CS 411 Project 2

- To run this project, import into netbeans and then run

```
Program        ::= Decl                                                    //1
                |
                Decl Program                                               //2
                ;

Decl           ::= VariableDecl                                            //3
                |
                FunctionDecl                                               //4
                |
                ClassDecl                                                  //5
                |
                InterfaceDecl                                              //6
                ;

VariableDecl    ::= Variable _semicolon                                    //7
                ;

Variable        ::= Type _id                                               //8
                ;

Type           ::= _int                                                    //9
                |
                _double                                                    //10
                |
                _boolean                                                   //11
                |
                _string                                                    //12
                |
                Type _leftbracket _rightbracket                            //13
                |
                _id                                                        //14
                ;

FunctionDecl    ::= Type _id _leftparen Formals _rightparen StmtBlock      //15
                |
                _void _id _leftparen Formals _rightparen StmtBlock         //16
                ;

Formals         ::= Variable                                               //17
```

```
                |
                Variable _comma List                                                    //18
                |
                ;                                                                       //19
List            ::= Variable _comma List                                                //20
                |
                Variable                                                                //21
                ;

ClassDecl       ::= _class _id Extends Implements _leftbrace KleeneField _rightbrace    //22
                ;

Extends         ::= _extends _id                                                        //23
                |
                ;                                                                       //24

Implements      ::= _implements IDS                                                     //25
                |
                ;                                                                       //26

KleeneField     ::= Field KleeneField                                                   //27
                |
                ;                                                                       //28

IDS             ::= _id                                                                 //29
                |
                _id _comma IDS                                                          //30
                ;

Field           ::= VariableDecl                                                        //31
                |
                FunctionDecl                                                            //32
                ;

InterfaceDecl   ::= _interface _id _leftbrace KleenePrototype _rightbrace               //33
                ;

KleenePrototype ::= Prototype KleenePrototype                                           //34
                |
                ;                                                                       //35

Prototype       ::= Type _id _leftparen Formals _rightparen _semicolon                  //36
                |
```

```
                _void _id _leftparen Formals _rightparen _semicolon                    //37
                ;

StmtBlock       ::= _leftbrace KleeneVarDec                                            //38
                ;

KleeneVarDec    ::= VariableDecl KleeneVarDec                                          //39
                |
                KleeneStmt                                                             //40
                ;

KleeneStmt      ::= Stmt KleeneStmt                                                    //41
                |
                _rightbrace                                                            //42
                ;

Stmt            ::= Expr _semicolon                                                    //43
                |
                _semicolon                                                             //44
                |
                IfStmt                                                                 //45
                |
                WhileStmt                                                              //46
                |
                ForStmt                                                                //47
                |
                BreakStmt                                                              //48
                |
                ReturnStmt                                                             //49
                |
                PrintStmt                                                              //50
                |
                StmtBlock                                                              //51
                ;

IfStmt          ::= _if _leftparen Expr _rightparen Stmt    %prec _if                  //52
                |
                _if _leftparen Expr _rightparen Stmt _else Stmt                        //53
                ;

WhileStmt       ::= _while _leftparen Expr _rightparen Stmt                            //54
                ;
```

```
ForStmt        ::= _for _leftparen Expr _semicolon Expr _semicolon Expr _rightparen Stmt   //55
                |
                _for _leftparen _semicolon Expr _semicolon Expr _rightparen Stmt            //56
                |
                _for _leftparen Expr _semicolon Expr _semicolon _rightparen Stmt            //57
                |
                _for _leftparen _semicolon Expr _semicolon _rightparen Stmt                 //58
                ;


BreakStmt      ::= _break _semicolon                                                        //59
                ;

ReturnStmt     ::= _return Expr _semicolon                                                  //60
                |
                _return _semicolon                                                          //61
                ;
PrintStmt      ::= _println _leftparen PositiveExpr _rightparen _semicolon                  //62
                ;

PositiveExpr   ::= Expr _comma PositiveExpr                                                 //63
                |
                Expr                                                                        //64
                ;

Expr           ::= Lvalue _assignop Expr                                                    //65
                |
                Constant                                                                    //66
                |
                Lvalue                                                                      //67
                |
                Call                                                                        //68
                |
                _leftparen Expr _rightparen                                                 //69
                |
                Expr _plus Expr                                                             //70
                |
                Expr _minus Expr                                                            //71
                |
                Expr _multiplication Expr                                                   //72
                |
                Expr _division Expr                                                         //73
                |
```

```
                Expr _mod Expr                                                  //74
                |
                _minus Expr                                                     //75
                |
                Expr _less Expr                                                 //76
                |
                Expr _lessequal Expr                                            //77
                |
                Expr _greater Expr                                              //78
                |
                Expr _greaterequal Expr                                         //79
                |
                Expr _equal Expr                                                //80
                |
                Expr _notequal Expr                                             //81
                |
                Expr _and Expr                                                  //82
                |
                Expr _or Expr                                                   //83
                |
                _not Expr                                                       //84
                |
                _readln _leftparen _rightparen                                  //85
                |
                _newarray _leftparen _intconstant _comma Type _rightparen       //86
                ;

Lvalue          ::= _id                                                         //87
                |
                Lvalue _leftbracket Expr _rightbracket                          //88
                |
                Lvalue _period _id                                              //89
                ;

Call            ::= _id _leftparen Actuals _rightparen                          //90
                |
                _id _period _id _leftparen Actuals _rightparen                  //91
                ;

Actuals         ::= PositiveExpr                                                //92
                |
                ;                                                               //93
```

```
Constant      ::= _intconstant                                    //94
                |
                _doubleconstant                                   //95
                |
                _stringconstant                                   //96
                |
                _booleanconstant                                  //97
                ;
```

