Assignment 3

Due:  12 October at 11:59pm

Modify your parser from Assignment 2 so that it returns an Abstract Syntax Tree.

|  |  |
| --- | --- |
| **Concrete Syntax** | **ASTNode** |
| Program ::= IDENTIFIER ( Declaration SEMI | Statement SEMI )\* | Program |
| Declaration :: = VariableDeclaration | ImageDeclaration | SourceSinkDeclaration |  |
| VariableDeclaration ::= VarType IDENTIFIER ( OP\_ASSIGN Expression | ε ) | Declaration\_Variable |
| VarType ::= KW\_int | KW\_boolean |  |
| SourceSinkDeclaration ::= SourceSinkType IDENTIFIER OP\_ASSIGN Source | Declaration\_SourceSink |
| Source ::= STRING\_LITERAL | Source\_StringLiteral |
| Source ::= OP\_AT Expression | Source\_CommandLIneParam |
| Source ::= IDENTIFIER | Source\_Ident |
| SourceSinkType := KW\_url | KW\_file |  |
| ImageDeclaration ::= KW\_image (LSQUARE Expression COMMA Expression RSQUARE | ε) IDENTIFIER ( OP\_LARROW Source | ε ) | Declaration\_Image |
| Statement ::= AssignmentStatement  | ImageOutStatement  | ImageInStatement |  |
| ImageOutStatement ::= IDENTIFIER OP\_RARROW Sink | Statement\_Out |
| Sink ::= IDENTIFIER | Sink\_Ident |
| Sink ::= KW\_SCREEN | Sink\_SCREEN |
| ImageInStatement ::= IDENTIFIER OP\_LARROW Source | Statement\_In |
| AssignmentStatement ::= Lhs OP\_ASSIGN Expression | Statement\_Assign |
| Expression ::= OrExpression OP\_Q Expression OP\_COLON Expression | Expression\_Conditional |
| Expression ∷= OrExpression |  |
| OrExpression ::= AndExpression ( OP\_OR AndExpression)\* | Expression\_Binary |
| AndExpression ::= EqExpression ( OP\_AND EqExpression )\* | Expression\_Binary |
| EqExpression ::= RelExpression ( (OP\_EQ | OP\_NEQ ) RelExpression )\* | Expression\_Binary |
| RelExpression ::= AddExpression ( ( OP\_LT | OP\_GT | OP\_LE | OP\_GE ) AddExpression)\* | Expression\_Binary |
| AddExpression ::= MultExpression ( (OP\_PLUS | OP\_MINUS ) MultExpression )\* | Expression\_Binary |
| MultExpression := UnaryExpression ( ( OP\_TIMES | OP\_DIV | OP\_MOD ) UnaryExpression )\* | Expression\_Binary |
| UnaryExpression ::= OP\_PLUS UnaryExpression  | OP\_MINUS UnaryExpression  | UnaryExpressionNotPlusMinus | Expression\_Unary |
| UnaryExpressionNotPlusMinus ::= OP\_EXCL UnaryExpression | Expression\_Unary |
| UnaryExpressionNotPlusMinus ::= Primary |  |
| UnaryExpressionNotPlusMinus ::= IdentOrPixelSelectorExpression |  |
| UnaryExpressionNotPlusMinus ::= KW\_x | KW\_y | KW\_r | KW\_a | KW\_X | KW\_Y | KW\_Z | KW\_A | KW\_R | KW\_DEF\_X | KW\_DEF\_Y | Expression\_PredefinedName |
| Primary ::= INTEGER\_LITERAL | Expression\_IntLit |
| Primary LPAREN Expression RPAREN |  |
| Primary ::= FunctionApplication |  |
| Primary ::= BOOLEAN\_LITERAL | Expression\_BooleanLit |
| IdentOrPixelSelectorExpression::= IDENTIFIER LSQUARE Selector RSQUARE | Expression\_PixelSelector |
| IdentOrPixelSelectorExpression::= IDENTIFIER | Expression\_Ident |
| Lhs::= IDENTIFIER ( LSQUARE LhsSelector RSQUARE | ε ) | LHS |
| FunctionApplication ::= FunctionName LPAREN Expression RPAREN | Expression\_FunctionAppWithExprArg |
| FunctionApplication ::= FunctionName LSQUARE Selector RSQUARE | Expression\_FunctionAppWithIndexArg |
| FunctionName ::= KW\_sin | KW\_cos | KW\_atan | KW\_abs  | KW\_cart\_x | KW\_cart\_y | KW\_polar\_a | KW\_polar\_r |  |
| LhsSelector ::= LSQUARE ( XySelector | RaSelector ) RSQUARE |  |
| XySelector ::= KW\_x COMMA KW\_y | Index |
| RaSelector ::= KW\_r COMMA KW\_A | Index |
| Selector ::= Expression COMMA Expression | Index |

* A starter implementation of ParserTest.java with a few test cases has been provided. You will need to rename your parser from Assignment 2 to Parser.java and modify it so that the parse method returns an instance of cop5556fa17.AST.Program. For example:

**public** Program parse() **throws** SyntaxException {

Program p = program();

matchEOF();

**return** p;

}

The expression method must return an Expression object.

* All of the classes for AST nodes have been provided in the jar file. You should not modify any of these classes in this assignment (although you will modify them later.) These classes include the boilerplate code needed to implement the visitor pattern, but that will not be needed until assignment 4, so ignore it for now.
* The abstract superclass of all of the abstract syntax tree nodes is ASTNode.java. It contains a single field Token firstToken which must be provided in the constructor. This is there to connect the AST with the program source for error messages later and should be the first Token in the construct being parsed.

**Turn in a jar file containing your source code for Parser.java, Scanner.java, and ParserTest.java.  Also include the source for the provided classes AST node classes so that your jar file is complete.**

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\_hw3.jar*

**Additional requirements:**

* Your parser should remain in package cop5556fa17(case sensitive) and the parse and expression methods must be public. The former should return a Program object, the latter an Expression object.
* Your code should not import any classes other than those from the standard Java distribution, Scanner.java, or the provided cop5556fa17.AST package
* All code, including the Scanner code and the Parser code you are using as a starting point must be your own work developed by you this semester.
* Your Parser should throw exceptions for exactly the same input as a correctly implemented SimpleParser from Assignment 2 would. An AST will only be returned for valid input.

**Submission Checklist**

See the checklist from Assignment 1.

**Comments and suggestions:**

Don’t attempt to do this assignment before you have looked at the relevant lecture.

Spend some time understanding the structure of the provided code. What is the inheritance hierarchy? How does that relate to the syntax?

You will need to look inside each class in order to see which fields it contains and what the constructor expects. If a field is optional in the syntax and is not provided in the input, you should set the corresponding field in the AST node to null. (The exception is the list of statements and declarations in Program. If there are no statements of declarations, the list should be empty, but not null.)

Each class contains methods visit, hashCode equals, and toString. The latter 3 were generated by eclipse; the visit method was systematically constructed to support the visitor pattern. It may be useful for you to use some of these methods (like toString) but otherwise you can ignore them.