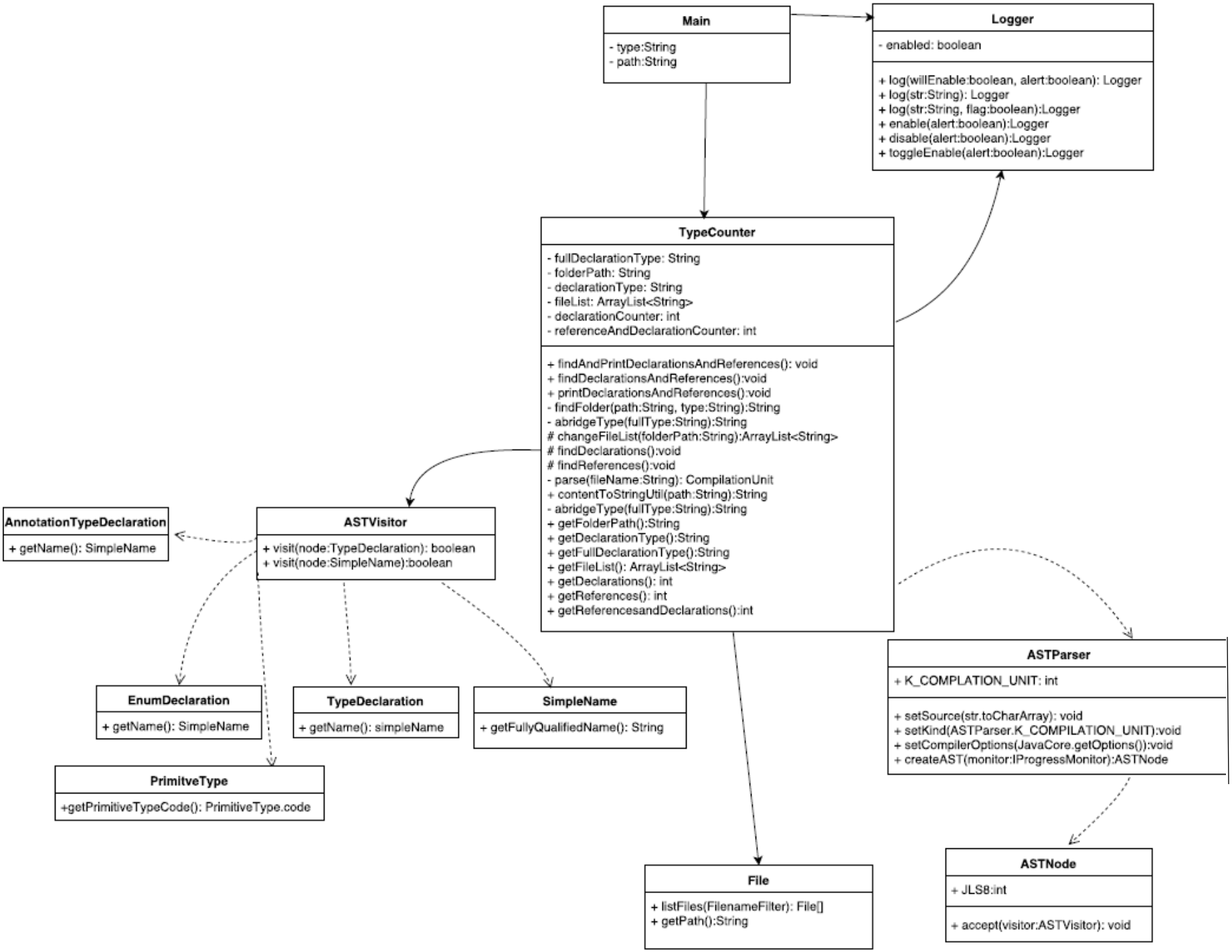
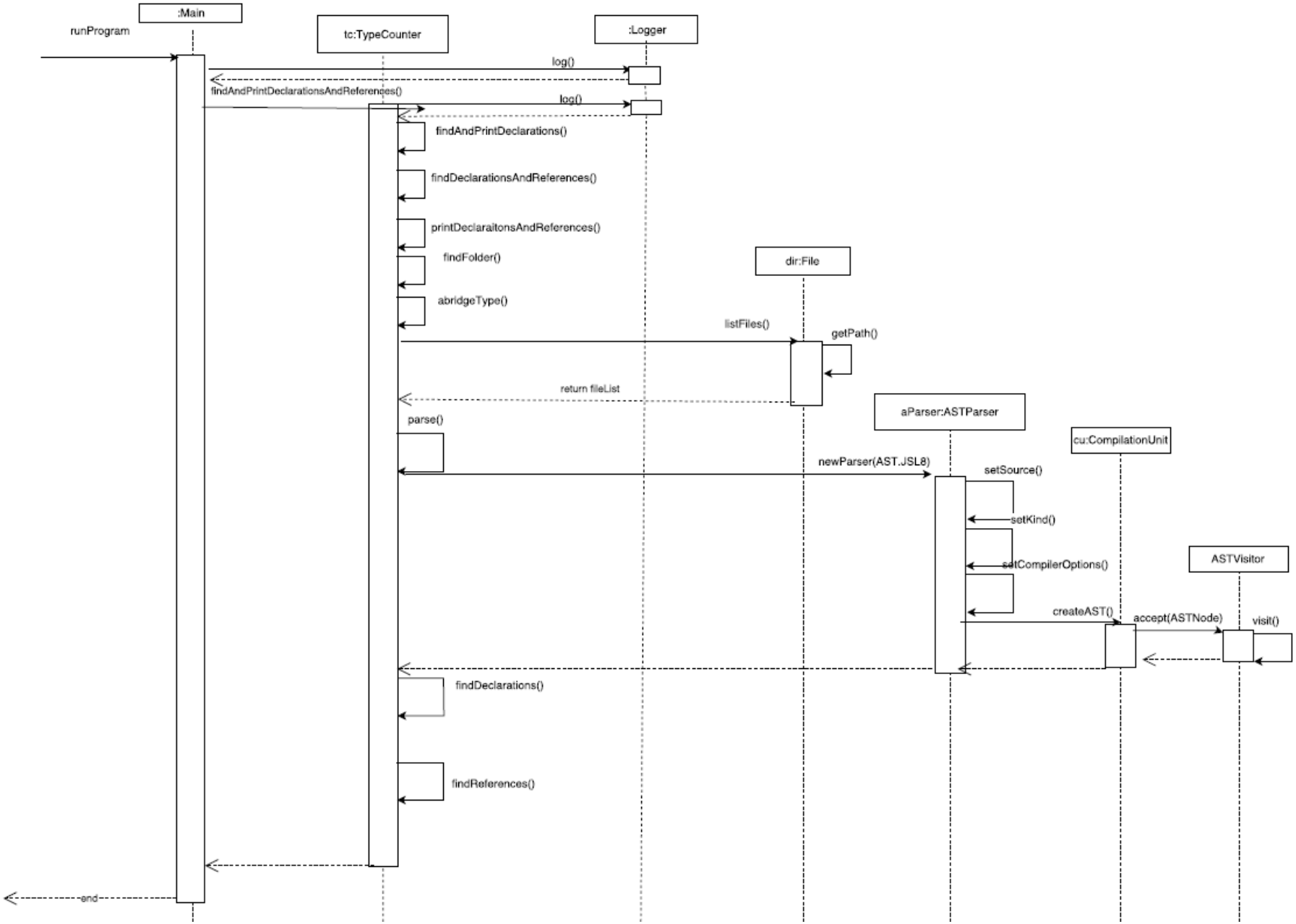
