

CASTRO CRUCES JORGE E.	INSTITUTO POLITÉCNICO NACIONAL	ESCUELA SUPERIOR DE CÓMPUTO
SEGUNDO PARCIAL	PRÁCTICA 4: ANÁLISIS SINTÁCTICO DE DESCENSO RECURSIVO	04/05/2021
3CV18	COMPILADORES	Hernández Olvera Luis E

## SECCIÓN TEÓRICA O INTRODUCCIÓN

### ANÁLISIS SINTÁCTICO DE DESCENSO RECURSIVO

Consiste en un conjunto de procedimientos, uno para cada no terminal. El descenso recursivo general puede requerir un rastreo hacia atrás; es decir, tal vez requiere exploraciones repetidas sobre la entrada.

#### Análisis Sintáctico Predictivo por Descenso Recursivo

Sea una gramática G cualquiera no ambigua como la mostrada a continuación:

expresión  $\rightarrow$  expresión + término  
                   | expresión - término  
                   | término

término  $\rightarrow$  término \* factor  
                   | término / factor  
                   | factor

factor  $\rightarrow$  núm  
                   | id  
                   | - expresión  
                   | ( expresión )

Para obtener el analizador sintáctico predictivo se siguen los siguientes pasos.

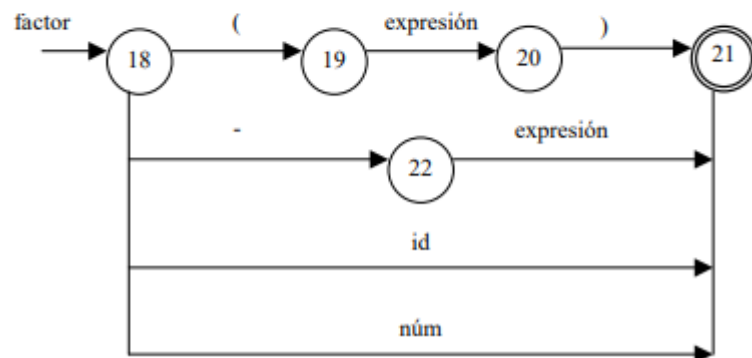
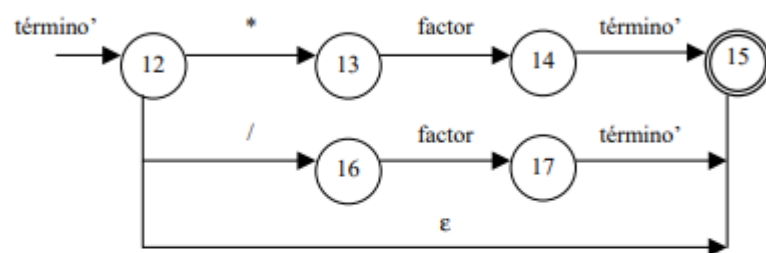
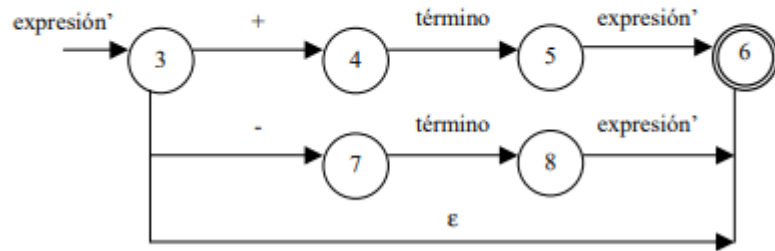
Primero: se elimina la recursión por la izquierda.

expresión  $\rightarrow$  término expresión'  
 expresión'  $\rightarrow$  + término expresión'  
                   | - término expresión'  
                   |  $\epsilon$   
 término  $\rightarrow$  factor término'  
 término'  $\rightarrow$  \* factor término'  
                   | / factor término'  
                   |  $\epsilon$   
 factor  $\rightarrow$  núm  
                   | id  
                   | - expresión  
                   | ( expresión )

Segundo: se factoriza por la izquierda. (la gramática mostrada no tiene reglas factorizables).

Tercero: se crea un diagrama de transiciones sintáctico para cada no terminal A mediante las siguientes reglas

- Se crea un estado de inicio y un estado final.
- Se crea un camino para cada producción  $A \rightarrow X_1 X_2 \dots X_n$  del no terminal, desde el estado de inicio al estado final, con aristas etiquetadas  $X_1, X_2, \dots, X_n$ .



*Diagramas de transiciones sintáctico para cada no terminal A*

Cuarto: se implementa cada diagrama como una rutina en algún lenguaje de programación.

```
Procedimiento expresión( )
Inicio
    Llamar_A término( )
    Llamar_A expresión'( )
Fin

Procedimiento expresión'( )
Inicio
    Llamar A obtenerSímbolo( )
    Si símbolo = '+' Entonces
        Llamar_A término( )
        Llamar_A expresión'( )
    Sino Si símbolo = '-' Entonces
        Llamar_A término( )
        Llamar_A expresión'( )
    Sino
        Llamar_A devolverSímbolo( )
Fin

Procedimiento término( )
Inicio
    Llamar_A factor( )
    Llamar_A término'( )
Fin

Procedimiento término'( )
Inicio
    Llamar A obtenerSímbolo( )
    Si símbolo = '*' Entonces
        Llamar_A factor( )
        Llamar_A término'( )
    Sino Si símbolo = '/' Entonces
        Llamar_A factor( )
        Llamar_A término'( )
    Sino
        Llamar_A devolverSímbolo( )
Fin

Procedimiento factor( )
Inicio
    Llamar A obtenerSímbolo( )
    Si símbolo = '(' Entonces
        Llamar_A expresión( )
        símbolo = obtenerSímbolo( )
        Si símbolo = ')' Entonces
            Fin_Procedimiento
        Sino
            escribir( "Error: se esperaba un paréntesis." )
        Fin_Si
    Sino Si símbolo = '-' Entonces
        Llamar_A expresión( )
    Sino Si símbolo = id Entonces
        Fin_Procedimiento
    Sino Si símbolo = núm Entonces
        Fin_Procedimiento
    Sino
        escribir( "Error: entrada incorrecta." )
        Fin_Programa
    Fin_Si
Fin
```

