Brandon Watanabe
921988506
30 March, 2022

Assignment 3 Documentation

Github Repository 1
https://github.com/sfsu-csc-413-fall-2022-roberts/assignment-3---parser-BrandonWatanabeStudentAccount 1
Project Introduction and Overview 1
Summary of Technical Work 2
Scope of Work 2
Execution and Development Environment 3
Compilation Result 3
Assumptions 6
Implementation 6
Parser 6
Constrainer and Visitors 6
OffsetVisitor and DrawOffsetVisitor 7
Code Organization 7
Class Diagram 8
Results and Conclusion 8
Challenges 9
Future Work 9

Github Repository

https://github.com/sfsu-csc-413-fall-2022-roberts/assignment-3---parser-BrandonWatanabeStudentAccount

Project Introduction and Overview

For this project I had to implement a Parser which builds an Abstract Syntax Tree from tokens made by the previous assignment. I had to add functionality so the compiler can accept new tokens as well as a new switch statement.

Summary of Technical Work

The compiler reads the source file and tokens are then created from the characters in the file. These tokens are then passed through the Parser which builds an AST for all of the tokens. This is then taken by the visitor classes which check all of the variables and gives the labels for all the different types of tokens, counts the depth of the tree, creates the offset to space the nodes properly, then generates an image of the tree.

Scope of Work

Task		Completed
Implemented and updated in Parser.java		X
	boolean startingDecl()	X
	boolean startingStatement()	X
	public AST rType()	X
	public AST rSwitchBlock()	X
	public AST rCaseBlock()	X
	public AST rStatement()	X
	public AST rFactor()	X
	private EnumSet<Tokens> relationalOps	X
Implemented and updated in ast package		X
	CaseTree	X
	DefaultTree	X
	SwitchBlockTree	X
	SwitchTree	X
	TimestampTree	X

	TimestampTypeTree	X
	Utf16StringTree	X
	Utf16StringTypeTree	X
Updated Constrainer.java		X
Updated Compiler.java		X
Implemented and updated in visitor package		X
	ASTVisitor	X
	CountVisitor	X
	DrawVisitor	X
	OffsetVisitor	X
	DrawOffsetVisitor	X
Tested using the following .x file		X

	program { int i utf16string u timestamp t
	u = \u15bD\uA93b
	t = 2019~02~02~20:00:00
	switch (mouse) {
	case [dogs, cats] # return 0
	case [mouse] # return 1
	default # return 0
	}
	if (i < 2) then
	{ return 1 }
	}
		X

Execution and Development Environment

I used Java 17 on the IntelliJ IDE on my Windows computer to complete this assignment and tested in both the IDE and shell.

