Analizador Léxico + Sintáctico

Martín Pascual Montesinos Abarca
Departamento de Computación
Universidad Católica San Pablo
martin.montesinos@ucsp.edu.pe

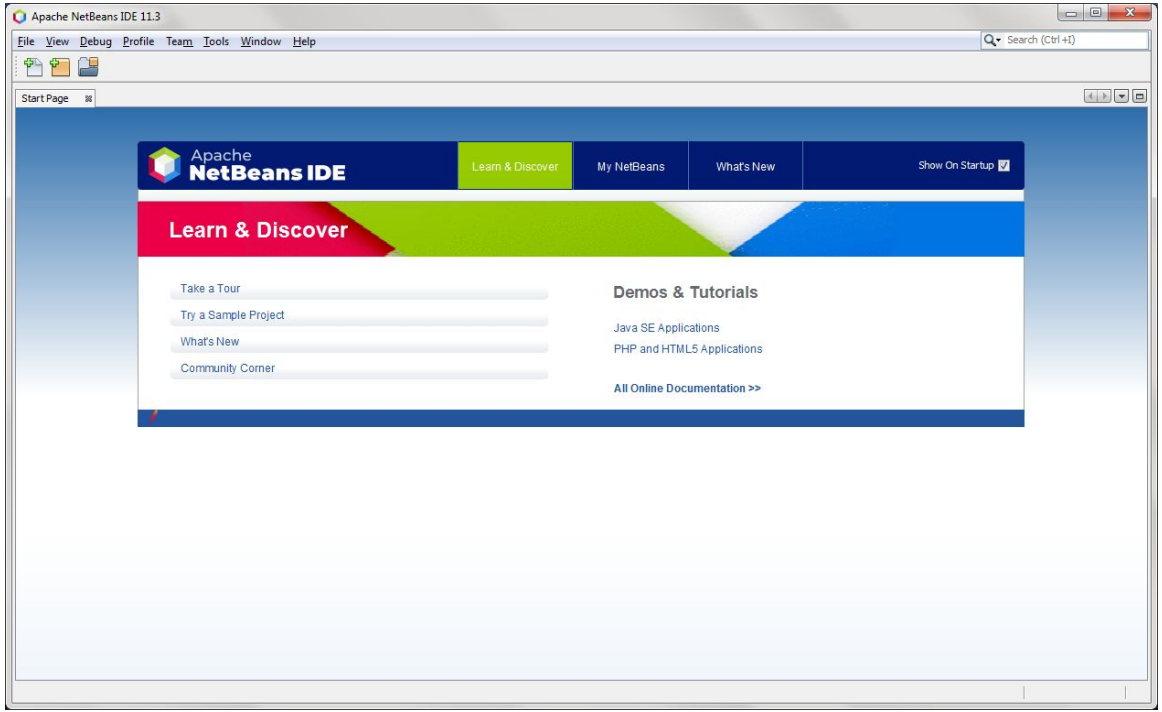
Missael Alejandro Rodriguez Ureta
Departamento de Computación
Universidad Católica San Pablo
missael.rodriguez@ucsp.edu.pe

Herramientas

antlr4, JAVA, NETBEANS 12.3

- ANTLR4(ANother Tool for Language Recognition)

NetBeans 12.3



Código

Configuración de POM

```xml
<dependencies>
    <dependency>
        <groupId>org.antlr</groupId>
        <artifactId>antlr4</artifactId>
        <version>4.7.2</version>
    </dependency>
</dependencies>
```

```xml
<build>
    <plugins>
        <plugin>
            <groupId>org.antlr</groupId>
            <artifactId>antlr4-maven-plugin</artifactId>
            <version>4.7.2</version>
            <configuration>
                <visitor>true</visitor>
            </configuration>
            <executions>
                <execution>
                    <id>antlr</id>
                    <goals>
                        <goal>antlr4</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
        <plugin>
            <artifactId>maven-assembly-plugin</artifactId>
            <configuration>
                <archive>
                    <manifest>
                        <mainClass>ucsp.compiladores.analizadores.Principal</mainClass>
                    </manifest>
                </archive>
                <descriptorRefs>
                    <descriptorRef>jar-with-dependencies</descriptorRef>
                </descriptorRefs>
            </configuration>
            <executions>
                <execution>
                    <id>make-assembly</id>
                    <phase>package</phase>
                    <goals>
                        <goal>single</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

JAVA

