CS335 Milestone 2 - Creating AST for java program

Tejas Ramakrishnan-201050, Ujwal Jyot Panda-201060, Uttam Kumar-201071

January 2023

To implement support for the symbol table data structure, perform semantic analysis to do limited error checking, and then convert the input source program into an Intermediate Representation (IR)
We have used flex and bison for lexical scanning and parsing respectively. We have also used C++ to implement typechecking and source program conversion.
Compilation instructions- To compile, first navigate to the directory 'src' in the terminal. The files in this directory are: scanner.l, parser.y, tree.h, tree.cpp, typecheck.h, typecheck.cpp, 3AC.h, 3AC.cpp, symbol_table.h and symbol_table.cpp. To compile the program, run the following command in the terminal:

```
make
```

You may get an warning stating 'clang: warning: treating 'c' input as 'c++' when in C++ mode, this behavior is deprecated', please ignore that.
Execution instructions- An executable 'ans' would have been created in the directory 'milestone1'. To execute the program with a test case file 'test1.java' in the "tests" directory, run the following commands from the src directory:

```
./ans -input ../tests/test1.java -output graph.dot
```

The output will be created in a txt file called 'final_3AC.txt' in the same directory, in the required format. Please open to check the output. Options supported in the parser are:

```
--help: For usage guidelines
-input <file>: Passes the input java file to the parser to be read
-output <file>: Creates the dot script containing the AST in <file>
-verbose: Prints entire line of error
```

Assumptions:

- main function is used at the end of the code, and the functions/classes used are declared above.