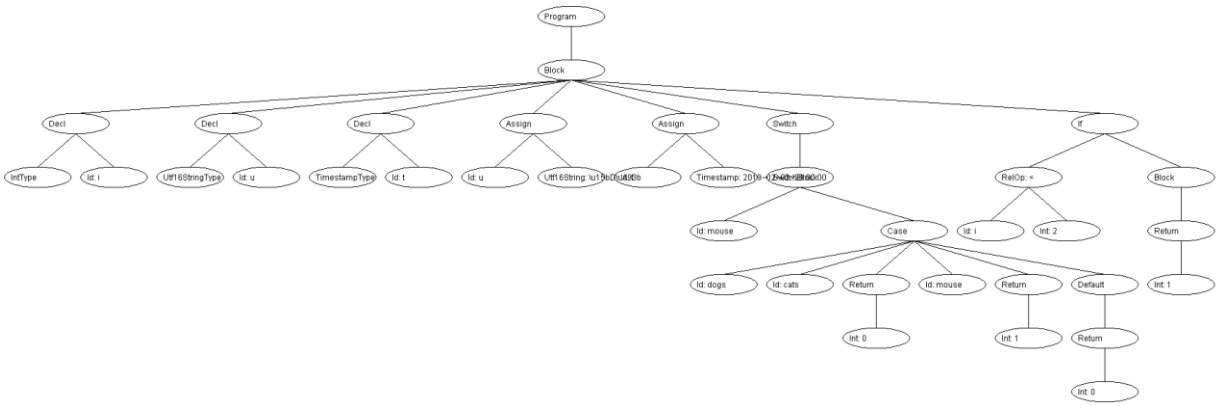
Compilation Result

Using the instructions provided in the assignment one specification:

```
program { int i utf16string u timestamp t
```

```
u = \u15bD\uA93b
t = 2019~02~02~20:00:00
switch (mouse) {
case [ dogs, cats ] # return 0
case [ mouse ] # return 1
default # return 0
}
if (i < 2) then
    { return 1 }
}
lexer.Lexer@37bba400
---------------AST-------------
1:  Program
2:    Block
5:      Decl
3:        IntType
4:        Id: i
8:      Decl
6:        Utf16StringType
7:        Id: u
11:     Decl
9:        TimestampType
10:       Id: t
13:     Assign
12:       Id: u
14:       Utf16String: \u15bD\uA93b
```

```
16:     Assign
15:       Id: t
17:       Timestamp: 2019~02~02~20:00:00
18:     Switch
19:       Switch Block
20:         Id: mouse
21:         Case
22:           Id: dogs
23:           Id: cats
24:           Return
25:             Int: 0
26:           Id: mouse
27:           Return
28:             Int: 1
29:           Default
30:             Return
31:               Int: 0
32:     If
34:       RelOp: <
33:         Id: i
35:         Int: 2
36:       Block
37:         Return
38:           Int: 1> java EvaluatorUI
```



No error messages or warnings were displayed, and the application ran as expected.

Assumptions

I assumed that all of the implementations of the other token types were correct because that is what I based my additions on. I assumed that all the other code that I did not touch.

Implementation

Parser

The Parser class was partially completed for me, but I needed to include the changes we made in the previous assignment. Before starting this assignment, I had to finish the previous one as this assignment builds on it. After that, I wrote out my plan on tackling this assignment. I broke it into multiple small sections to make it easier to understand. First, I walked through the existing code to see how all of it functions. Then, I implemented the utf16string and timestamp tokens into the methods to be able to be parsed. I did this by following the templates given to my by the int type as the logic is very similar. I then implemented the Greater and GreaterEqual tokens to be able to be parsed in a similar way to how the less than is parsed. Finally, I had to implement stack and case which was a lot more complicated. By looking at how rFunHead() interacted with declarations, I figured out how to solve this problem. It first expects “switch”, then handles the requirements for a SwitchBlock. In the SwitchBlock, it expects the requirements for that, which includes a CaseBlock. When it reaches the CaseBlock, it expects the requirements for that which includes a CaseList and a DefaultStatement. It checks for the existence and requirements for these as well.

Constrainer and Visitors

The implementation of this assignment in the Constrainer class was fairly straightforward, but it did have a lot of other classes related. First, I fixed the buildIntrinsicTrees method which was giving me an error as the arguments for calls it was making needed to be updated with line number. I then added in all of the visit trees. I added the respective trees to the ast package. I then added in the visit tree methods into the ASTVisitor, CountVisitor, and DrawVisitor classes. These implementations followed a similar format to the other methods since they are all

structured the same way. After that I deleted all of the debug information as it was not needed anymore.

