Assignment 3: Parser

Nehemya McCarter-Ribakoff, ID# 916571687

GitHub repo: github.com/nemimccarter/SFSU-CSC-413/assignment-3-nemimccarter

Introduction

This program parses .x files and uses the parsed content to build an abstract syntax. Parsing involves following a provided set of rules (a grammar) to detail a program's logical structure, which is illustrated by an abstract syntax tree.

Tokens are identified in the `tokens` file. Tokens that are not identified in this file will be reported as illegal characters, and the lexer will terminate execution. If you wish to add new tokens, add them to the `tokens` file and run `TokenSetup` via the instructions in **Running the program**

Execution and development environment

The program is written in Java and executed on JDK 1.8.0_121. It was developed in the Intellij IDEA IDE.

Scope of work

The assignment asked the student to add additional functionalities to an existing lexer program. These requirements are as follows:

1. Add a new token, ';' (following the procedure for adding tokens learned in Assignment 2).
2. Modify the Parser to accommodate the modified grammar specified below. This grammar includes new production rules to integrate the tokens added in Assignment 2, as well as the new semicolon token.

Production	Description
TYPE → 'float'	FLOAT token
TYPE → 'void'	VOID token
S → 'return' ';'	Return statement for void functions
NAME → '(' (E list ',')? ')'	Calling void functions
E → SE '>' SE	GREAT token

3. Remove all debug statements not required - this includes the listing of tokens from the Assignment 2 output. You will also need to follow the instructions in Compiler.java to obtain the desired output (see Sample Output section).
4. Correct the implementation of DrawVisitor.java to display a tree whose children are aligned as discussed in class.

To the best of my understanding, I have completed all the above requirements except for requirement 4.

Running the program

Run the following commands from your terminal. Replace `filename.x` with your .x file of choice.

```
git clone git@github.com:SFSU-CSC-413/assignment-3-nemimccarter.git
cd assignment-3-nemimccarter
javac lexer/setup/TokenSetup.java
java lexer.setup.TokenSetup
javac compiler/*.java lexer/*.java parser/*.java visitor/*.java ast/*.java
java compiler.Compiler filename.x
```