```java
import java.io.IOException;
import org.antlr.v4.runtime.CharStream;
import org.antlr.v4.runtime.CharStreams;
import org.antlr.v4.runtime.CommonTokenStream;
import org.antlr.v4.runtime.Token;

/**
 *
 * @authors Gino
 */
public class Principal {
    public static void main(String[] args){
        try {
            CharStream cs = CharStreams.fromFileName(args[0]);
            System.out.println("-------------- INICIO --------------");
            mini0lexerLexer lexer = new mini0lexerLexer(cs);
            CommonTokenStream tokens = new CommonTokenStream(lexer);
            mini0lexerParser parser = new mini0lexerParser(tokens);
            //parser.program();
            Token t = null;
            while((t=lexer.nextToken()).getType()!=Token.EOF){
                System.out.println("["+t.getType()+","+t.getText()+"]");
            }

        } catch (IOException ex) {
        }
        System.out.println("-------------- FIN --------------");
    }
```

La activación del analizador sintáctico se hace con parser.program()

Antlr4

```
TRUE            : 'true';

FALSE           : 'false';

RESERVEDWORDS   :  'if' | 'else' | 'end' | 'while' | 'loop' | 'fun' |
                   'return' | 'new' | 'string' | 'int' | 'char' |
                   'bool' | TRUE | FALSE | 'and' | 'or' | 'not' ;

NUMERICLITERAL  :  ( ('+'|'-')?('0'..'9') | (('0x') ('0'..'9'|'a'..'f'|'A'..'F')+) )+;

IDENTIFIER      : ('a'..'z'|'A'..'Z'|'_') ('a'..'z'|'A'..'Z'|'0'..'9'|'_')*;

STRINGLITERAL   : '"' ( ('\\'|'\n'|'\r'|'\t') | ~('\'') )* '"' ;

COMMENTBLOCK    : '/*' .*? '*/' WHITESPACE -> skip;

COMMENTLINE     : '//' ~('\n'|'\r')* '\r'? '\n' -> skip;

NEWLINE         : ('\n')+;

WHITESPACE      : ( ' ' |'\t' | '\r' | NEWLINE) -> skip;

RELATIONALOP    : '>' | '>=' | '<' | '<=' | '=' | '<>';

ARITHMETICOP    : '+' | '-' | '*' | '/';

PUNCTUATION     : '(' | ')' | ',' | ':' | '[' | ']';
```

```
39  program         : NEWLINE* declaration (declaration)* EOF;
40
41  declaration     : function | global;
42
43  newLine         : NEWLINE NEWLINE*;
44
45  global          : declvar newLine;
46
47  function        : 'fun' IDENTIFIER '(' parameters? ')' (':' type)? newLine
48                  /*java code*/
49                  { System.out.println("Funcion:\n\tNombre="+$IDENTIFIER.text+", Tipo="+$type.text); }
50                  block 'end' NEWLINE;
51
52  block           : (declvar newLine)*(cmd=command
53                  /*java code*/
54                  { System.out.println("Tipo de comando:\n\t"+$cmd.X); }
55                  newLine)*;
56
57  parameters      : parameter (',' parameter)*;
58
59  parameter       : IDENTIFIER ':' type;
60
61  type            : baseType | '[' ']' type;
62
63  baseType        : 'int' | 'bool' | 'char' | 'string';
64
```

