

Práctica 1

Pulido Reséndiz Aarón Isai

Lenguajes y Autómatas II

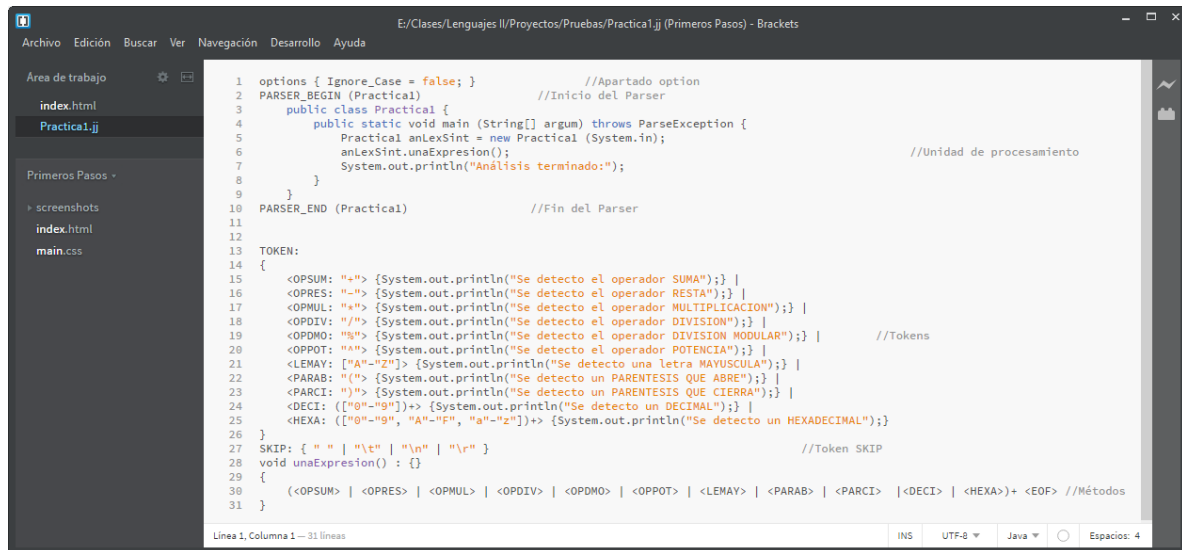
Introducción

En esta práctica se verán y aplicarán conceptos vistos anteriormente en la materia. Se modificará una práctica sencilla propuesta por el profesor, a la cual se le agregaran más tokens y la opción de imprimir mensajes en la consola dependiendo de si introducimos:

- a. Paréntesis.
- b. Operadores aritméticos.
- c. Letras mayúsculas.
- d. Dígitos decimales.
- e. Dígitos hexadecimales.

Desarrollo

Para explicar esta práctica se pidió que se realizara una captura de pantalla del programa junto con comentarios de las diferentes partes del programa (estas partes se explicaran en breve), la cual es la siguiente:



```
1 options { Ignore_Case = false; } //Apartado option
2 PARSER_BEGIN (Practical) //Inicio del Parser
3 public class Practical {
4     public static void main (String[] argum) throws ParseException {
5         Practical anLexSint = new Practical (System.in);
6         anLexSint.unaExpresion(); //Unidad de procesamiento
7         System.out.println("Análisis terminado:");
8     }
9 }
10 PARSER_END (Practical) //Fin del Parser
11
12
13 TOKEN:
14 {
15     <OPSUM> "+" {System.out.println("Se detecto el operador SUMA");} |
16     <OPRES> "-" {System.out.println("Se detecto el operador RESTA");} |
17     <OPMUL> "*" {System.out.println("Se detecto el operador MULTIPLICACION");} |
18     <OPDIV> "/" {System.out.println("Se detecto el operador DIVISION");} |
19     <OPDMO> "%" {System.out.println("Se detecto el operador DIVISION MODULAR");} | //Tokens
20     <OPPOT> "^" {System.out.println("Se detecto el operador POTENCIA");} |
21     <LEMAY> "[A-Z]" {System.out.println("Se detecto una letra MAYUSCULA");} |
22     <PARAB> "(" {System.out.println("Se detecto un PARENTESIS QUE ABRE");} |
23     <PARCI> ")" {System.out.println("Se detecto un PARENTESIS QUE CIERRA");} |
24     <DECI> "[0-9]" {System.out.println("Se detecto un DECIMAL");} |
25     <HEXA> "[0-9a-fA-z]" {System.out.println("Se detecto un HEXADECIMAL");}
26 }
27 SKIP: { " " | "\t" | "\n" | "\r" } //Token SKIP
28 void unaExpresion() { }
29 {
30     (<OPSUM> | <OPRES> | <OPMUL> | <OPDIV> | <OPDMO> | <OPPOT> | <LEMAY> | <PARAB> | <PARCI> | <DECI> | <HEXA>)+ <EOF> //Métodos
31 }
```

Figura 1. Captura de pantalla del programa de la Práctica 1.

