UNIVERSIDAD SERGIO ARBOLEDA

CALCULADORA EN ANTLR4

DOCENTE

JOAQUIN SANCHEZ CIFUENTES

PRESENTADO POR

MARIO JIMENEZ LOPEZ

26 FEBRERO 2026

```
majilo@majiloPC:~/Universidad/lenguajesProgramacion/Antlr4/antCalculadora$ nano Expr.g4
majilo@majiloPC:~/Universidad/lenguajesProgramacion/Antlr4/antCalculadora$ ls
Expr.g4
majilo@majiloPC:~/Universidad/lenguajesProgramacion/Antlr4/antCalculadora$
```

En la carpeta creada para realizar la implementación de la calculadora se genera un nano en el cual se le añadió la gramatica de la calculadora

```
 GNU nano 7.2                                  Expr.g4
grammar Expr;

prog:   stat+ ;

stat:   expr NEWLINE                # printExpr
    |   ID '=' expr NEWLINE         # assign
    |   NEWLINE                     # blank
    ;

expr:   expr op=('*'|'/') expr      # MulDiv
    |   expr op=('+'|'-') expr      # AddSub
    |   INT                         # int
    |   ID                          # id
    |   '(' expr ')'                # parens
    ;

MUL :   '*' ;
DIV :   '/' ;
ADD :   '+' ;
SUB :   '-' ;
ID  :   [a-zA-Z]+ ;
INT :   [0-9]+ ;
NEWLINE:'\r'? '\n' ;
WS  :   [ \t]+ -> skip ;



^G Ayuda       ^O Guardar     ^W Buscar      ^K Cortar      ^T Ejecutar    ^C Ubicación
^X Salir       ^R Leer fich.  ^\ Reemplazar  ^U Pegar       ^J Justificar  ^/ Ir a línea
```

Se ejecuta el codigo “antlr4 Expr.g4” para generar el parser y el lexer del archivo para posteriormente compilarlo con java

```
majilo@majiloPC:~/Universidad/lenguajesProgramacion/Antlr4/antCalculadora$ antlr4 Expr.g4
majilo@majiloPC:~/Universidad/lenguajesProgramacion/Antlr4/antCalculadora$ javac *.java
majilo@majiloPC:~/Universidad/lenguajesProgramacion/Antlr4/antCalculadora$ ls
 ExprBaseListener.class   ExprListener.class               'ExprParser$MulDivContext.class'
 ExprBaseListener.java    ExprListener.java                'ExprParser$ParensContext.class'
 Expr.g4                 'ExprParser$AddSubContext.class'  'ExprParser$PrintExprContext.class'
 Expr.interp             'ExprParser$AssignContext.class'  'ExprParser$ProgContext.class'
 ExprLexer.class         'ExprParser$BlankContext.class'   'ExprParser$StatContext.class'
 ExprLexer.interp        'ExprParser$ExprContext.class'     ExprParser.class
 ExprLexer.java          'ExprParser$IdContext.class'       ExprParser.java
 ExprLexer.tokens        'ExprParser$IntContext.class'      Expr.tokens
majilo@majiloPC:~/Universidad/lenguajesProgramacion/Antlr4/antCalculadora$
```

Se realizan pruebas con el grun para probar el arbol del parseo

```
^Cmajilo@majiloPC:~/Universidad/lenguajesProgramacion/Antlr4/antCalculadora$ grun Expr prog -tree
1+2
(prog (stat (expr (expr 1) + (expr 2)) \n))
majilo@majiloPC:~/Universidad/lenguajesProgramacion/Antlr4/antCalculadora$ grun Expr prog -tree
1line 1:1 missing NEWLINE at '<EOF>'
(prog (stat (expr 1) <missing NEWLINE>))
majilo@majiloPC:~/Universidad/lenguajesProgramacion/Antlr4/antCalculadora$
```

Y se evidencia en la 2da prueba un “error” ya que hizo falta un salto de linea segun la gramatica que se establecio en el archivo de gramatica

```
majilo@majiloPC:~/Universidad/lenguajesProgramacion/Antlr4/antCalculadora$ antlr4 -visitor Expr.g
4
majilo@majiloPC:~/Universidad/lenguajesProgramacion/Antlr4/antCalculadora$ javac *.java
lsmajilo@majiloPC:~/Universidad/lenguajesProgramacion/Antlr4/antCalculadora$ ls
 ExprBaseListener.class    ExprListener.class               'ExprParser$PrintExprContext.class'
 ExprBaseListener.java     ExprListener.java                'ExprParser$ProgContext.class'
 ExprBaseVisitor.class    'ExprParser$AddSubContext.class'  'ExprParser$StatContext.class'
 ExprBaseVisitor.java     'ExprParser$AssignContext.class'   ExprParser.class
 Expr.g4                  'ExprParser$BlankContext.class'    ExprParser.java
 Expr.interp              'ExprParser$ExprContext.class'     Expr.tokens
 ExprLexer.class          'ExprParser$IdContext.class'       ExprVisitor.class
 ExprLexer.interp         'ExprParser$IntContext.class'      ExprVisitor.java
 ExprLexer.java           'ExprParser$MulDivContext.class'
 ExprLexer.tokens         'ExprParser$ParensContext.class'
majilo@majiloPC:~/Universidad/lenguajesProgramacion/Antlr4/antCalculadora$ nano EvalVisitor.java
majilo@majiloPC:~/Universidad/lenguajesProgramacion/Antlr4/antCalculadora$
```

Luego se generará el visitor del archivo Expr.g4, para el evaluador del visitor para posterior mente crear el main y el archivo de entrada de texto.

Para probar el funcionamiento se creo el archivo de texto con varias instrucciones y asignaciones de variables.

```
majilo@majiloPC:~/Universidad/lenguajesProgramacion/Antlr4/antCalculadora
$ nano t.expr
majilo@majiloPC:~/Universidad/lenguajesProgramacion/Antlr4/antCalculadora
$
majilo@majiloPC:~/Universidad/lenguajesProgramacion/Antlr4/antCalculadora
$ cat t.expr
193
a = 5
b = 6
a+b*2
(1+2)*3
majilo@majiloPC:~/Universidad/lenguajesProgramacion/Antlr4/antCalculadora
$ java Calc t.expr
193
17
9
majilo@majiloPC:~/Universidad/lenguajesProgramacion/Antlr4/antCalculadora
$
```

El programa realiza:

Lectura del archivo para realizar el analisis lexico en el cual el texto se divide en tokens identificando cada simbolo, analisis sintáctico en esta etapa desde los tokens se construye el arbol sintactico siguiendo las reglas de la gramatica para finalmente la semantica en el cual se recorre el arbol sintactico para resultar los resultados de las expresiones