CS335 Project Milestone1

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 produces Abstract Syntax Tree (AST) of it

2 Feature Implemented

- We have implemented all basic features given in Milestone 1
- Extra Primitive data types (byte,short) used. **(test_1.java)**
- Java-style declarations (e.g., int[][] a = ...) **(test_2.java)**
- Flow via do-while and switch **(test_3.java)**
- Class definitions support protected access modifiers. **(test_4.java)**
- Support for import ... statements. **(test_5.java)**
- Support for Strings, including a few operations like concatenation, support for printing with println() **(test_6.java)**
- Support for Interfaces **(test_7.java)**
- Static polymorphism via method overloading **(test_8.java)**
- Dynamic polymorphism via method overloading **(test_9.java)**
- import statements like import java.util.* **(test_10.java)**
- Try Catch Exception Handling **(test_11.java)**

- Class and Interface inheritance **(test_12.java)**
- Type casting **(test_13.java)**

In this way , we have implemented all basic features as mandated as well as almost all advanced features that were made optional. We have submitted 25 non trivial test programs which in all covers all constructs/aspects of JAVA language and compiles and parses them all perfectly

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

```
./parser -h
./parser -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

```
./parser --input ..//tests/test_1.java
./parser --input ..//tests/test_5.java --output final.dot
./parser --input ..//tests/test_5.java --output final.dot --verbose
```

- $--output$: It is similar to output, here we will mention the path where the dotfile is expected. Path can not be empty.
  Giving outfilename(i.e. this command) is optional. In case you don't give output filename , by default "output.dot" file is generated which contains output (DOT notation) of input file you have given
- $--verbose$: The output of verbose is seen in the file *parser.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 :

```
./parser  --input <inputfilepath> --output <outputfilepath> --verbose
```

  where $< inputfilepath >$ and $< outputfilepath >$ are file paths of input and output files.

  Note that input file is of format .java and output file is of format .dot. This command provides input and output to the program.
  Also giving input filename is mandatory , but giving outfilename and verbose command is optional . In case you don't give output filename , by default "output.dot" file is generated which contains output (DOT notation) of input file you have given

- Then Use this command :

```
dot -Tps <outputfilepath> -o graph.ps
```

  to create the postscript file(.ps) of AST This will create graph.ps file which is postscript file of <inputfile> JAVA program

- Moreover,on Linux based distribution, you can use the command

```
xdg-open graph.ps
```

  to visually see the AST