```
declvar          : IDENTIFIER ':' type
                 /*java code*/
                 { System.out.println("Declaracion:\n\tNombre="+$IDENTIFIER.text+", Tipo="+$type.text); };

command  returns [ String X ]:
                 commandIf
                    { $X = "if"; }
                 | commandWhile
                    { $X = "while"; }
                 | commandAsign
                    { $X = "asign"; }
                 | commandReturn
                    { $X = "return"; }
                 | call
                    { $X = "call"; };

commandIf        : 'if' expression newLine block ('else' 'if' expression newLine block)* ('else' newLine block)? 'end';

commandWhile     : 'while' expression newLine block 'loop';

commandAsign     : variable '=' expression
                 /*java code*/
                 { System.out.println("Variable:\n\t"+$variable.text+" = "+$expression.text); };
```

Pruebas Léxicas

Strings

```
C:\Users\Martin\Documents\NetBeansProjects\analizadores\target>java -jar analiza
dores-1.0-SNAPSHOT-jar-with-dependencies.jar 06-strings.m0
-------------- INICIO --------------
[37,"soy un string"
"contengo un escape \\"
"\\\\ hace el escape de \\ en strings"
"un enter se hace con \\n"
"tab\ttab\ttab"
"vea: \nsoy otra linea"
"comillas dentro de \"strings\" asi"
"se escribe \" usando \\\""
""
" yo no soy un comentario: /* hello */ "]
[40,
]
-------------- FIN --------------
```

```
1  "soy un string"
1  "contengo un escape \\"
2  "\\\\ hace el escape de \\ en strings"
3  "un enter se hace con \\n"
4  "tab\ttab\ttab"
5  "vea: \nsoy otra linea"
6  "comillas dentro de \"strings\" asi"
7  "se escribe \" usando \\\""
8  ""
9  " yo no soy un comentario: /* hello */ "
```

Palabras reservadas

```
[13,if]
[14,else]
[5,end]
[15,while]
[16,loop]
[1,fun]
[18,return]
[19,new]
[36,goto]
[12,string]
[9,int]
[11,char]
[10,bool]
[32,true]
[33,false]
[29,and]
[30,or]
[31,not]
[40,
]
[37,"this and that are not keywords"]
[40,
]
[36,function]
[40,
]
[25,<]
[5,end]
[24,>]
[40,
]
[27,<=]
[5,end]
[17,=]
[24,>]
[40,
]
[2,(]
[36,x]
[24,>]
[35,2]
[3,)]
[29,and]
[2,(]
[36,y]
[25,<]
[35,3]
[3,)]
[40,
]
[36,returnif]
[40,
]
```

```
1  if else end while loop fun return new goto string int char bool true false
   and or not
1  "this and that are not keywords"
2  /* or these either */
3  function
4  <end>
5  <=end=>
6  (x>2)and(y<3)
7  returnif
8  andor
```

Números

```
C:\Users\Martin\Documents\NetBeansProjects\analizadores\target>java -jar analiza
dores-1.0-SNAPSHOT-jar-with-dependencies.jar 08-ints.m0
-------------- INICIO --------------
[35,0]
[40,
]
[35,1]
[40,
]
[35,15]
[40,
]
[35,-2]
[40,
]
[35,450]
[40,
]
[35,99999999999999999999999999999]
[40,

]
-------------- FIN --------------
```

```
1   0
  1 1
  2 15
  3 -2
  4 450
  5 99999999999999999999999999999
  6
```

Hexadecimales

```
C:\Users\Martin\Documents\NetBeansProjects\analizadores\target>java -jar analiza
dores-1.0-SNAPSHOT-jar-with-dependencies.jar 09-hex.m0
-------------- INICIO --------------
[35,0x10]
[40,
]
[35,0x0]
[40,
]
[35,0]
[36,x]
[40,
]
[35,0xa]
[40,
]
[35,0x1B]
[40,
]
[35,0xaAaA]
[40,
]
[35,0xa0xa]
[40,
]
[35,0xffffffffffffffffffffffffffffffffffffffff]
[40,
]
-------------- FIN --------------
```

```
1  0x10
1 0x0
2 0x
3 0xa
4 0x1B
5 0xaAaA
6 0xa0xa
7 0xffffffffffffffffffffffffffffffffffffffff
```

Pruebas Sintácticas

If y while