Apartado option: En esta sección se encuentran ciertas “opciones” las cuales pueden modificar el funcionamiento ya definido del programa, como por ejemplo en este caso, donde se especifica que la opción *Ignore_Case* este desactivada (ya que por defecto esta activada) para que pueda diferenciar entre las letras minúsculas y mayúsculas.

Inicio y fin del parser: Sección obligatoria para el programa; delimita la unidad de compilación dentro del mismo.

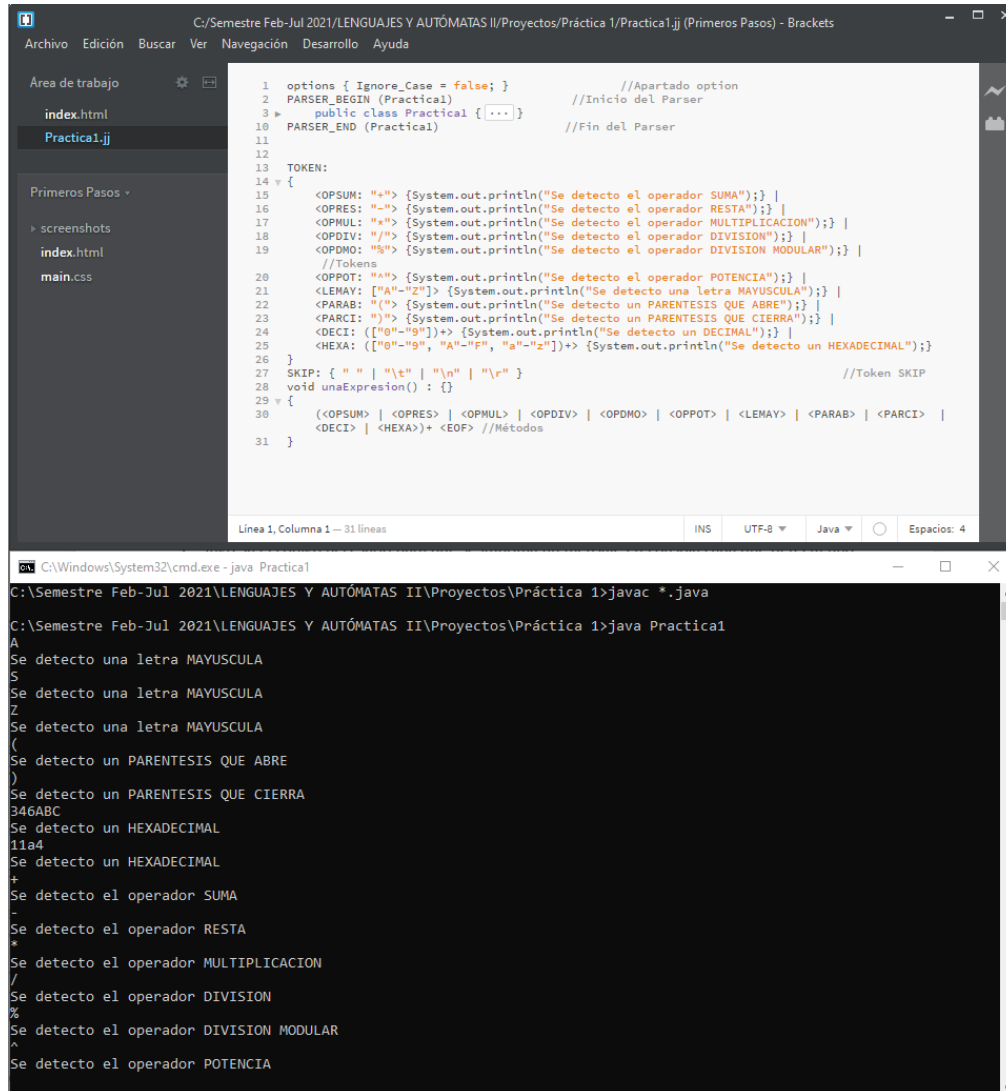
Unidad de compilación: Esta escrita en lenguaje Java. Es la parte que contiene la clase principal junto con su excepción *ParseException* y da inicio al programa. En este caso se crea un objeto de la clase *Practica1* para poder leer distintas cadenas y caracteres.