*Pseudocódigo de cada uno de los diagramas anteriores en forma de rutina*

Estas rutinas son apoyadas por otros procedimientos y declaraciones de ámbito global. Existe una variable llamada *símbolo* de tipo entero que mantiene el último símbolo reconocido por el analizador léxico. Existe también una variable llamada *símbolo\_devuelto* que actúa como una *bandera* que indica si el analizador sintáctico desechó el último símbolo leído. Para graficar esta situación considere el caso de la rutina *expresión()*, si el símbolo leído fue un '+' o un '-' entonces se ejecuta la acción, de lo contrario se simula que el símbolo fue devuelto a la entrada cambiando el valor por defecto de esta variable. Con esta estrategia la siguiente llamada a *obtenerSímbolo()* regresará el mismo símbolo que la vez anterior.

```
símbolo = 0
```

```
símbolo_devuelto = FALSO
```

```
Procedimiento obtenerSímbolo( )
Inicio
    Si símbolo_devuelto = FALSO Entonces
        Símbolo = funciónDeAnálisisLéxico( )
        Regresar símbolo
    Fin Si
    Símbolo_devuelto = FALSO
    Regresar símbolo
Fin

Procedimiento devolverSímbolo( )
Inicio
    Símbolo_devuelto = CIERTO
Fin
```

La función *DeAnálisisLéxico()* es en propiedad el analizador léxico, por lo tanto su nombre depende de cual haya sido la estrategia que utilizó para construirlo. Por ejemplo, en el caso del apunte "Diagramas de Transiciones" esta función toma el nombre *siguienteSímbolo()*. En el caso del apunte "Lenguaje Lex", la función toma el nombre *yylex()*, el cual es asignado automáticamente por el generador y cuyo código se puede observar dentro del archivo *lex.yy.c*. También podría tratarse de una función del usuario escrita sin ninguna metodología como las mencionadas.

El origen de la denominación del nombre *predictivo*, de este analizador sintáctico, se debe al hecho que basta con leer un sólo símbolo de la entrada para reconocer la producción que se debe expandir. El origen de la denominación *recursivo* se debe a la naturaleza inherentemente recursiva de los lenguajes de programación, la cual debería quedar de manifiesto en el ejemplo simple que se ha mostrado.

Si analiza en detalle el funcionamiento de este código notará que en todo momento se lee desde la entrada el símbolo de más a la izquierda y que las reglas se expanden siempre desde los no terminales que están más a la izquierda. Este funcionamiento podría graficarse como una construcción del árbol de análisis sintáctico desde la raíz hacia las hojas.

#### PROBLEMAS EN EL ANALISIS DESCENDENTE:

- La recursividad a izquierdas da lugar a un bucle infinito de recursión.
- Problemas de indeterminismo cuando varias alternativas en una misma producción comparten el mismo prefijo. No sabríamos cual elegir. Probaríamos una y si se produce un error haríamos back tracking. Si queremos que el método sea determinista hay que evitarlas.

Existen transformaciones de gramáticas para su la eliminación. Estas transformaciones no modifican el lenguaje que se está reconociendo, pero sí que cambian el aspecto de la gramática, que es en general menos intuitiva.

## **PROBLEMAS RELACIONADOS AL TEMA A LOS CUALES TE ENFRENTASTE AL PROGRAMAR LA PRÁCTICA**

Para esta práctica, el principal problema al que me enfrenté fue aprender y comprender un nuevo lenguaje; Porque, a pesar de que Java es un lenguaje de programación orientado a objetos sumamente popular y robusto, yo nunca había tenido el placer de ocuparlo.

Otra gran problemática a la que me enfrenté fue: La programación recursiva, y es que a lo largo de mi estancia en la ESCOM, son contadas las veces que he tenido la necesidad de programar utilizando la programación recursiva.

Dicho lo anterior, es acertado pensar que me costó trabajo llevar a cabo esta práctica y que a pesar de las dificultades, logré sacar adelante la tarea y eso me demuestra a mi mismo la calidad y la determinación que tengo como persona y como estudiante.

## CAPTURAS DE PANTALLA

El objetivo de la práctica es programar un Analizador Sintáctico de Descenso Recursivo que permita reconocer n sumas y restas de n elementos:

$E \rightarrow TZ$

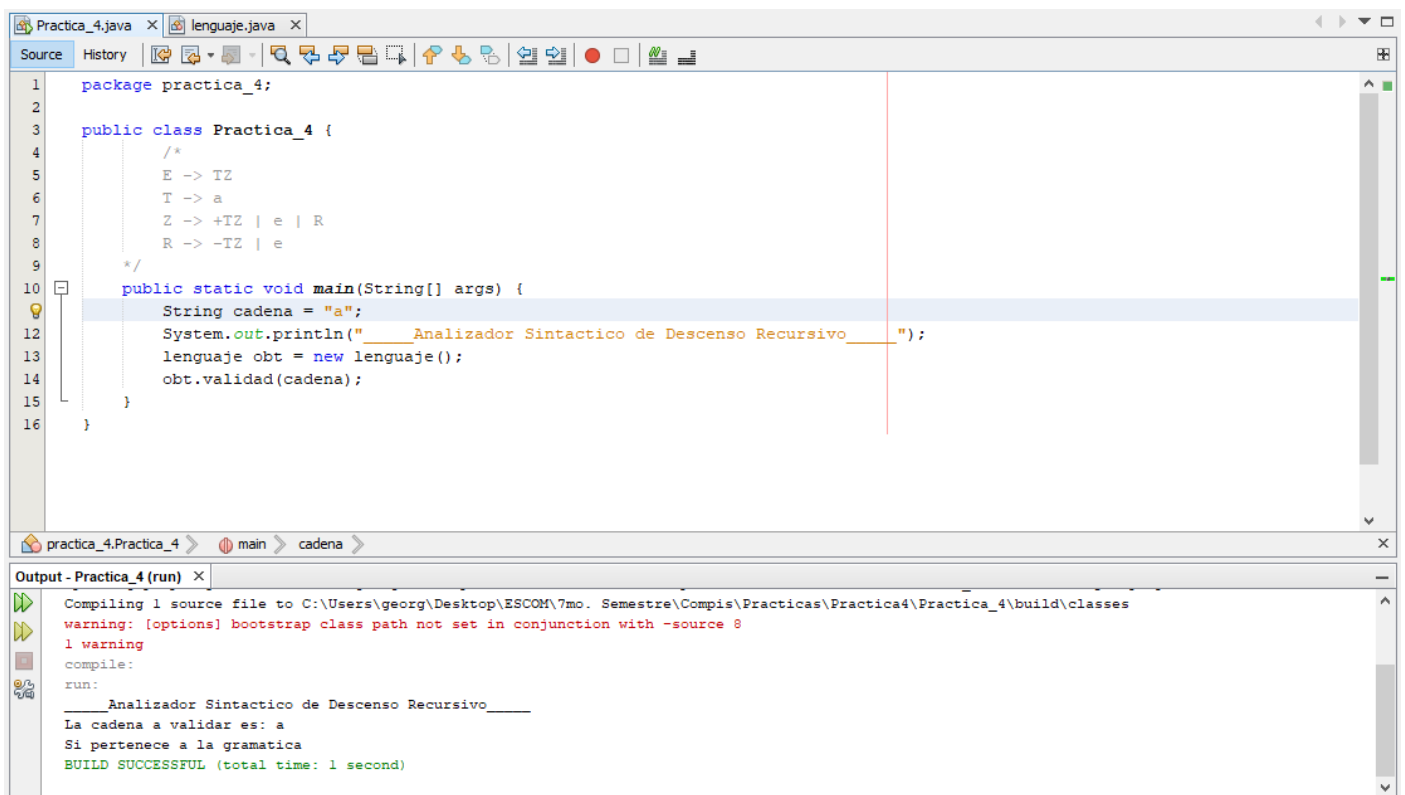
$T \rightarrow a$

$Z \rightarrow +TZ \mid e \mid R$

$R \rightarrow -TZ \mid e$

Ahora pasamos a la ejecución del programa con las siguientes cadenas:

**a**



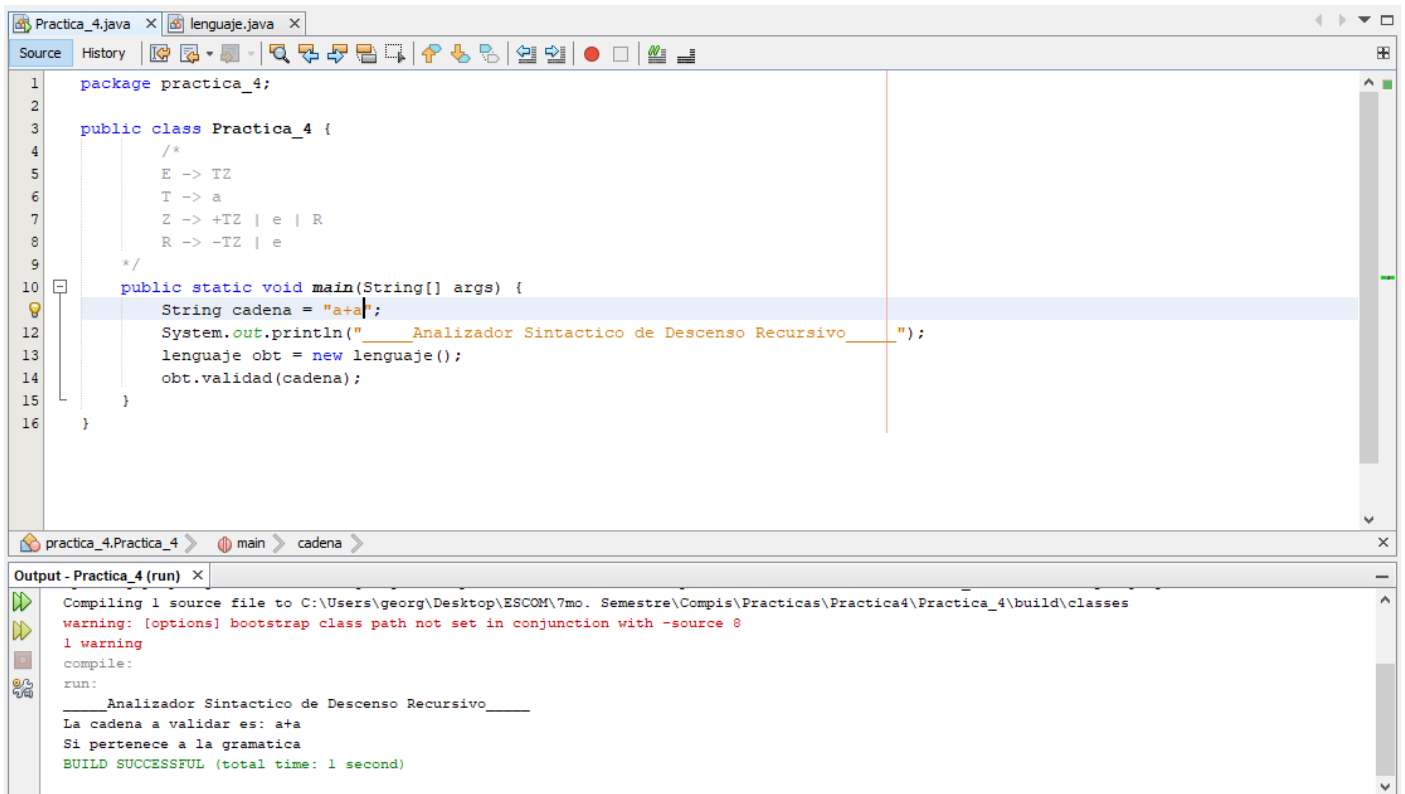
The screenshot shows an IDE with two tabs: 'Practica\_4.java' and 'lenguaje.java'. The 'Practica\_4.java' tab is active, displaying the following code:

```
1 package practica_4;
2
3 public class Practica_4 {
4     /*
5      * E -> TZ
6      * T -> a
7      * Z -> +TZ | e | R
8      * R -> -TZ | e
9      */
10    public static void main(String[] args) {
11        String cadena = "a";
12        System.out.println("____ Analizador Sintactico de Descenso Recursivo ____");
13        lenguaje obt = new lenguaje();
14        obt.validad(cadena);
15    }
16 }
```