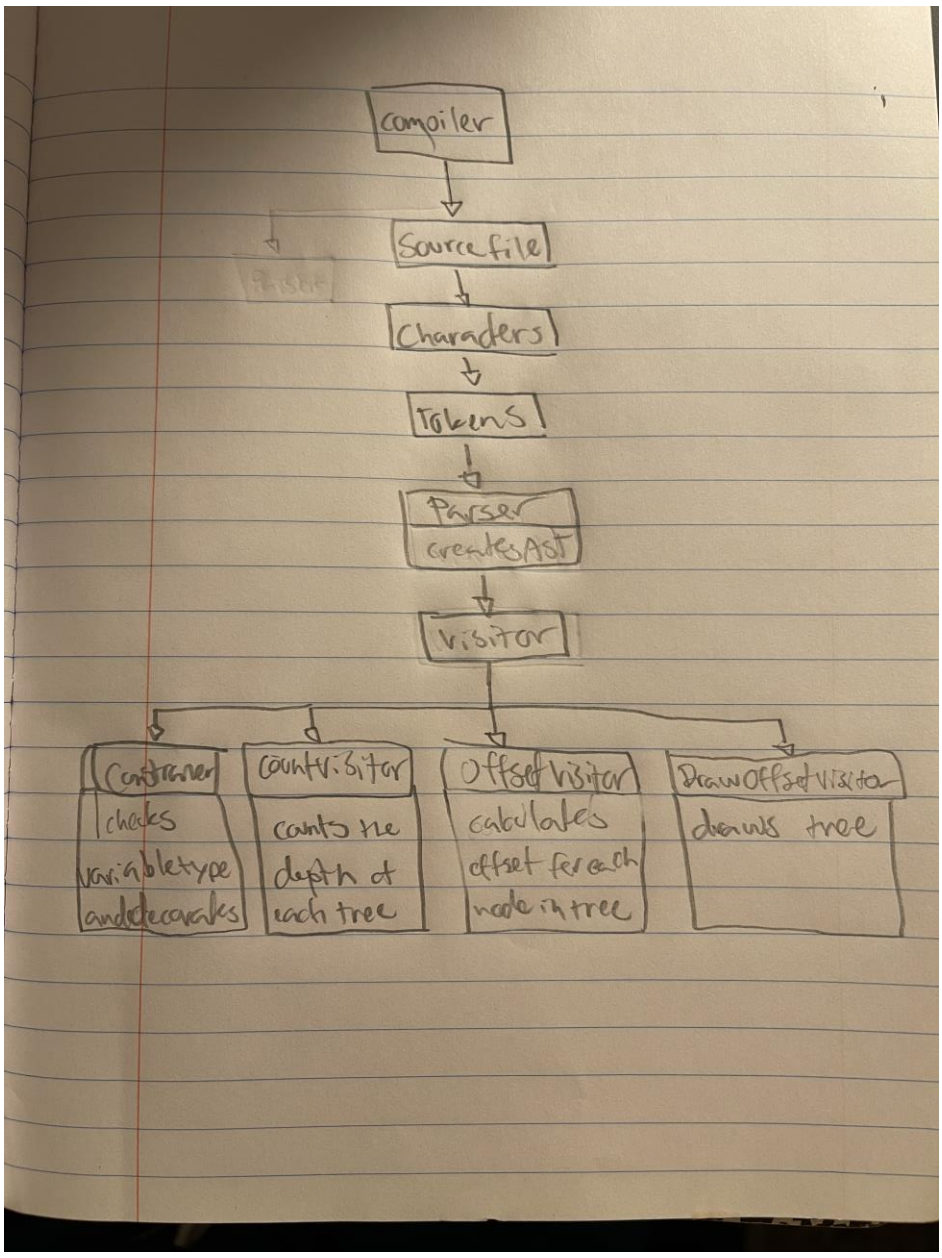
OffsetVisitor and DrawOffsetVisitor

This was the most difficult part of the assignment conceptually for me. Up until this point, I felt like I understood what to do. However, I did end up figuring out what to do and it was not too complicated. These classes are based on the CountVisitor and DrawVisitor classes, with some significant changes. I had to implement the algorithm discussed in class to calculate the offset of each node in the AST for the OffsetVisitor class. Then I had to use the offsets to correctly display the nodes using math to figure out the placement in relation to its parent. After this, I added them into the Compiler class so that it would use the new offset classes instead.

Code Organization

I implemented all of the new classes and methods in the same style as the ones provided so that it was easy to read and understand what additions were made.

Class Diagram



Results and Conclusion

This project challenged my understanding of how object oriented programming works. I got to see how different classes and methods can be used. I also finally understand the purpose of an abstract class. The ASTVisitor abstract class really helped me stay organized and keep track of all of the methods I needed to implement into the various visitor classes. It would be very easy to miss something without the abstract class keeping the structure.

Challenges

The most challenging portion of this project for me was the final step of creating the offset classes. This was a challenge for me because I did not really understand how to do this part and was stuck for a while until it clicked. By looking at how the other visitor classes were made and used, I figured out how to write the new ones.

Future Work

A way to build on this would be to add functionality for more tokens. A step beyond that would be checking that a variable is declared and assigned a value before using it in something like an if statement as it does not check for that at the moment.