Trabalho Prático I – Compiladores

Gramática

```
Program ::= MainClass ( ClassDeclaration )* <EOF>
MainClass ::= "class" Identifier "{" "public" "static" "void"
              "main" "(" "String" "[" "]" Identifier ")"
              "{" Statement "}" "}"
ClassDeclaration ::= "class" Identifier ( "extends" Identifier )? "{"
              ( VarDeclaration )* ( MethodDeclaration )*
              "}"
VarDeclaration ::= Type Identifier ";"
MethodDeclaration ::= "public" Type Identifier "("
              ( Type Identifier ( "," Type Identifier )* )? ")"
              "{" ( VarDeclaration )* ( Statement )*
              "return" Expression ";" "}"
Type ::= "int" "[" "]"
       | "boolean"
       | "int"
       | Identifier
Statement ::= "{" ( Statement )* "}"
       | "if"
         "(" Expression ")" Statement "else" Statement
       | "while" "(" Expression ")" Statement
       | "System.out.println" "(" Expression ")" ";"
       | Identifier "=" Expression ";"
       | Identifier "[" Expression "]"
         "=" Expression ";"
Expression ::= Expression ( "&&" | "<" | "+" | "-" | "*"
         ) Expression
       | Expression "[" Expression "]"
       | Expression "." "length"
       | Expression "." Identifier "(" ( Expression (
         "," Expression )* )? ")"
       | <INTEGER_LITERAL>
       | "true"
       | "false"
       | Identifier
       | "this"
       | "new" "int" "[" Expression "]"
       | "new" Identifier "(" ")"
```

| "!" Expression
| "(" Expression ")"
Identifier ::= <IDENTIFIER>

Identifier: Uma sequência de letras e dígitos começando por uma letra. Maiúsculas e minúsculas precisam ser diferenciadas.

Integer Literal: É uma sequência de dígitos decimais, correspondente a um valor inteiro.

Comentários começam com /* e terminam com */.

Exemplo de um programa:

```
class Factorial{
    public static void main(String[] a){
        System.out.println(new Fac().ComputeFac(10));
    }
}

class Fac {
    public int ComputeFac(int num){
        int num_aux ;
        if (num < 1)
            num_aux = 1 ;
        else
            num_aux = num * (this.ComputeFac(num-1)) ;
        return num_aux ;
    }
}
```

1) Dado um programa, faça um analisador léxico que retorne os seguintes tokens:

   ID:
   OP: “+”, “-“, “*”, “&&”, “<”, “!”, “.”, “,”, ”=”
   DELIM: “(“, “)”, “[“, “]”, “{“, “}”, “;”
   DECL:"class" "extends" "public" "static" "void" "main" "length" "this" "new"
   FLUXO: "if", "else", "while", "return", "System.out.println"
   TIPO: "boolean", "int", "true", "false", "String"
   COMENT: “/*”, “*/”

   obs: Tokens dentro de comentários não podem ser retornados.

   Exemplo de saída:

```
<OP,=> número de ocorrências 2
<ID,x> número de ocorrências 1
```

2) Faça um ou mais AFDs do analisador.