The 'lenguaje.java' tab is also visible, showing the implementation of the parser. The output window at the bottom shows the following text:

```
Output - Practica_4 (run) X
Compiling 1 source file to C:\Users\georg\Desktop\ESCOM\7mo. Semestre\Compis\Practicas\Practica4\Practica_4\build\classes
warning: [options] bootstrap class path not set in conjunction with -source 8
1 warning
compile:
run:
____ Analizador Sintactico de Descenso Recursivo ____
La cadena a validar es: a
Si pertenece a la gramatica
BUILD SUCCESSFUL (total time: 1 second)
```

# a+a



```
1 package practica_4;
2
3 public class Practica_4 {
4     /*
5      E -> TZ
6      T -> a
7      Z -> +TZ | e | R
8      R -> -TZ | e
9     */
10    public static void main(String[] args) {
11        String cadena = "a+a";
12        System.out.println("____ Analizador Sintactico de Descenso Recursivo ____");
13        lenguaje obt = new lenguaje();
14        obt.validad(cadena);
15    }
16 }
```

practica\_4.Practica\_4 > main > cadena >

Output - Practica\_4 (run) X

```
Compiling 1 source file to C:\Users\georg\Desktop\ESCOM\7mo. Semestre\Compis\Practicas\Practica4\Practica_4\build\classes
warning: [options] bootstrap class path not set in conjunction with -source 8
1 warning
compile:
run:
____ Analizador Sintactico de Descenso Recursivo ____
La cadena a validar es: a+a
Si pertenece a la gramatica
BUILD SUCCESSFUL (total time: 1 second)
```

a+a+a

```
1 package practica_4;
2
3 public class Practica_4 {
4     /*
5     E -> TZ
6     T -> a
7     Z -> +TZ | e | R
8     R -> -TZ | e
9     */
10    public static void main(String[] args) {
11        String cadena = "a+a+a";
12        System.out.println("____Analizador Sintactico de Descenso Recursivo____");
13        lenguaje obt = new lenguaje();
14        obt.validad(cadena);
15    }
16 }
```

