Part 2 - Parser

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

dec ::= ( KW\_INTEGER | KW\_BOOLEAN | KW\_IMAGE | KW\_FRAME) IDENT

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

arrowOp ∷= ARROW | BARARROW

relOp ∷= LT | LE | GT | GE | EQUAL | NOTEQUAL

weakOp ∷= PLUS | MINUS | OR

strongOp ∷= TIMES | DIV | AND | MOD

paramDec ::= ( KW\_URL | KW\_FILE | KW\_INTEGER | KW\_BOOLEAN) IDENT

factor ∷= IDENT | INT\_LIT | KW\_TRUE | KW\_FALSE

| KW\_SCREENWIDTH | KW\_SCREENHEIGHT | **(** expression **)**

elem ∷= factor ( strongOp factor)\*

term ∷= elem ( weakOp elem)\*

expression ∷= term ( relOp term)\*

statement ::= OP\_SLEEP expression **;** | whileStatement | ifStatement | chain **;** | assign **;**

whileStatement ::= KW\_WHILE **(** expression **)** block

ifStatement ::= KW\_IF **(** expression **)** block

block ::= **{** ( dec | statement) \* **}**

program ::= IDENT block

program ::= IDENT param\_dec ( **,** param\_dec )\* block

assign ::= IDENT ASSIGN expression

chain ::= chainElem arrowOp chainElem ( arrowOp chainElem)\*

chainElem ::= IDENT | filterOp arg | frameOp arg | imageOp arg

arg ::= ε | **(** expression ( ,expression)\* **)**

* 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.