Assumptions

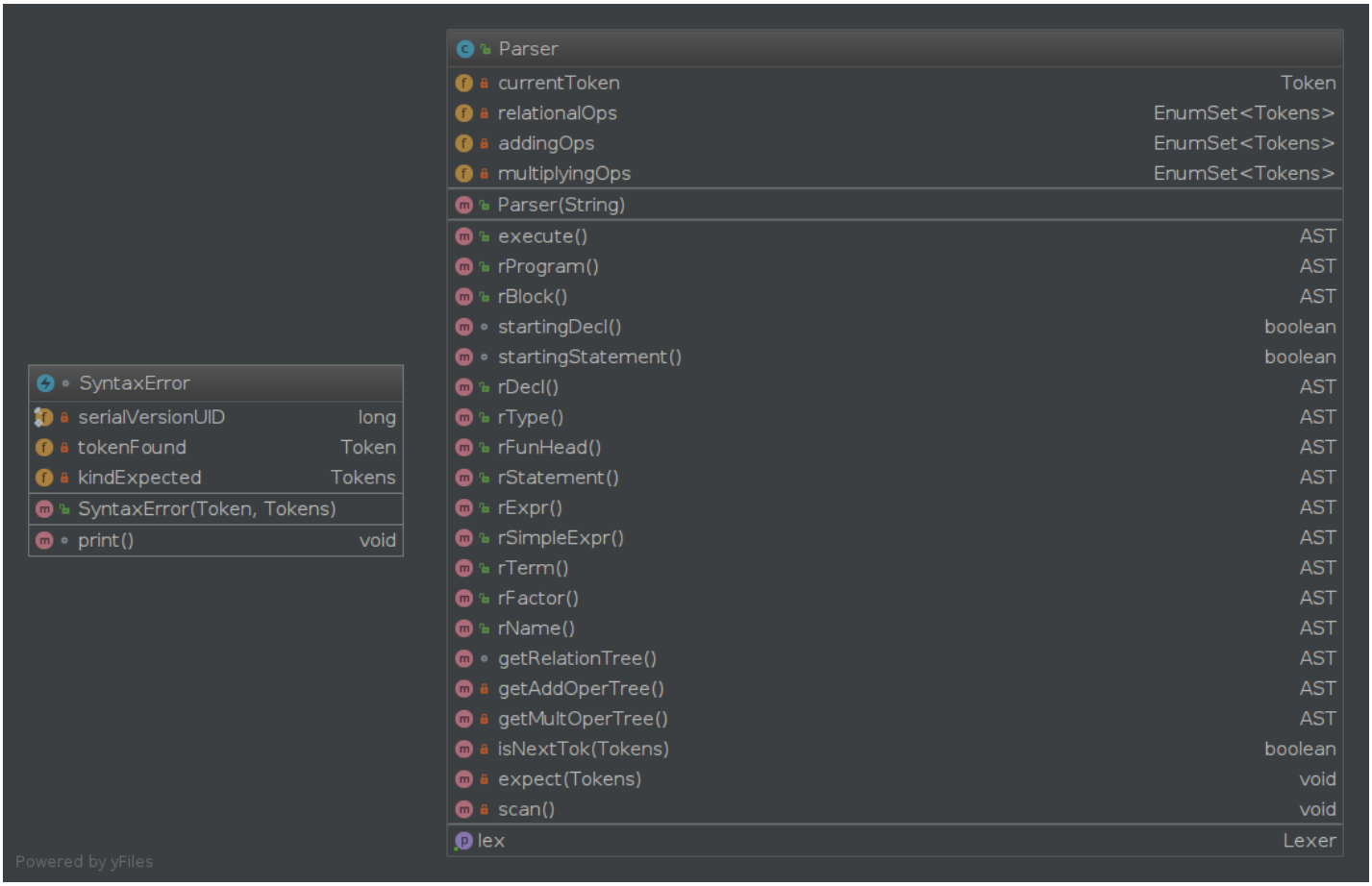
I assume the user has the current version of Java installed on their system. I assume the user can follow the above instructions in a command line interface. In completing the assignment, I assumed it would be graded in a similar fashion to assignment 2, so I referenced the assignment 2 rubric to improve upon areas I missed points for that assignment. The OffsetVisitor class assumes the tree structure is fixed – nodes will not be added by the time the AST reaches OffsetVisitor.

Implementation

Below we explore in detail the purpose and behavior of the relevant classes.

Parser

Purpose: build abstract syntax tree.



Parser UML

`Parser` verifies the syntactic structure of the `.x` program and builds an abstract syntax tree from this program.

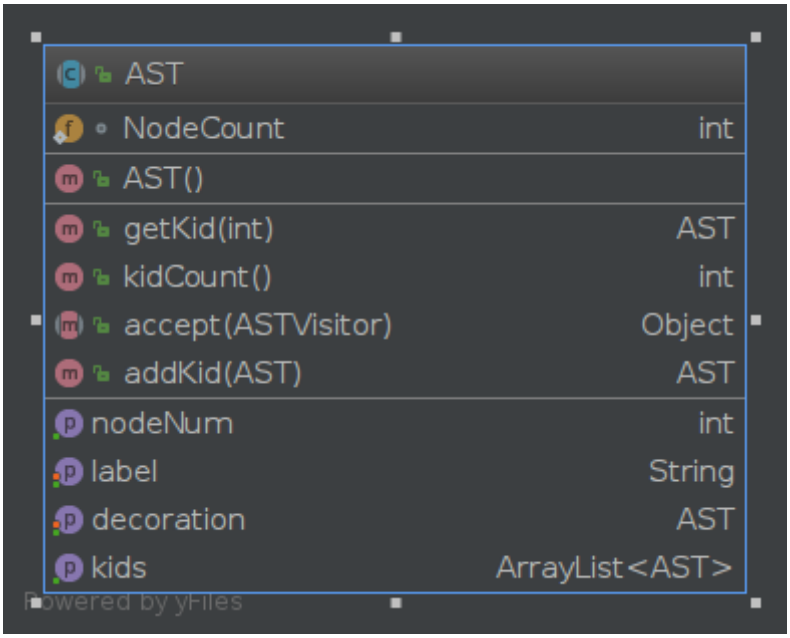
`Parser` receives these tokens from `Lexer`, builds the AST, and passes this AST to `Constrainer` to be decorated.