Originally, I started with two conflicts. One was an issue with the if else statement, but I created a way to give priority to the else token if it appears which fixed this issue. The other conflict I had was the one with Type and Lvalue when they produce the _id token. The professor posted a solution to this conflict that was caused by left recursion, but when I resolved the conflict and tried to parse any statement it would just stop partially through the file. When using the debug_parse method of the parser it would show that any time it shifted an id, it would just end the program for some strange reason. The parser appeared to be really buggy. So I left the conflict there, and although I still faced some problems, it was better than what it was with 0 conflicts. Another issue I noticed is that it would terminate the program similarly (without actually having a syntax error) when I would declare a function without parameters. So when testing the toy program from project 1, the project would stop on the second function. I threw in an “int x” as a parameter and it would continue parsing after that. Then, I faced another issue with the class declaration. Similarly to the function declaration, once the left bracket was reached the program would just terminate but without error. These are issues I faced since Saturday and I rewrote my grammar multiple times but couldn’t resolve it. It just appeared as if the parser was riddled with issues for me.

As far as the rest of the program goes, I spent a lot of time trying to figure out how to format the output like the professor requested. But there isn’t much information about Java Cup out there and the implementation isn’t available anywhere either so it’s hard to find where exactly the actions are being done. Through playing with the parser.java file, I found that the parser doesn’t output anything until it has actually chosen a rule, so it is seemingly impossible to actually print the shift or reduce actions with each given token, because the tokens are printed out prior to the rule being decided. A good alternative was the parser debug_parse method, but this method only outputs the int value of the token and has a lot of bloated information that isn’t needed. I still used this for the project 1 toy program test because it was much clearer than my formatting, but its still tedious to switch back and forth between the grammar page and symbols page. Ultimately, my output is really ugly but it basically prints out all the tokens involved and the production rules used in the parse. If the test isn’t syntactically correct, than the program will print out an error and end.

As for the other test cases I created, this one is valid, and is a function with 2 variable declarations, 2 parameters, has a for loop, and a return statement. The program accepts this correctly.

```
double function (int x, int y) {
```

```
    int a;
    int b;
    for (a; i < a; i)
    {
        b = b + b;
    }
    return b;
}
```

My other case proved that the if else was working. It was inside of a function declaration, so the project accepted it.

```
void f (double x, double y) {
    if (h>w) g=h;
    else h=g;
}
```

For the other test cases:

1: 1 4 16 38 42 18 21 8 10 8 10 were the expected productions and those were correctly given

2. Error, cannot assign. Only variable declarations are allowed

3. This also fails because there is no type and the id is just assigned a value. But it doesn't fit the variable, function, class or interface declaration

4. Same reasons as above.

5. Same reasons as above.

6. 1 3 7 8 13 13 13 9 and correctly given

7. This was incorrect because it can be declared, but not assigned a value. Then double assignment as such wouldn't work either

8. This also wouldn't be accepted unless it was inside of a function or class.

9. This one, I'm uncertain if it should be accepted or not. When inside of a function, my project rejected it once it reached the double declaration.

10. My class declaration would cause the program to terminate (without error) for some reason, so I can't explain this one.

Ultimately, I spent a considerable amount of time on this project but couldn't get the expected result and was basically lost as to how I could fix some of these issues so I essentially gave up.