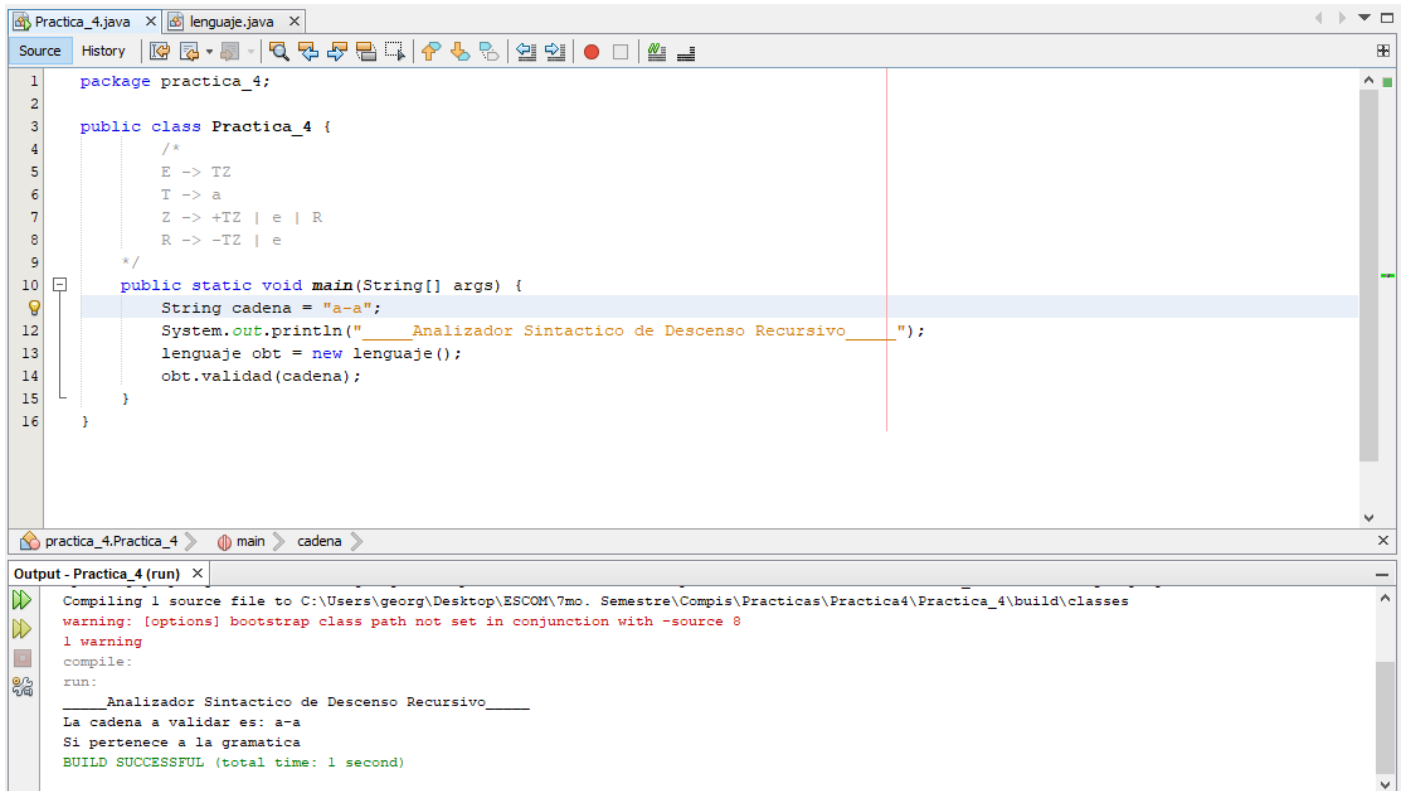
practica\_4.Practica\_4 > main > cadena >

Output - Practica\_4 (run) X

Compiling 1 source file to C:\Users\georg\Desktop\ESCOM\7mo. Semestre\Compis\Practicas\Practica4\Practica\_4\build\classes  
warning: [options] bootstrap class path not set in conjunction with -source 8  
1 warning  
compile:  
run:  
\_\_\_\_Analizador Sintactico de Descenso Recursivo\_\_\_\_  
La cadena a validar es: a+a+a  
Si pertenece a la gramatica  
BUILD SUCCESSFUL (total time: 1 second)



a-a



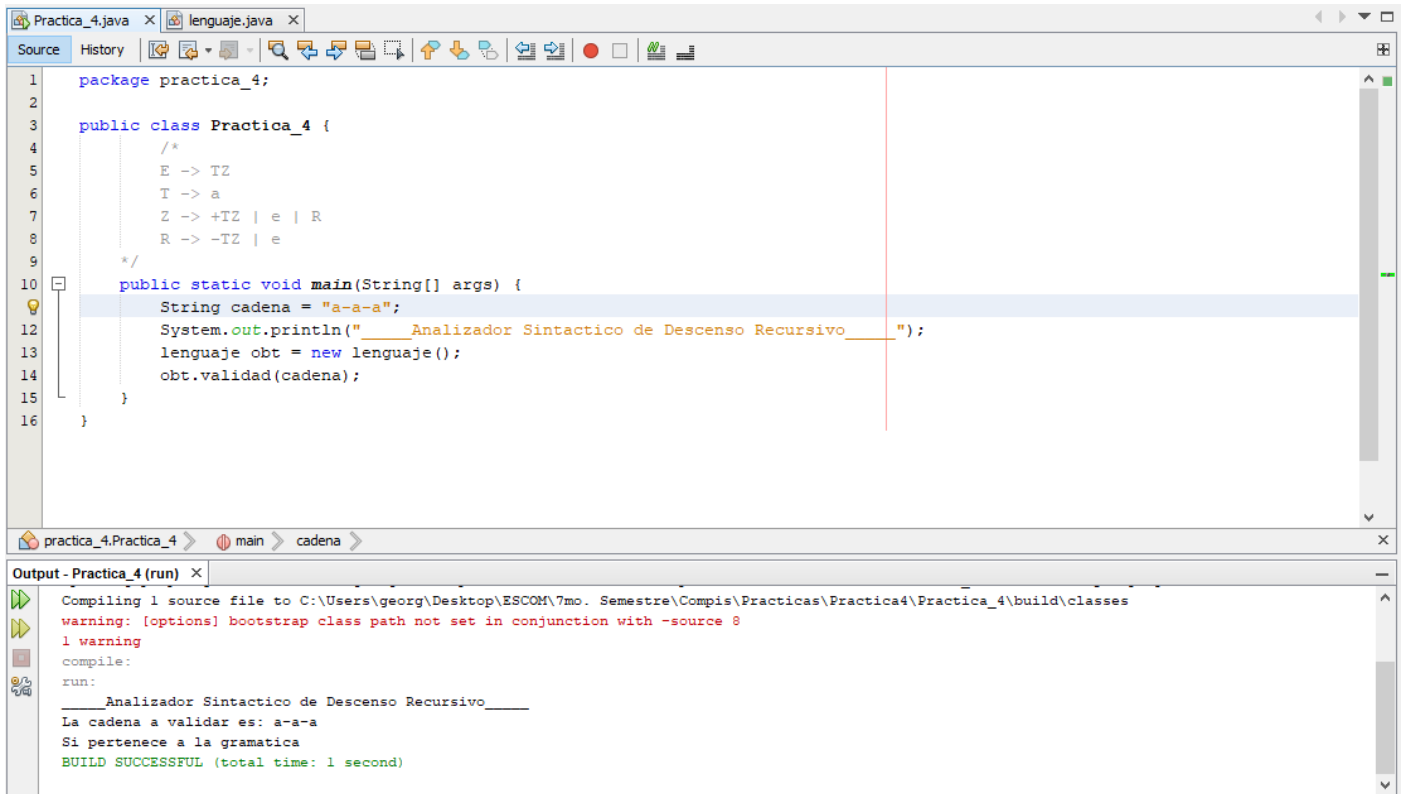
The screenshot shows an IDE with two tabs: 'Practica\_4.java' and 'lenguaje.java'. The 'Practica\_4.java' tab is active, displaying the following Java code:

```
1 package practica_4;
2
3 public class Practica_4 {
4     /*
5      * E -> TZ
6      * T -> a
7      * Z -> +TZ | e | R
8      * R -> -TZ | e
9      */
10    public static void main(String[] args) {
11        String cadena = "a-a";
12        System.out.println("____Analizador Sintactico de Descenso Recursivo____");
13        lenguaje obt = new lenguaje();
14        obt.validad(cadena);
15    }
16 }
```

