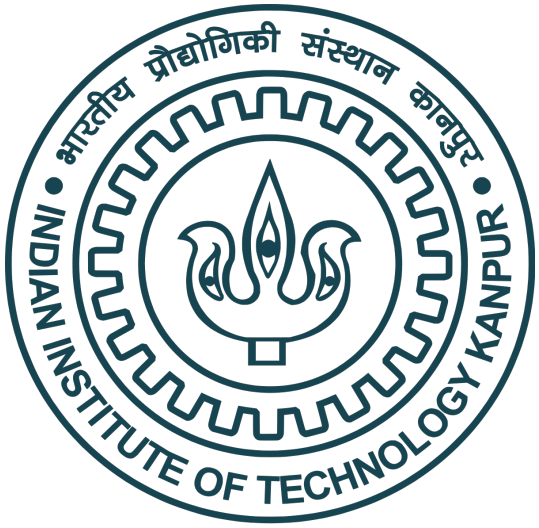

CS 335A

Project : Milestone 1

Samarth Arora (200849)
Sarthak Kohli (200886)
Abhimanyu Sethia (190023)

Instructor: **Swarnendu Biswas**

Date: March 1, 2023

1 Folder Content:

The submitted folder consists of the milestone1 directory which includes the following sub-directories and files:

- **milestone1/src** sub-directory that contains all the source files
- **milestone1/tests** sub-directory that contains all the test files
- **milestone1/dot_outputs** sub-directory which contains the '.dot' output files for all the test files.
- **milestone1/ast_outputs** which contains the '.png' images which represent the Abstract Syntax Tree of all the test files.
- **milestone1/helpers** which contains the code files used during the project.
- **milestone1/readme.pdf** which contains this PDF.

2 Requirements and Tools Used:

We have the following tools to build an Abstract Syntax tree given a java file input:

- **Flex**: For Lexical Analysis
- **Bison**: For Syntactical Analysis
- **Graphviz and DOT**: To visualize the Abstract Syntax Tree
- To install the above requirements use the following commands:
  - sudo apt-get update
  - sudo apt-get install flex
  - sudo apt-get install bison
  - sudo apt-get install graphviz

3 Compilation Instructions:

Follow the given steps to compile and execute the files:

- **Step1:** Open the terminal in the *milestone1* folder and go to the *src* directory. The following Command will get you there:

```
cd src
```

- **Step2:** To compile the files enter the following command:

```
make compile
```

- **Note** The compiled files will create the following extra files in the *src* folder. the **output** file is the executable. The files are:

```
lex.yy.c
parser.output
parser.tab.h
parser.tab.c
output
```

To remove these files to clean the *src* repository enter the following:

```
make clean
```

4 Execution Instructions:

4.1 Individual file from the Command Line

- To execute a file directly from the command line, enter the following command where ../tests/test_1.java is the destination of the input file from the src folder and ../dot_outputs/test_1.dot is the destination of the output file from the src folder.

```
./output --input=../tests/test_1.java --output=../dot_ouputs/test_1.dot
```

- One can enter just the input file also. This will create a corresponding ".dot" file in the location of the input folder. For example, the following command will create a test_1.dot file in ../tests/

```
./output --input=../tests/test_1.java
```

- Use the verbose option to list all the shift, reduce and lexer actions. For example:

```
./output --input=../tests/test_1.java --verbose
```

- Enter the following command to view about all the available options:

```
./output --help
```

- If a file which does not have the extension ".java" is entered or a invalid option is entered then an appropriate error will be thrown.
- To visualize the ".dot" file enter either of the following commands to generate a "postscript" or a "png" file respectively.

```
dot -Tps my_dot_file.dot -o my_ast.ps
dot -Tpng my_dot_file.dot -o my_ast.png
```

4.2 Run the Tests Folder

- To run the Tests Folder, put all your java files in it.
- To generate ".dot" files for the whole folder ensure that the **dot_outputs** folder exists and then enter the command:

```
make dot
```

- To generate the ".png" files to visualize the ".dot" files, ensure that the **ast_outputs** folder exists and enter the following command:

```
make ast
```

5 Bonus features Implemented:

We have implemented the following Bonus features:

- Support for import statements like import java.util.*
- Support for Strings including operations like concatenation and printing with println()
- Support for interfaces
- Support for TypeCasting
- Support for Packages