SENG 300 Group Assignment #1

Anastasiya Lazarenko (10175402),
Zachary Hull (10109756)
Matthew Buhler (00332036)

March 14, 2018

The code provided is expected to be able to take in user Input of a Java Type and Pathname of where files are located. If it does not receive this input as arguments, it explicitly asks for user input. If input is not received the second time, it throws an exception. Provided valid inputs, the program returns the number of references and declarations of the java type passed in.

We have gone through a few iterations over the week as we understood the assignment requirements further. However, given the assignment details and the significant changes made, it is highly likely there are parts of this assignment that do not function to expectations. We emphasized updating the code to remove redundant classes and functions to help with the second group project.

Unit tests are in the test/tests directory. Since we were unable to get Enum and Annotation declarations to work as they were expected to, some of our tests fail and our coverage of the VisitDeclarations class is low. The VisitDeclarations class includes a way to visit Enum and Annotation declarations, but these methods are never called even when the appropriate structures are parsed into the AST (as shown in tests) for unknown reasons.

There are four main classes:

- Main – User input class. Validates and manipulates user input to pass over to MyParser Class and then calls for final outputs from MyParser.
- MyParser – Loads files from directory, parses each file into text, builds an Abstract Syntax Tree (AST), calls the VisitReference and VisitDeclartion classes, maintains the count of references and declarations.
- VisitReferences – Looks for all references in the provided ASTNode and compares the name of every found node to the type name provided by the user. If matching, increments a counter.
- VisitDeclarations – Looks for all declarations in the provided ASTNode and compares name of every found node to the value provided by the user. If matching, increments a counter.

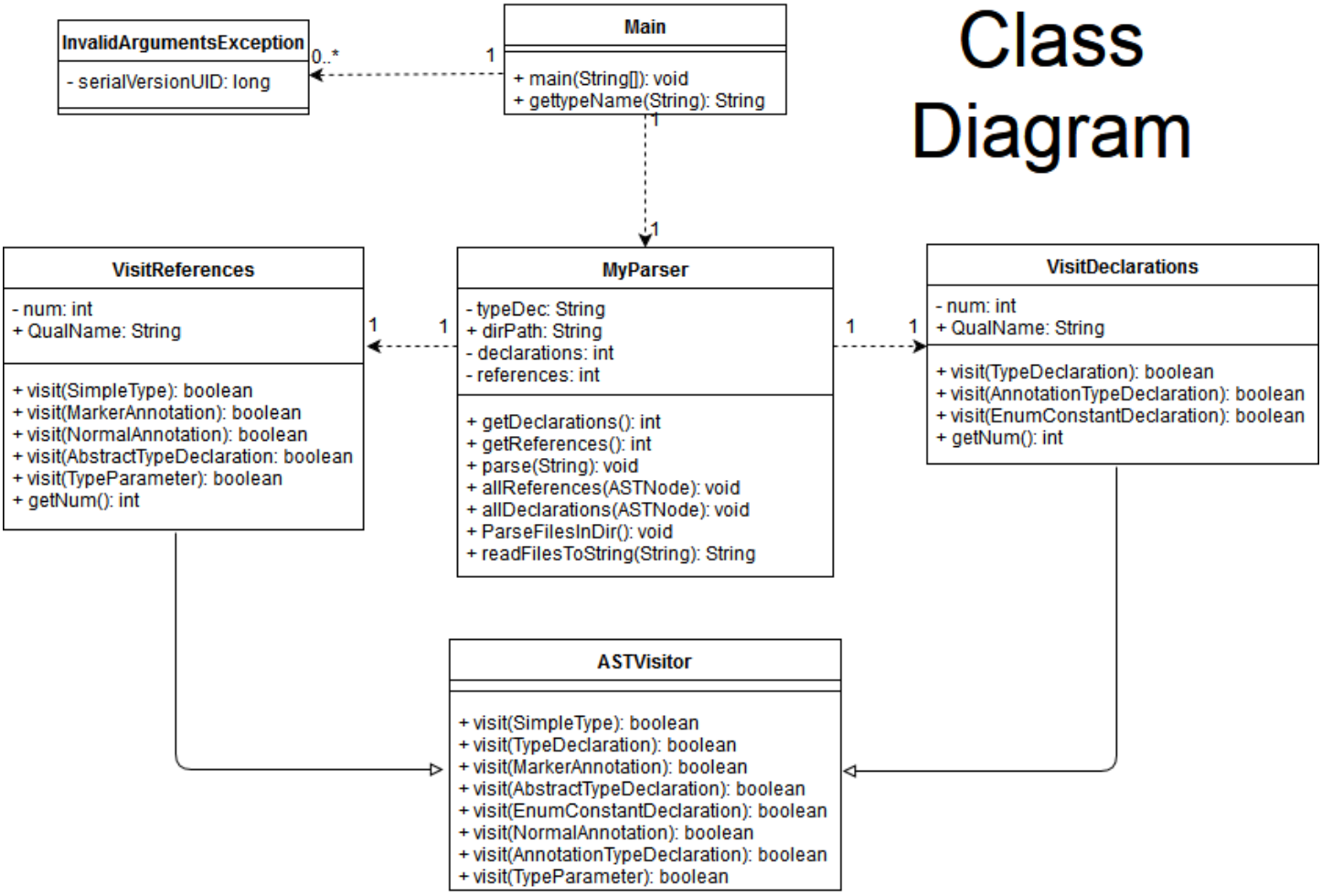
Diagrams:

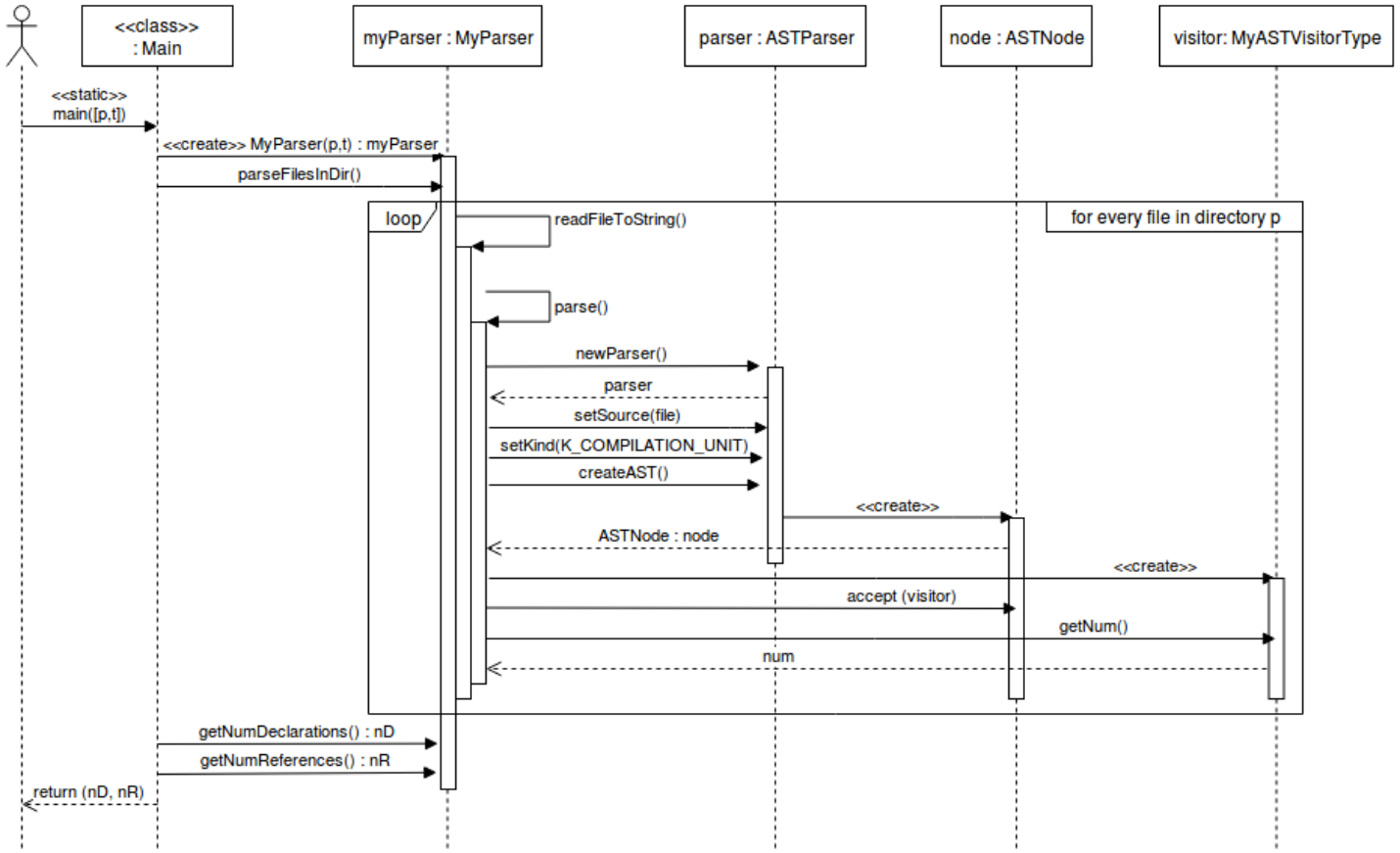
- State Diagram: Is designed to show the different states a user can interact with in the program. They can supply the program with 2 or more arguments for the program to run as intended or they can supply the program with less than 2 arguments and exceptions will be thrown.
- Class Diagram: Shows our four main classes as well as an exception class and a parent class and its methods used.
- Sequence Diagram: Shows the main flow of execution of the program. Details like functions that do little have been abstracted out. The classes VisitReference and VisitDeclaration are represented as MyASTVisitorType. Interactions with the user are assumed to be through a console.