The 'lenguaje.java' tab is also visible, showing the implementation of the 'lenguaje' class. The 'Output - Practica\_4 (run)' window at the bottom shows the following output:

```
Compiling 1 source file to C:\Users\georg\Desktop\ESCOM\7mo. Semestre\Compis\Practicas\Practica4\Practica_4\build\classes
warning: [options] bootstrap class path not set in conjunction with -source 8
1 warning
compile:
run:
____Analizador Sintactico de Descenso Recursivo____
La cadena a validar es: a-a
Si pertenece a la gramatica
BUILD SUCCESSFUL (total time: 1 second)
```

a-a-a



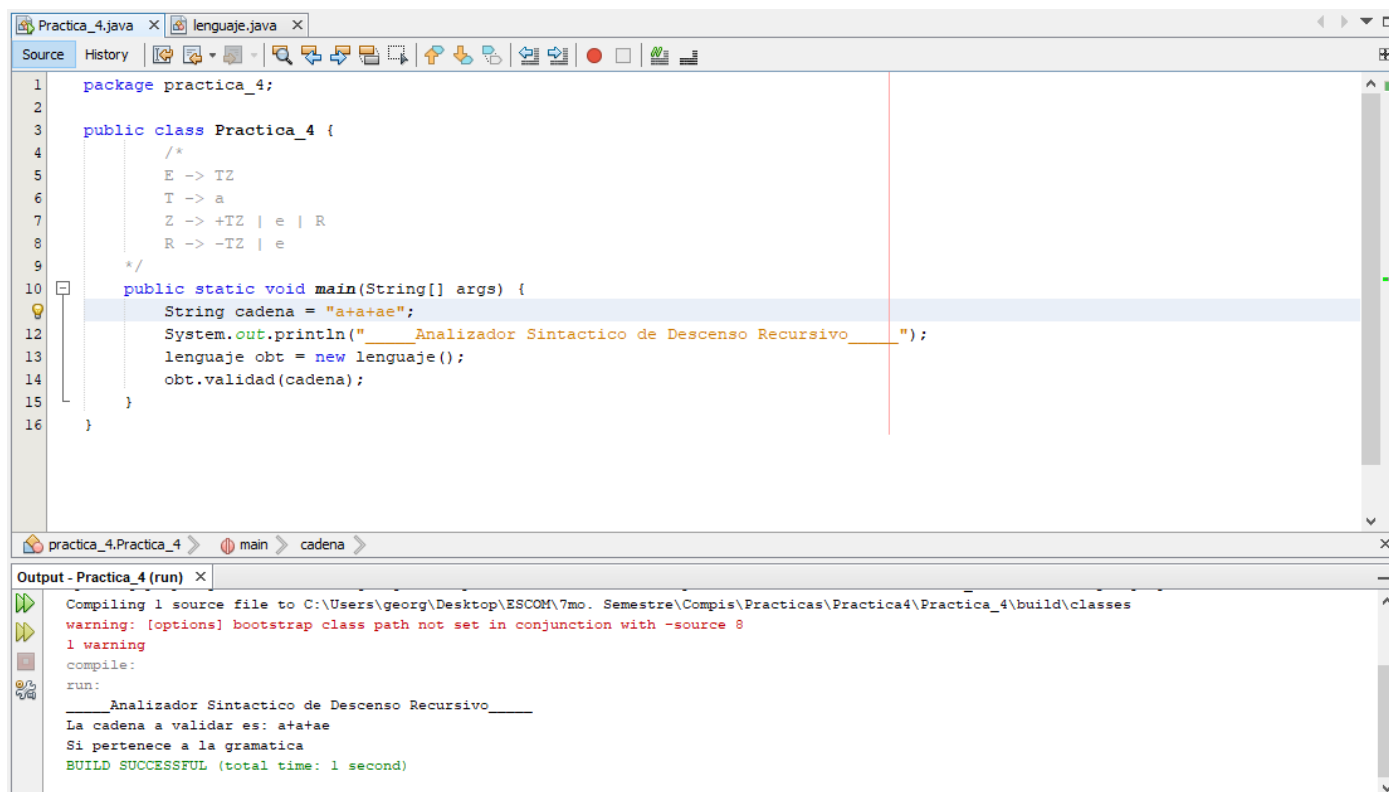
The screenshot shows an IDE with two tabs: 'Practica\_4.java' and 'lenguaje.java'. The 'Practica\_4.java' tab is active, displaying the following Java code:

```
1 package practica_4;
2
3 public class Practica_4 {
4     /*
5      * E -> TZ
6      * T -> a
7      * Z -> +TZ | e | R
8      * R -> -TZ | e
9      */
10    public static void main(String[] args) {
11        String cadena = "a-a-a";
12        System.out.println("____Analizador Sintactico de Descenso Recursivo____");
13        lenguaje obt = new lenguaje();
14        obt.validad(cadena);
15    }
16 }
```