Each nonterminal in the grammar rules corresponds to a function in `Parser` (S is implemented by `rStatement()`, NAME is implemented by `rName()`, etc.).

`Parser` has three enumerated types, one for each kind of operator (addition, multiplication, and relational).

AST

Purpose: represent the abstract syntax tree and perform necessary operations to conceptually build the tree.

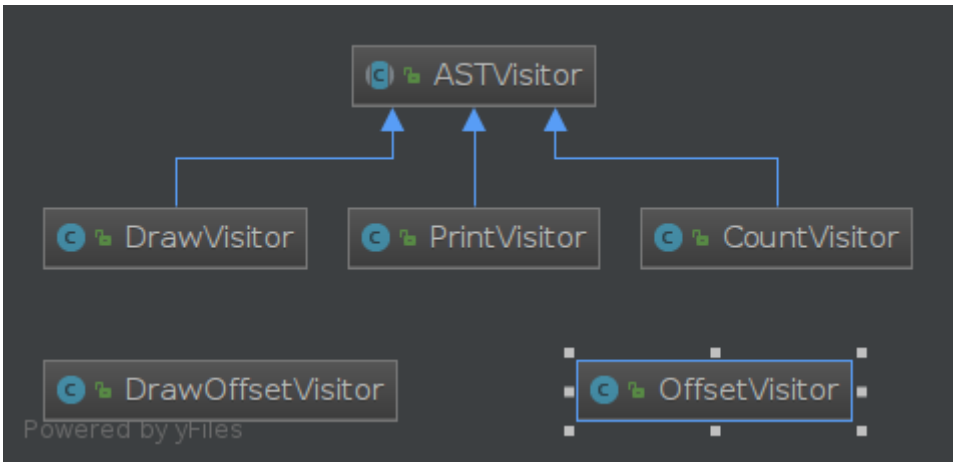


AST UML

`AST` is an abstract class that represents a generalized node of an AST. Each potential node of AST is a subclass of AST.

ASTVisitor

Purpose: abstract class to visit the nodes of AST



ASTVisitor UML

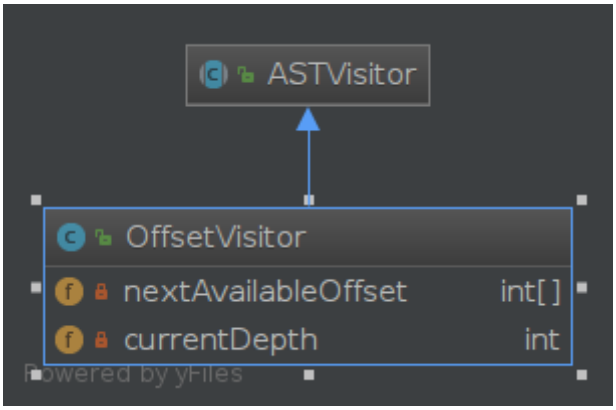
These classes are responsible for turning the conceptual work done in `AST` and turning it into a tangible diagram.

The implementation of abstract functions from `AST` are located here.

Every terminal and nonterminal (e.g., BLOCK, Statement, Int, Void, etc.) has a function in each of these classes. Members are hidden in the diagram for simplicity.

OffsetVisitor

Purpose: Calculate the horizontal left-margin offset for each node



OffsetVisitor UML

nextAvailableOffset is an array that uses currentDepth as an index and tracks what the next available offset is on each depth.

