Passing & Failing Test Cases

The first tests the string ***void f (double x, double y) { }*** which passes through the parser and generates the parse tree. We tried to modify the format to show it in a stack form vs rule output, however we didn't have access to the tokens once they were parsed. Through having the format as a tree we hope to showcase how the parser traversed through the CFG.

The second tests the string

```
int fact (int x) {
// recursive factorial function
if (x>1) return x * fact(x-1);
else return 1;
}
void main () {
/* Fall Semester 2018
CS 4110 project #1
A lexical analyzer */
int x;
int total;
println ("factorial of 10 is ", fact (10), "from the recursive function");
total = 1; x = 1;
for ( ; x<=10; ) { total = total * x; x = x + 1; }
println ("iterative result of 10! is ", total);
}
class cs4110 {
int Funny;
double funny;
boolean flag;
string s;
int [] a;
flag = true;
Funny = 0X89aB; funny = 123456E+7;
s = "hello world";
while (x = (Funny/10) <0) println (s, " have fun !");
a = newarray (20, int);
```

}

This string fails at line 23 due to the string *flag = true;* because we do not allow assignments outside a function which is the only place where expressions are allowed. Inside a class statement block we only allow variable declarations or function declarations (which then would allow the assignment expression).

C:\Users\Working\Documents\EclipseWorkspace\ Fall2018\Compilers Project 2\bin>java compilers_project_2.MyNewGrammar ../test/test1.txt file specified! Reading from file...	Printing parse tree...
	Start
	->Program
	->->Decl
	->->->FunctionDecl
_void	->->->->Formals
_id	->->->->->VariableList
_leftparen	->->->->->->Variable
_double	->->->->->->->Type
_id	->->->->->VariableList
_comma	->->->->->->Variable
_double	->->->->->->->Type
_id	->->->StmtBlock
_rightparen	->->->->StmtBlockPrime
_leftbrace	
_rightbrace	

C:\Users\Working\Documents\EclipseWorkspace\ Fall2018\Compilers Project 2\bin>java compilers_project_2.MyNewGrammar ../test/test11.txt file specified! Reading from file...	_intconstant	_id
	_semicolon	_assignop
	_rightbrace	_intconstant
	_void	_semicolon
_int	_id	_for
_id	_leftparen	_leftparen
_leftparen	_rightparen	_semicolon
_int	_leftbrace	_id
_id	_int	_lessequal
_rightparen	_id	_intconstant
_leftbrace	_semicolon	_semicolon
_if	_int	_rightparen
_leftparen	_id	_leftbrace
_id	_semicolon	_id
_greater	_println	_assignop
_intconstant	_leftparen	_id
_rightparen	_stringconstant	_multiplication
_return	_comma	_id
_id	_id	_semicolon
_multiplication	_leftparen	_id
_id	_intconstant	_assignop
_leftparen	_rightparen	_id
_id	_comma	_plus
_minus	_stringconstant	_intconstant
_intconstant	_rightparen	_semicolon
_rightparen	_semicolon	_rightbrace
_semicolon	_id	_println
_else	_assignop	_leftparen
_return	_intconstant	
	_semicolon	

_stringconstant	_id	_semicolon
_comma	_assignop	_id
_id	_intconstant	_assignop
_rightparen	_semicolon	_newarray
_semicolon	_id	_leftparen
_rightbrace	_assignop	_intconstant
_class	_intconstant	_comma
_id	_semicolon	_int
_leftbrace	_id	_rightparen
_int	_assignop	_semicolon
_id	_stringconstant	_rightbrace
_semicolon	_semicolon	
_double	_while	
_id	_leftparen	NOK.
_semicolon	_id	Encountered " "=" "= "" at line
_boolean	_assignop	23, column 6.
_id	_leftparen	Was expecting one of:
_semicolon	_id	"[" ...
_string	_division	< _id> ...
_id	_intconstant	"[" ...
_semicolon	_rightparen	< _id> ...
_int	_less	
_leftbracket	_intconstant	
_rightbracket	_rightparen	
_id	_println	
_semicolon	_leftparen	
_id	_id	
_assignop	_comma	
_booleanconstant	_stringconstant	
_semicolon	_rightparen	