The 'lenguaje.java' tab is also visible, showing the grammar rules for the parser. The output window, titled 'Output - Practica\_4 (run)', shows the following text:

```
Compiling 1 source file to C:\Users\georg\Desktop\ESCOM\7mo. Semestre\Compis\Practicas\Practica4\Practica_4\build\classes
warning: [options] bootstrap class path not set in conjunction with -source 8
1 warning
compile:
run:
____Analizador Sintactico de Descenso Recursivo____
La cadena a validar es: a-a-a
Si pertenece a la gramatica
BUILD SUCCESSFUL (total time: 1 second)
```

**a+a+ae**



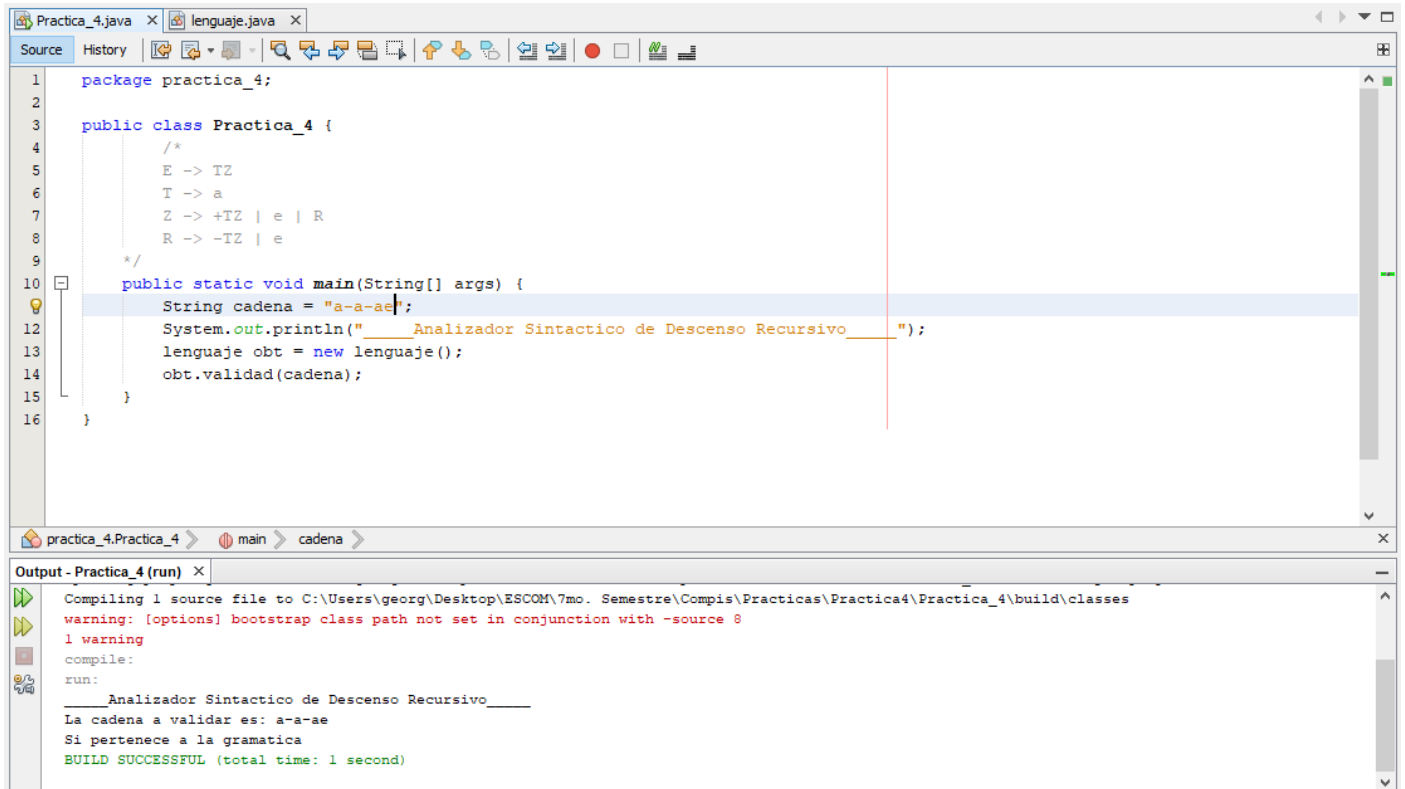
The screenshot shows an IDE with two tabs: 'Practica\_4.java' and 'lenguaje.java'. The 'Practica\_4.java' tab is active, displaying the following Java code:

```
1 package practica_4;
2
3 public class Practica_4 {
4     /*
5      * E -> TZ
6      * T -> a
7      * Z -> +TZ | e | R
8      * R -> -TZ | e
9      */
10    public static void main(String[] args) {
11        String cadena = "a+a+ae";
12        System.out.println("____Analizador Sintactico de Descenso Recursivo____");
13        lenguaje obt = new lenguaje();
14        obt.validad(cadena);
15    }
16 }
```

The bottom of the IDE shows the 'Output - Practica\_4 (run)' window with the following text:

```
Compiling 1 source file to C:\Users\georg\Desktop\ESCOH\7mo. Semestre\Compis\Practicas\Practica4\Practica_4\build\classes
warning: [options] bootstrap class path not set in conjunction with -source 8
1 warning
compile:
run:
____Analizador Sintactico de Descenso Recursivo____
La cadena a validar es: a+a+ae
Si pertenece a la gramatica
BUILD SUCCESSFUL (total time: 1 second)
```