```
D:\MMA\Universidad\Compiladores\trabajofinal\analizadores\target>java -jar analizadores-1
.0-SNAPSHOT-jar-with-dependencies.jar 14-ifwhile.m0
-------------- INICIO --------------
Declaracion:
        Nombre=gobo, Tipo=int
Funcion:
        Nombre=hehe, Tipo=bool
Declaracion:
        Nombre=i, Tipo=int
Variable:
        i = 0
Tipo de comando:
        asign
Tipo de comando:
        call
Tipo de comando:
        if
Variable:
        i = i
Tipo de comando:
        asign
line 14:14 extraneous input '+1' expecting NEWLINE
Tipo de comando:
        while
Tipo de comando:
        return
Tipo de comando:
        return
Tipo de comando:
```

Ejecución de la prueba

```
1
2  gobo:int
3
4  fun hehe(c:char):bool
5     i: int
6     i = 0
7     if c = 64
8        while i < 10
9           if i / 2 * 2 = i
10             printf("%d\n", i)
11          else
12             printf("******\n", i)
13          end
14          i = i+1
15       loop
16       return true
17    else
18       return false
19    end
20 end
```

Prueba 14

Declaración de variables

```
D:\MMA\Universidad\Compiladores\trabajofinal\analizadores\target>java -jar
 analizadores-1.0-SNAPSHOT-jar-with-dependencies.jar 06-declvar.m0
-------------- INICIO --------------
Sintacticas
Funcion:
        Nombre=foo, Tipo=null
Declaracion:
        Nombre=x, Tipo=[]int
Variable:
        x = new[10]int
Tipo de comando:
        asign
Funcion:
        Nombre=_, Tipo=null
Declaracion:
        Nombre=x, Tipo=[]int
Declaracion:
        Nombre=y, Tipo=bool
Variable:
        x = new[10]int
Tipo de comando:
        asign
Tipo de comando:
        call
Tipo de comando:
        if
-------------- FIN --------------
```

```
1
1  fun foo()
2     x:[]int
3     x = new [10]int
4  end
5
6  fun _()
7     x:[]int
8     y: bool
9     x = new [10]int
10    if y = true
11       print("ola")
12    end
13 end
14
```

Falla de expresión

```
D:\MMA\Universidad\Compiladores\trabajofinal\analizadores\target>java -jar analizadores-1
.0-SNAPSHOT-jar-with-dependencies.jar 22-fail-exp.m0
------------- INICIO -------------
Funcion:
        Nombre=exps, Tipo=null
line 3:15 extraneous input 'or' expecting {'(', 'new', '-', 'not', 'true', 'false', NUMER
ICLITERAL, IDENTIFIER, STRINGLITERAL}
Tipo de comando:
        call
Tipo de comando:
        if
------------- FIN -------------

D:\MMA\Universidad\Compiladores\trabajofinal\analizadores\target>
```

Ejecución de la prueba

```
1 fun exps()
2    if 2 < x <= or 5
3       print("woof")
4    end
5 end
6
```

Prueba 22

Fallo en función

```
D:\MMA\Universidad\Compiladores\trabajofinal\analizadores\target>java -jar analizadores-1.0-SNAPSHOT-jar-with-dependenci
es.jar 25-fail-fun3.m0
-------------- INICIO --------------
Sintacticas
line 2:6 extraneous input '12' expecting {')', IDENTIFIER}
Funcion:
        Nombre=x, Tipo=null
-------------- FIN --------------

D:\MMA\Universidad\Compiladores\trabajofinal\analizadores\target>
```

```
1
  1 fun x(12)
  2 end
~
```

Repositorio

https://github.com/pascmma/TrabajoCompiladores

Analizador Léxico + Sintáctico

Martín Pascual Montesinos Abarca
Departamento de Computación
Universidad Católica San Pablo
martin.montesinos@ucsp.edu.pe

Missael Alejandro Rodriguez Ureta
Departamento de Computación
Universidad Católica San Pablo
missael.rodriguez@ucsp.edu.pe