```
Summary file for decaf-daniel-byers
Name: put your name here
Student number: put your student number here
Lab class: put the day and time of your Software Architectures lab class here
********************
   PARSER TESTING
*** Legal tests ***
15/15 correct parser test cases
*** Legal tests with debug ***
*** Illegal tests ***
illegal-01 generated expected error
illegal-02 generated expected error
illegal-03 generated error:
line 2:8 missing INTLITERAL at ']'
illegal-04 generated expected error
illegal-05 generated expected error
illegal-06 generated expected error
illegal-07 generated error:
line 3:8 mismatched input 'callout' expecting IDENTIFIER
line 3:15 mismatched input ';' expecting '('
illegal-08 generated error:
line 4:14 mismatched input '=' expecting {';', '+', '-', '/', '*', '%', '==', '!
=', '<', '>', '<=', '>=', '&&', '||'}
illegal-09 generated expected error
illegal-10 generated error:
line 6:2 extraneous input 'void' expecting {'for', 'break', 'if', 'callout', 're
turn', 'continue', '{', '}', IDENTIFIER}
illegal-11 generated error:
line 7:7 missing IDENTIFIER at ';'
illegal-12 generated expected error
illegal-13 generated expected error
illegal-14 generated error:
line 2:11 extraneous input 'a' expecting {'boolean', 'int', 'void', ')'}
illegal-15 generated error:
line 3:8 extraneous input 'int' expecting {'callout', '(', ')', '-', NOT, INTLIT
ERAL, BOOLEANLITERAL, CHARLITERAL, STRINGLITERAL, IDENTIFIER}
illegal-16 generated error:
line 3:8 mismatched input ')' expecting {'callout', '(', '-', NOT, INTLITERAL, B
OOLEANLITERAL, CHARLITERAL, STRINGLITERAL, IDENTIFIER}
illegal-17 generated error:
line 3:4 mismatched input '0xcafe' expecting {'boolean', 'for', 'break', 'if', '
callout', 'int', 'return', 'continue', 'void', '{', '}', IDENTIFIER}
line 5:0 extraneous input '}' expecting <EOF>
illegal-18 generated error:
line 3:12 mismatched input '5' expecting STRINGLITERAL
illegal-19 generated error:
line 5:11 no viable alternative at input 'forpari'
line 5:16 mismatched input ',' expecting {';', '+', '-', '/', '*', '%', '==', '!
=', '<', '>', '<=', '>=', '&&', '||'}
illegal-20 generated error:
line 5:11 no viable alternative at input 'forpari'
line 5:16 mismatched input ',' expecting {';', '+', '-', '/', '*', '%', '==', '!
=', '<', '>', '<=', '>=', '&&', '||'}
    20 /20 correctly detected parser errors
```

Please make sure that you only give extra output iff the -debug flag is used (otherwise crashes automated testing).

```
/**
 * @author Daniel Byers | 13121312
 *
 * This code builds on examples provided by the following book:
 * Parr, Terence (2012). The Definitive ANTLR 4 Reference. USA: The Pragmatic Bo
okshelf. 322.
 */

parser grammar DecafParser;
options { tokenVocab = DecafLexer; }

// top level. The context of the whole Decaf application.
program: CLASS IDENTIFIER LCURLY fieldDecl* methodDecl* RCURLY EOF;

// global scope. Arrays are only allowed there!
// int a; int a[10]; int a, b; int a[1], b, c;
fieldDecl : type (IDENTIFIER | arrayDecl) (COMMA (IDENTIFIER | arrayDecl))* EOL;

// This is a "named indentifier" so a method is generated to access it through
// the FieldDeclContext object.
arrayDecl: IDENTIFIER LBRACE INTLITERAL RBRACE;

methodDecl:
  (type | VOID) methodName LPAREN ((type IDENTIFIER) (COMMA type IDENTIFIER)*)?
RPAREN block;

// This is a "named indentifier" so a method is generated to access it through
// the MethodDeclContext object.
methodName: IDENTIFIER;

// Variables have to be declared first.
block: LCURLY varDecl* statement* RCURLY;

// local scope
varDecl: type IDENTIFIER (COMMA IDENTIFIER)* EOL;

// basic types.
type: (INT | BOOLEAN | VOID);

statement :
          ( location assignOp expr EOL
          | methodCall EOL
          | IF LPAREN expr RPAREN block (ELSE block)?
          | FOR IDENTIFIER ASSIGNMENT expr COMMA expr block
          | RETURN expr? EOL
          | BREAK EOL
          | CONTINUE EOL
          | block
          );

assignOp: (ASSIGNMENT | ASSIGNMENTP | ASSIGNMENTS);

methodCall :
           ( methodName LPAREN (expr (COMMA expr)*)? RPAREN
           | CALLOUT LPAREN STRINGLITERAL (COMMA calloutArg (COMMA calloutArg)*
)? RPAREN);

calloutArg: (expr | STRINGLITERAL);

location: (IDENTIFIER | IDENTIFIER LBRACE expr RBRACE);

// MINUS causes an ambguity check to be raised because it's not clear
```



