CSCI3120 Programming Assignment #2 Due: 7 Mar 2018

**Objectives**: (a) To use JavaCC to produce a parser, (b) to add semantic action in the parser to build an abstract syntax tree (AST), and (c) to write code that uses “visitor pattern” to process the AST.

**MiniJava Variant**

In this assignment, we will make the following changes to the MiniJava described by the grammar in “grammar.htm”.

1. Additional operators:
   1. **||**  Logical OR
   2. **/** Integer division
   3. **-** Unary minus
   4. **^^** Exponential operator (x ^^ y means x to the power y)

The operator is right associative and its precedence is between level 12 and level 13 in the precedence table at <https://introcs.cs.princeton.edu/java/11precedence/>. That is, its precedence is higher than the multiplication operator but lower than the new operator.

Note: This operator does not exist in Java. MiniJava is a subset of MiniJava variant but MiniJava Variant is not a subset of Java.

1. Inclusion of FOR loop as described by the following grammar:

|  |  |  |
| --- | --- | --- |
| Statement | ::=  | | "for" "(" ForInit ";" Expression ";" StmtExprList ")" Statement  … |
| ForInit | ::=  | | LocalVariableDeclaration  StatementExpressionList |
| StmtExprList | ::= | StmtExprList ( "," StmtExprList )\* |
| LocalVariableDeclaration | ::= | Type Identifier "=" Expression ( "," Identifier "=" Expression )\* |

Note: A FOR loop is a kind of statement.

In addition to the above changes, we will reuse the definition of identifier and integer literal in programming assignment #1. That is, characters in the source program are limited to those with ASCII code in the range 0-127, and integer literals can be in octal, decimal, and hexadecimal formats.

**Todo**

**Task 0: Warm Up**

The parser produced by minijava.jj in “chap4.zip” can build an abstract syntax tree (AST) for any MiniJava program (described by the MiniJava grammar specified in “grammar.htm’) when there is no lexing and parsing errors.

Your first task is to build the parser successfully from the JavaCC and Java source files in “chap4.zip”.

**Task 1** (Please name your JavaCC file as **Asgn2.jj**)

Write a parser to parse MiniJava Variant program. The parser should be able to produce an AST for any syntactically valid MiniJava Variant program.

Whenever the parser detects a lexing or parsing error (excpet those specified in the “Error Recovery” section), it should produce an error message and terminate.

In this task, you will probably need to

* Modify minijava.jj to support new grammar rules (to be able to parse MiniJava Variant program correctly)
* Introduce new classes (in the “syntaxtree” package) to model new kinds of node in the AST
* Add semantic actions to minijava.jj to construct new kinds of node in the AST

**Task 2** (Please name this Visitor class file as **Task2Visitor.java**)

Modify “PrettyPrintVisitor.java” so that it can support the modified grammar. That is, the class should be able to pretty print a MiniJava Variant program when given an AST built by your parser in Task 1.

You should be able to obtain an identical AST from the MiniJava Variant program outputted by your version of PrettyPrintVisitor.java.

**Task 3** (Please name this Visitor class file as **Task3Visitor.java**)

Implement a visitor named “TranslateToJava.java” that can produce a semantically identical Java program but without the FOR loops and without the exponential operator.

For this task, you may assume the input is a syntactically and semantically valid MiniJava Variant program. (So that your code should always be able to produce a semantically correct Java program.)

**Error Recovery**

Your parser should be able to recovery from certain syntax errors **(yet to be determined)** so that it can detect as many recoverable syntax errors as possible in one pass.

Even if your parser can recover from the specified errors, it should still terminate after it has parsed the whole program.

**Other Requirements**

Please design your programs so that it takes input from the standard input stream (System.in) so that your program can be executed on the command line as

**java YourProgram < input\_file**

**Submission**

Please submit a ZIP file containing the following files to Blackboard:

1. All your files needed to build your solution.
2. The input files you used to test your program.
3. **README.txt** – A readme file with the following info
   1. Description of the files you added or modified (with respect to the files in chap4.zip)
   2. Instructions to run your program.
   3. The assumptions you make, if any.
   4. Acknowledgement of third party code, files, and library (besides those included in chap4.zip) you used in your solution.
   5. Any additional information that could help us understand your implementation.

You can assume we have JDK8 (java, javac) and JavaCC (javacc) installed and they can be executed directly on command line.