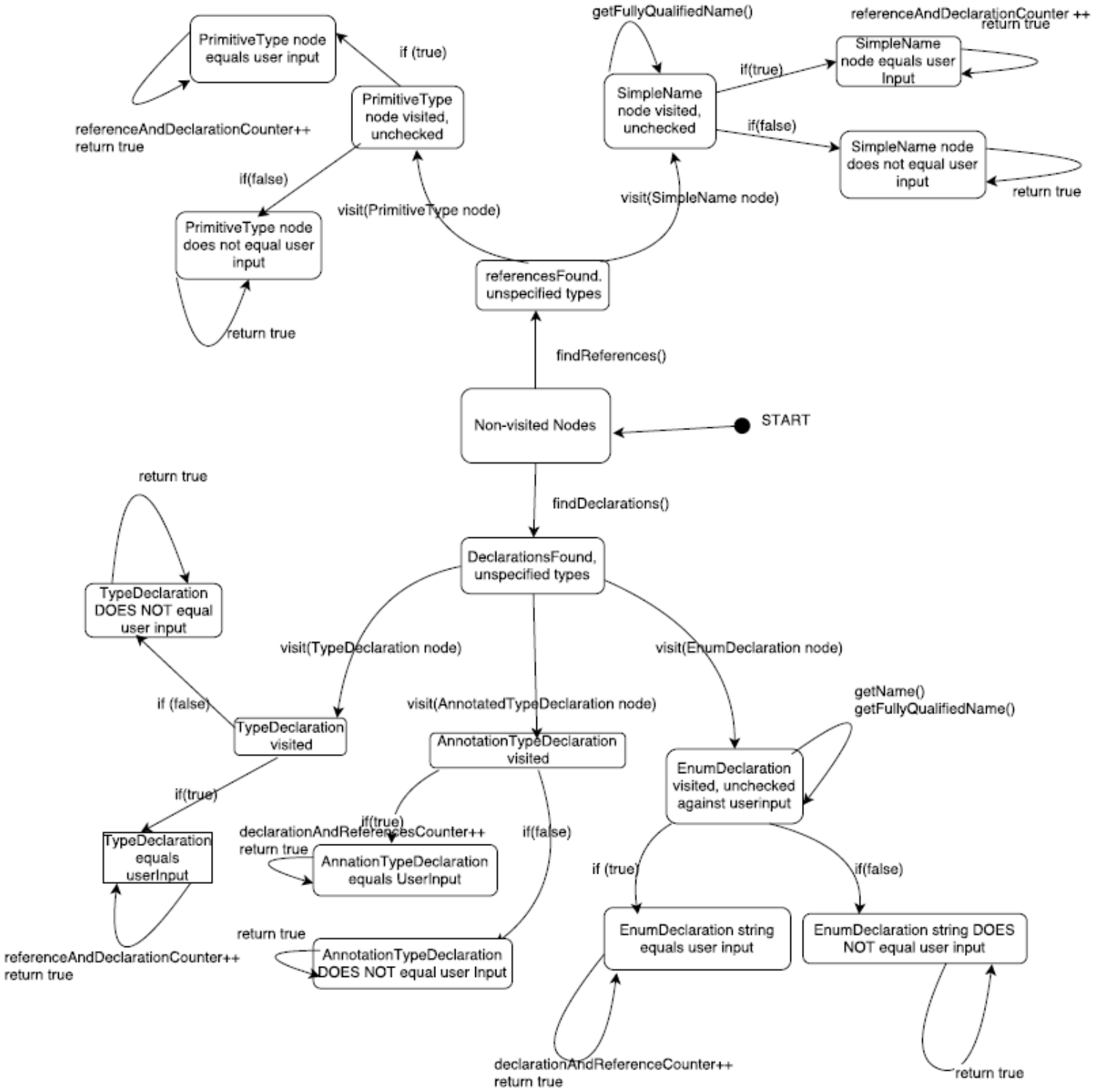
Class Structural Diagram for GroupAssign Iteration One