The algorithm to calculate the offset is detailed below:

a. Visit nodes in the AST in post order (this is effectively done already with the visit implementation), tracking the next available X offset for the current depth, starting at 0

b. Base case: if the node has no children, then record that node's X offset to be the next available offset for the current depth, update the offset, and return

c. After visiting all children, calculate the X offset for the parent - (rightmost child's X offset, plus leftmost child's X offset), quantity divided by 2.

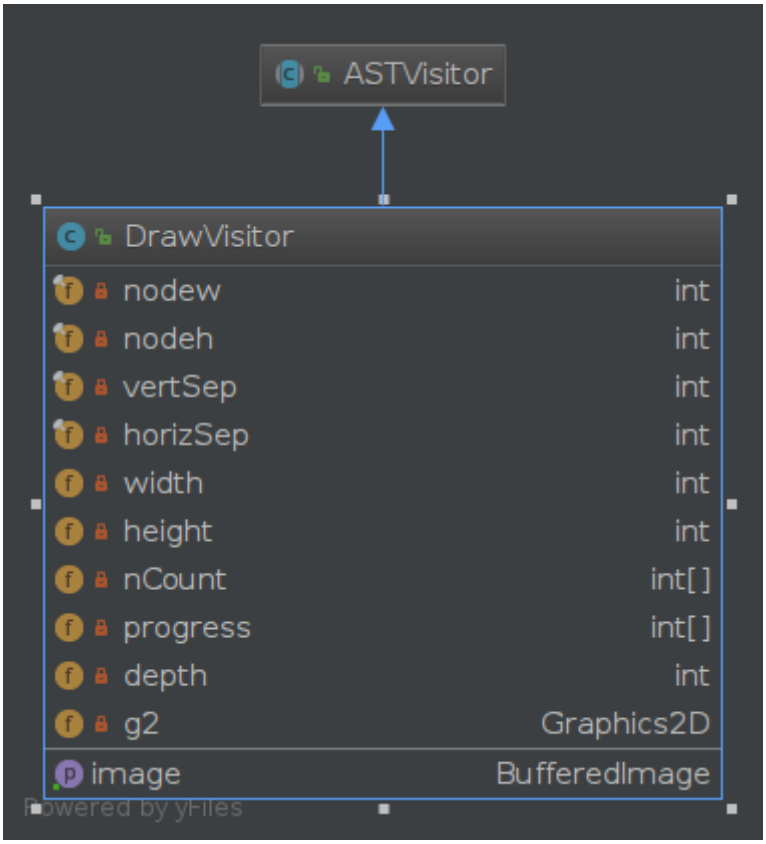
1. If the calculated offset is less than the next available offset for the current depth, we must recursively shift all children by (next available offset for the current depth - calculated X offset).

2. Set the current node's offset

3. Update the next available offset for the current depth

DrawVisitor

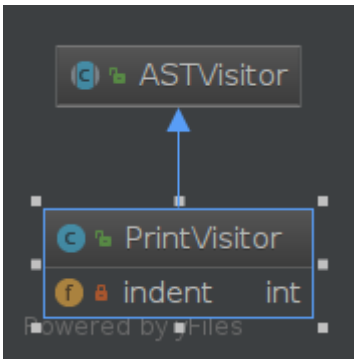
Purpose: create graphical representation of AST



DrawVisitor UML

PrintVisitor

Purpose: display graphical AST



PrintVisitor UML

Results and Conclusion

This assignment was another good exercise in working on the Java compilation process. Though the codebase has many components as the last assignment, the Parser, AST, and Visitor packages were unfamiliar. I used a more methodical approach to this assignment than the last ones. I started writing this readme before writing a line of code. I also stepped through the code's execution on an .x file using the Intellij IDEA debugger. Each of these was helpful in understanding what the code is doing, and thus I felt much less like I was throwing code at a problem. I intend to continue using this approach in the future.

I was not able to complete Requirement 4 of this assignment. I feel I received enough instruction in class to have completed it, but personal obligations limited the amount of time I had available to work on school assignments. I will accept the deductions I receive for an incomplete solution to requirement 4.