Assignment 3

Due: 27 February at 11:59pm

Modify your parser from Assignment 2 to return an abstract syntax tree specified by the following abstract syntax:

Program ::= List<ParamDec> Block
ParamDec ::= type ident
Block ::= List<Dec> List<Statement>
Dec ::= type ident
Statement ::= SleepStatement | WhileStatement | IfStatement | Chain
| AssignmentStatement
SleepStatement ::= Expression
AssignmentStatement ::= IdentLValue Expression
Chain ::= ChainElem | BinaryChain
ChainElem ::= IdentChain | FilterOpChain | FrameOpChain | ImageOpChain
IdentChain ::= ident
FilterOpChain ::= filterOp Tuple
FrameOpChain ::= frameOp Tuple
ImageOpChain ::= imageOp Tuple
BinaryChain ::= Chain (arrow | bararrow) ChainElem
WhileStatement ::= Expression Block
IfStatement ::= Expression Block
Expression ::= IdentExpression | IntLitExpression | BooleanLitExpression
| ConstantExpression | BinaryExpression
IdentExpression ::= ident
IdentLValue ::= ident
IntLitExpression ::= intLit
BooleanLitExpression ::= booleanLiteral
ConstantExpression ::= screenWidth | screenHeight
BinaryExpression ::= Expression op Expression
Tuple ::= List<Expression>
op ::= relOp | weakOp | strongOp
type ::= integer | image | frame | file | boolean | url

In this syntax, upper case names correspond to classes representing nodes in the AST. Lower case names like op or ident will be represented by their Tokens. The correspondence between the concrete and abstrax syntax should be clear, except possibly that arg in the concrete syntax from Assignment 2 will correspond to a Tuple here.

The AST classes have been provided for you in the attached jar file. These classes are instrumented to use with the Visitor pattern which will be used in later assignments. The ASTNode class is the abstract superclass of all the other classes provided and has one field, a Token firstToken. This should be set to the first token in the construct of all nodes. In later assignments, you can use the firstToken to identify the location in the source code where errors occur.

In some cases there is only one Token inside the firstToken will be the only field. For example IdentExpression contains a single ident which is also the firstToken. Also, in Program, the program name is stored in firstToken.

In Chain, the arrow is left associative.

A few junit tests have been provided in ASTTest.java. This will not compile with your Parser until you modify at least the expression() method to return an ASTNode.

Turn in a jar file containing your source code Parser.java, Scanner.java, all of the AST nodes, and your ASTTest.java.

Although in this assignment, you probably will not need to change any of the provided classes, you will need to in later assignments, so to keep our process consistent, include them in this assignment as well.

Your ASTTest will not be graded, but may be looked at in case of academic honesty issues. We will subject your parser to our set of junit 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_hw3.jar

Additional requirements:

- Your code must remain in package cop5556sp17(case sensitive): do not create additional packages.
- The provided classes are in package cop5556sp17.AST. These should be in a folder called AST inside the cop5556sp17 folder
- Names (of classes, method, variables, etc.) in the provided AST classes 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:

- Work incrementally, starting with small constructs and moving to bigger ones, and adding tests each time. As in assignment 2, you can call the routines corresponding to fragments of the grammar in Junit tests. For example, see the testFactor test in the provided 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.
- To write a junit test, you will need to know the structure of the AST that should be created. I recommend using assertEquals(ExpectedClass.class, obj.getClass()); to check that obj is an instance of ExpectedClass as this will give you better error messages from JUnit than the alternative assertTrue(obj instanceof ExpectedClass).