MileStone-1

Tools:

We have used flex for lexer, bison for parser (c++ in code section), and Dot language.

Compiling and Execution Instructions:

First, make sure to see if your computer has flex,g++,bison(latest version=bison (GNU Bison) 3.8.2),dot installed.
In the src folder, we have the following files:

- java.l
- java.ypp
- Makefile

Makefile consists of the following commands:

- flex -o java.l
- bison -d java.ypp
- g++ lex.yy.c java.tab.cpp -o myASTGenerator

Open the terminal from the src folder, enter the following commands:

- make
- ./myASTGenerator ../tests/test5.java –output=test5.dot

The output tree is created in test5.dot file.
We can generate ps file, which can be obtained by following command:

- dot -Tps test5.dot -o test5.ps

Features:

We have included all the basic features described in the document,like multidimensional arrays,recursions, classes and objects,etc.
We have also included Enhanced For loop along with if,else and while loops.
We have also added some of the optional features.

- Support for Strings and Interfaces.
- Support for Import statements (like import java.util.*)
- Support for byte and short data types.

There are **no Shift-Reduce or Reduce-Reduce** conflicts in our grammar.
We have added 10 testcases in testcase directory, and the output produces trees for all the outputs.

Limitations:

The output is Parse tree and not an AST, so we get some extra states/non-terminals in the tree, which results in enlargement of the tree.
We can adjust that by adding size commands in dot language command(i.e dot -Tps graph.dot -o graph.ps).