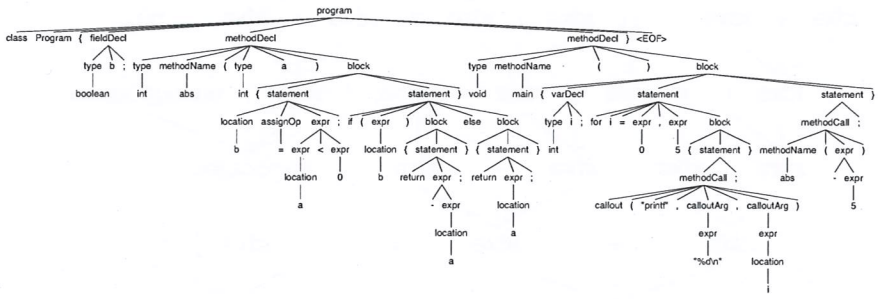
```
// if "x-foo()"" is "- foo()" or "-foo()".
// ANTLR has a "longest-match" rule that is used in these situations.
expr:   MINUS expr
      | NOT expr
      | expr (MULTIPLY | DIVISION | MODULO) expr
      | expr (ADDITION | MINUS) expr
      | expr (LESSTHAN | GREATERTHAN | LSSTHNEQTO | GRTTHNEQTO) expr
      | expr (EQUAL | NOTEQUAL) expr
      | expr AND expr
      | expr OR expr
      | location
      | methodCall
      | (INTLITERAL | CHARLITERAL | BOOLEANLITERAL | STRINGLITERAL)
      | LPAREN expr RPAREN
    ;
```

NO!

But otherwise, excellent work.

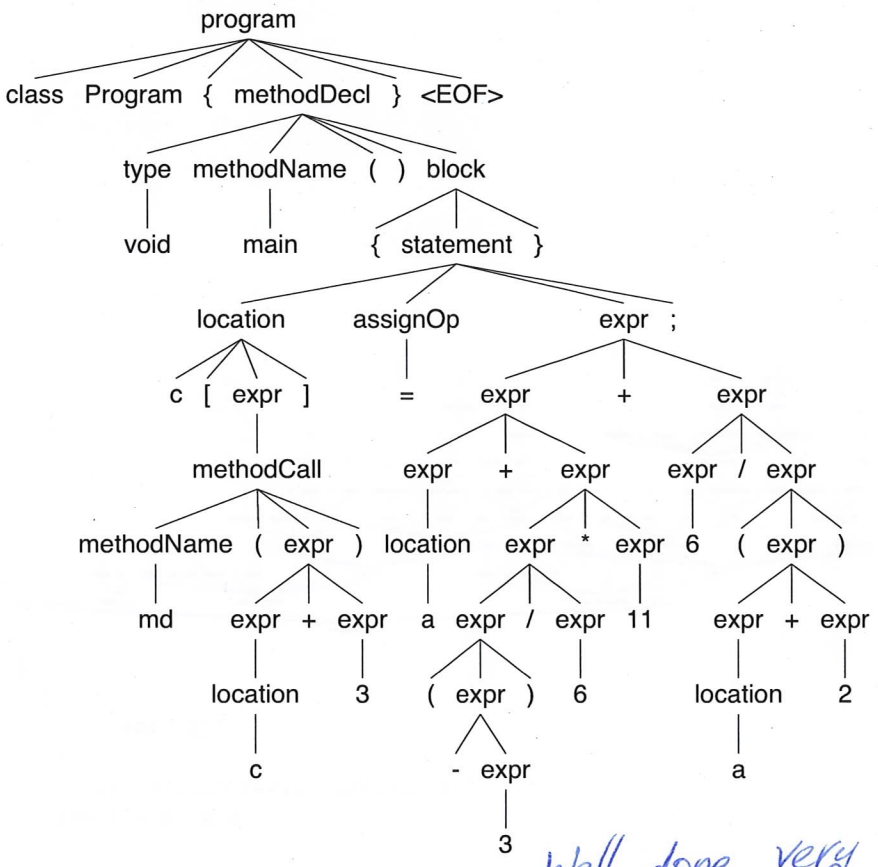
01

```
class Program {
  boolean b;

  int abs(int a) {
    b = a < 0;

    if (b) {
      return -a;
    }
    else {
      return a;
    }
  }

  void main() {
   int i;

   for i = 0, 5 {
      callout("printf", "%d\n", i);
   }
        abs(-5);
  }
}
```



02

```
class Program {
  void main() {
    c[md(c+3)] = a + (-3) / 6 * 11 + 6 / (a + 2);
  }
}
```



Well done, very few generated correct tree here.