CS335 Project Milestone2

Pratham Jain (200712) , Gopal Aggarwal(200390) , Het Hitendrakumar Patel (200440)

March 2023

1 Tools and Utilities

- We have used flex for scanning the input file and it returns tokens.
- We have used bison for implementing the parser.We have generated grammar having no conflicts.
- Make utility is used to ease the process of writing multiple commands into one big command (i.e make).
- So, finally we have generated scanner and parser which takes as input an JAVA program and performs 2 functions. One of the parser creates Symbol table along with type checking and other parser generates 3ac codes.

2 Feature Implemented

- We have created type checking for various constructs and have created symbol table according to their scopes.
- Symbol table consists of Name, Type, Line number, Function Input Type, Function Output Type, Size and Scope.
- For representation of arrays, we have considered the type of element followed by number of brackets. For example, a 3D array of 'int', we have int[][][] as our type.
- In case of class, the type name stored in symbol table is 'class' itself.
- In case of method, the type name stored in symbol table is 'method' itself.
- In case of constructor, the type name stored in symbol table is 'constructor' itself.
- In case of for loop, the type name stored in symbol table is 'for' itself. Similarly for other such cases we have done the same thing.

- We have used a feature that expands the data-type in order to preserve the information for example, a = 1 + 2.3, so here a is considered to be a 'float'.
- Apart from these features we have created another parses which produces 3ac codes for corresponding Java programs.
- We have used vector of strings to store the 3ac codes in which each element of vector indicates one line or one statement of 3ac. In the end these codes are dumped into a text file where you can see the required output.

In this way , we have implemented all basic features as mandated. We have submitted 10 non trivial test programs which in all covers all constructs/aspects of JAVA language and compiles, parses and creates both the symbol table as well as generates the 3ac codes.

3 Command Line Options

- $--help$ or $--h$: This command helps the user running the program in knowing about various other commands required to run program or use any additional features.
  Examples of use

```
./parser1 -h
./parser1 -help
```

- $--input$: This command is used for giving input to the program, i.e the path of the testfile (which is our input) is to written after a space after this command There should be a space between this command and inputfilename. Path can not be empty. Note that input file is of format .java
  Giving input filename(i.e. this command) is mandatory
  Examples of use
  One of them -

```
./parser1 --input ..//tests/test_1.java
./parser1 --input ..//tests/test_5.java --verbose
```

  Here parser1 takes input from Java program and converts it into Symbol table stored in a csv file.
  Similarly other parser takes input from Java program and converts it into 3ac codes which are stored (dumped) into a text file.

- $--verbose$: The output of verbose is seen in the file $parser1.out$ or $parser2.out$. It shows if grammar has any conflicts and also has the actions and goto definitions for all states, it also has information about all the states. **The submitted parser.y has neither shift/shift nor any shift/reduce conflict**.
  Giving this command is optional
  You can give input, output , verbose command in any order

4 Instructions for Running Test Cases

- First go to directory $./milestone1/src/$
- Execute following commands :

```
> make clean
> make Java
```

  in succession

- Now write the command :

```
./parser1  --input <inputfilepath> --verbose
./parser2  --input <inputfilepath> --verbose
```

  where $< inputfilepath >$ is file path of input file.

  Note that input file is of format .java and output file can be of 2 types since we are having 2 different parsers giving 2 different kinds of output, one is giving a .csv file (which is the symbol table) and other is giving a .txt file (which is the list of 3ac codes for the given program).

  Also giving input filename is mandatory, by default the output file will be "symboltable.csv" file is generated in case of parser1 and "3ac.txt" in case of parser2 file since we have 2 different parsers for 2 purposes.

  Note that since we are using 2 parsers thus we require 2 scanner files that is the same file duplicated for using it in another parser.