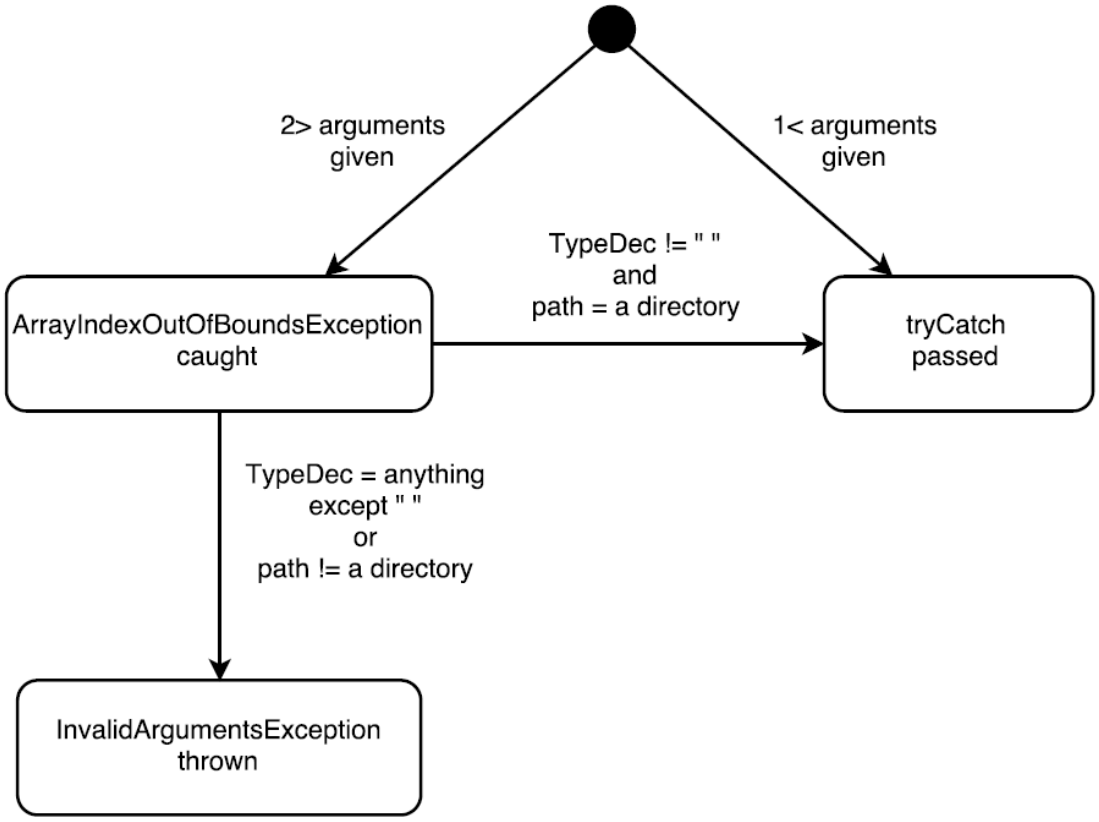
Sources referenced:

1. https://www.programcreek.com/2011/11/use-jdt-astparser-to-parse-java-file
2. org.eclipse.jdt.core.dom API

Class Diagram




<<class>>
: Main
myParser : MyParser
parser : ASTParser
node : ASTNode
visitor: MyASTVisitorType
<<static>>
main([p,t])
<<create>> MyParser(p,t) : myParser
parseFilesInDir()
loop
for every file in directory p
readFileToString()
parse()
newParser()
parser
setSource(file)
setKind(K_COMPILATION_UNIT)
createAST()
<<create>>
ASTNode : node
<<create>>
accept (visitor)
getNum()
num
getNumDeclarations() : nD
getNumReferences() : nR
return (nD, nR)


2> arguments
given
1< arguments
given
TypeDec != " "
and
path = a directory
ArrayIndexOutOfBoundsException
caught
tryCatch
passed
TypeDec = anything
except " "
or
path != a directory
InvalidArgumentsException
thrown