC |KW_YLOC
imageOp ::= OP_WIDTH |OP_HEIGHT | KW_SCALE
arg ::= ε | (expression (,expression)*)
expression ::= term (relOp term)*
term ::= elem (weakOp elem)*
elem ::= factor (strongOp factor)*
factor ::= IDENT | INT_LIT | KW_TRUE | KW_FALSE
| KW_SCREENWIDTH | KW_SCREENHEIGHT | (expression)
relOp ::= LT | LE | GT | GE | EQUAL | NOTEQUAL
weakOp ::= PLUS | MINUS | OR
strongOp ::= TIMES | DIV | AND | MOD

- Use the provided Parser.java and ParserTest.java as a starting point. Your Scanner.java from assignment 1 (with any errors corrected) will also be needed.
- The parser should simply determine whether the given sentence is legal or not. If not, the parser should throw a SyntaxException. If the sentence is legal, the parse method should simply return.
- Use the approach described in the lectures to systematically build the parser. Identify whether the language is LL(1) or not.

Turn in a jar file containing your source code Parser.java, Scanner.java, and ParserTest.java.

Your ParserTest will not be graded, but may be looked at in case of academic honesty issues. We will subject your parser to our set of unit tests and your grade will be determined solely by how many tests are passed. Name your jar file in the following format:
firstname_lastname_ufid_hw2.jar

Additional requirements:

- This code must remain in package cop5556sp17(case sensitive): do not create additional packages.
- Names (of classes, method, variables, etc.) in starter code must not be changed.
- Unless otherwise specified, your code should not import any classes other than those from the standard Java distribution or your Scanner.java.

Submission Checklist

See the checklist from Assignment 1.

Comments and suggestions:

- The given Parser.java and ParserTest.java should compile correctly when combined with your Scanner.java from Assignment 1. When executed, only one test will pass, but all should pass in your completed parser.
- Work incrementally. Note that you can call the routines corresponding to fragments of the grammar in Junit tests. For example, see the testFactor test in the provided code.
- It is useful when working incrementally to ensure that you are not calling an unimplemented procedure without realizing it. To this end, the sample code throws an UnimplementedFeatureException in every method that has not yet been implemented. After all of your methods have been implemented, you may delete the class for this exception to clean up your code.
- You will want to provide better error messages than given in the sample code. In particular, you will want to output the location of the offending token.