Reglas de producción: En esta sección se encuentran los tokens, tokens especiales, métodos y la etiqueta <EOF> (End-Of-Line).

En la parte de tokens se declararon los operadores aritméticos, letras mayúsculas, paréntesis, decimales y hexadecimales. En esta parte se agregó el token *SKIP* para que omitiera saltos de línea, tabulaciones, espacios y retornos. Y, por último, se agregó el método *unaExpresion*, la cual recibe distintos caracteres (mínimo 1) y cadenas de manera indefinida.

Conclusiones

El programa opero como se esperaba, como se muestra en la siguiente imagen.



The image shows a screenshot of the Brackets IDE interface. The top window displays the source code for 'Practical1.jj', which is a JavaCC grammar file. The code defines a parser for a language with various tokens and operators. The bottom window shows the output of the JavaCC compiler and the execution of the generated parser. The output lists the detected tokens and operators for the input string 'A S Z (346ABC 11a4 + * / % ^'.

```
1 options { Ignore_Case = false; } //Apartado option
2 PARSER_BEGIN (Practical) //Inicio del Parser
3 public class Practical { ... }
10 PARSER_END (Practical) //Fin del Parser
11
12
13 TOKEN:
14 {
15 <OPSUM: "+"> {System.out.println("Se detecto el operador SUMA");} |
16 <OPRES: "-"> {System.out.println("Se detecto el operador RESTA");} |
17 <OPMUL: "*"> {System.out.println("Se detecto el operador MULTIPLICACION");} |
18 <OPDIV: "/"> {System.out.println("Se detecto el operador DIVISION");} |
19 <OPDMO: "%"> {System.out.println("Se detecto el operador DIVISION MODULAR");} |
20 //Tokens
21 <OPPOT: "^"> {System.out.println("Se detecto el operador POTENCIA");} |
22 <LEMAY: "[A-Z]"> {System.out.println("Se detecto una letra MAYUSCULA");} |
23 <PARAB: "("> {System.out.println("Se detecto un PARENTESIS QUE ABRE");} |
24 <PARCI: ")"> {System.out.println("Se detecto un PARENTESIS QUE CIERRA");} |
25 <DECI: "[0-9]"> {System.out.println("Se detecto un DECIMAL");} |
26 <HEXA: "[0-9a-fA-F]"> {System.out.println("Se detecto un HEXADECIMAL");}
27 }
28 SKIP: { " " | "\t" | "\n" | "\r" } //Token SKIP
29 void unaExpresion() : {}
30 {
31 {<OPSUM> | <OPRES> | <OPMUL> | <OPDIV> | <OPDMO> | <OPPOT> | <LEMAY> | <PARAB> | <PARCI> |
32 <DECI> | <HEXA>}+ <EOF> //Métodos
33 }
```

```
C:\Windows\System32\cmd.exe - java Practical1
C:\Semestre Feb-Jul 2021\LENGUAJES Y AUTÓMATAS II\Proyectos\Práctica 1>javac *.java
C:\Semestre Feb-Jul 2021\LENGUAJES Y AUTÓMATAS II\Proyectos\Práctica 1>java Practical1
A
S
Z
(
Se detecto un PARENTESIS QUE ABRE
)
Se detecto un PARENTESIS QUE CIERRA
346ABC
Se detecto un HEXADECIMAL
11a4
Se detecto un HEXADECIMAL
+
Se detecto el operador SUMA
*
Se detecto el operador RESTA
/
Se detecto el operador MULTIPLICACION
%
Se detecto el operador DIVISION
^
Se detecto el operador DIVISION MODULAR
Se detecto el operador POTENCIA
```

Figura 2. Resultados en el CMD de la Práctica 1.

Durante las últimas clases se dio un repaso general de las expresiones regulares extendidas, vistas anteriormente en la materia de Lenguajes y Autómatas I; conceptos que serán esenciales a lo largo de la presente materia

También durante esta sencilla práctica se comprendió como operar con código Java dentro de los tokens, al igual que se repasaron las reglas básicas para el compilado y ejecución de los programas de JavaCC.