Sequence Diagram for the GroupAssign Iteration One



State Diagram for the GroupAssign Iteration One



Supplementary Document

Class structural diagram

The package contains three important classes: Main, TypeCounter, and Logger which are modelled in the class structural diagram. Other important classes that are used in the program are ASTVisitor, ASTParser, ASTNode, SimpleName, AnnotationTypeDeclaration, EnumDeclaration, TypeDeclaration, PrimitiveType, and File. The program starts at main, where the fields are type and path of the type String, this is received as input from users. An object of TypeCounter is created and the method findAndPrintDeclarationsAndReferences is called, since the object of TypeCounter was created in Main, this has an associative relationship with the class TypeCounter. A logger was also created in Main, thus also has an associative relationship exists here. The important methods are shown in Logger. The program runs primarily through the TypeCounter which essentially counts all the declarations and references of the type specified by the user within the pathname that was provided as an input from the user as well. ASTParser was used to parse the compilationunits of interest to generate an AST, the nodes are visited using ASTVisitor, thus these were included in this diagram, where only the relevant methods/fields that were used for our purpose was included, for example, the method visit() from ASTVisitor was used so therefore it was included in this diagram. Different types are considered and checked against the userinput, the types checked were SimpleNames, PrimitiveTypes, AnnotationTypeDeclaration, EnumDeclaration, and class declarations, each of these are classes of their own and were parameters in sub methods within the findReferences() or findDeclarations() method in TypeCounter. File I/O was involved to read/parse the files from the pathname provided by the user, therefore File was included in this diagram.

Sequence Diagram

The program begins through Main as soon as the user runs the program. When an object of TypeCounter is created and the method findAndPrintDeclarationsAndReferences() is called, the class TypeCounter is activated, A logger is created and therefore activates he Logger class. Relevant methods within TypeCounter such as findAndPrintDeclarations(), findDeclarationsAndReferences(), printDelcarationsAndReferences(), findFolder(), abridgeType(), parse(), findDeclarations(), and findReferences() are included within the activation. The call to the method listFIles() activates the class File, the call was through the object dir, from here the method getPath() is called and a fileList is returned. The call to newParser() activates ASTParser. Within the parser(), setSource(), setKind(), and setCompilerOptions() is called, an AST is returned through the a call to createAST(), which activates the compilationUnit. Accept methods activate ASTVisitor which allows visitation of nodes in the AST.

State Diagram

The state of the program starts with Unvisited Nodes, after the call to findDeclaration() or findReferences(), they then check for the unspecified types, the various nodes are visited and then checked against the user's input, if it is true (i.e in the diagram if(true)), then the declarationAndReferenceCounter is incremented and true is returned, if false then only true is returned.

Sources:

1. Diagrams were generated using http://draw.io/
2. Information was obtained from https://help.eclipse.org/mars/index.jsp?topic=%2Forg.eclipse.jdt.doc.isv%2Freference%2Fapi%2Forg%2Feclipse%2Fjdt%2Fcore%2Fdom%2Fpackage-summary.html