# a-a-ae



The screenshot shows an IDE with two tabs: 'Practica\_4.java' and 'lenguaje.java'. The 'Practica\_4.java' tab is active, displaying the following Java code:

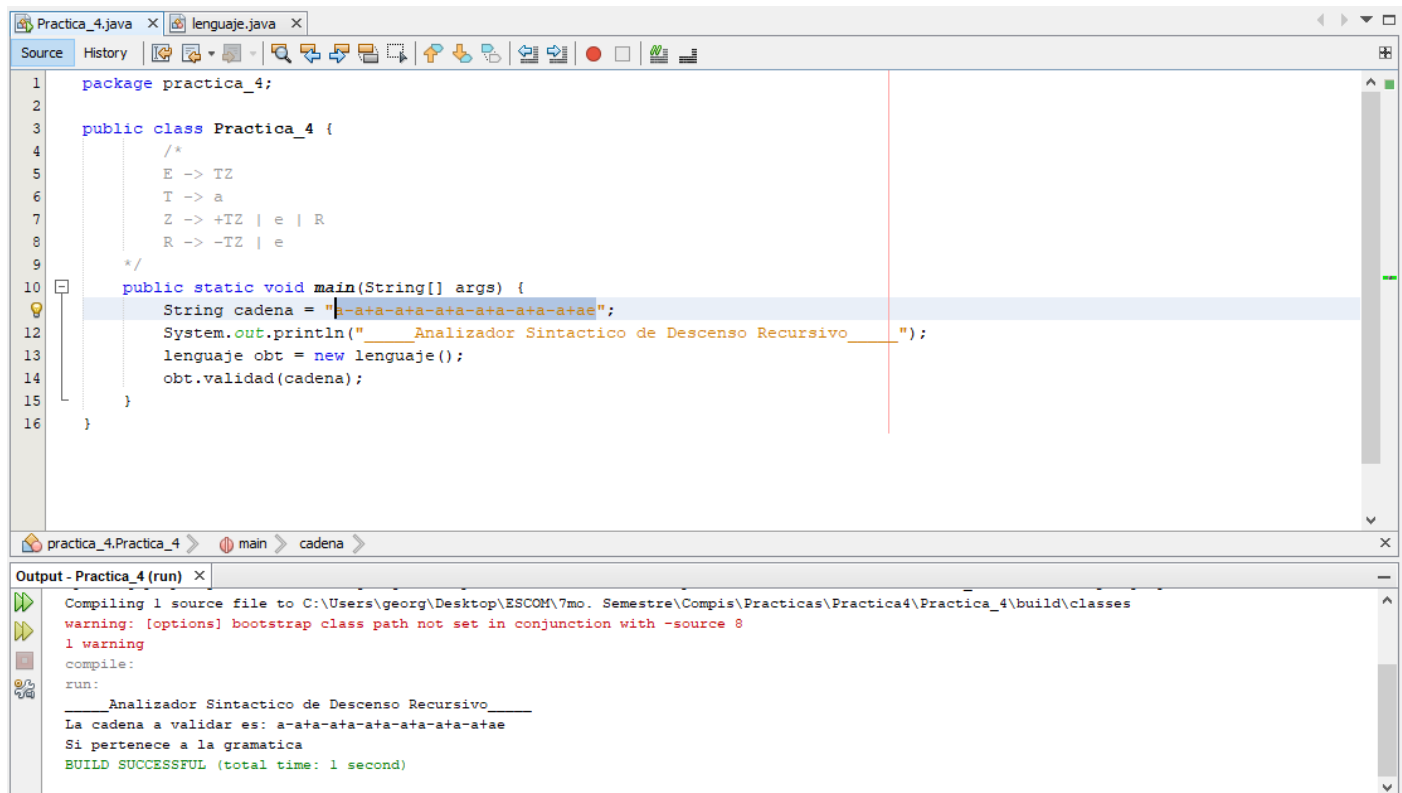
```
1 package practica_4;
2
3 public class Practica_4 {
4     /*
5     E -> TZ
6     T -> a
7     Z -> +TZ | e | R
8     R -> -TZ | e
9     */
10    public static void main(String[] args) {
11        String cadena = "a-a-ae";
12        System.out.println("____ Analizador Sintactico de Descenso Recursivo ____");
13        lenguaje obt = new lenguaje();
14        obt.validad(cadena);
15    }
16 }
```

The 'lenguaje.java' tab is also visible, showing the grammar rules for the language.

The output window, titled 'Output - Practica\_4 (run)', shows the following text:

```
Compiling 1 source file to C:\Users\georg\Desktop\ESCOM\7mo. Semestre\Compis\Practicas\Practica4\Practica_4\build\classes
warning: [options] bootstrap class path not set in conjunction with -source 8
1 warning
compile:
run:
____ Analizador Sintactico de Descenso Recursivo ____
La cadena a validar es: a-a-ae
Si pertenece a la gramatica
BUILD SUCCESSFUL (total time: 1 second)
```

**a-a+a-a+a-a+a-a+a-a+ae**



```
1 package practica_4;
2
3 public class Practica_4 {
4     /*
5      * E -> TZ
6      * T -> a
7      * Z -> +TZ | e | R
8      * R -> -TZ | e
9      */
10    public static void main(String[] args) {
11        String cadena = "a-a+a-a+a-a+a-a+ae";
12        System.out.println("____ Analizador Sintactico de Descenso Recursivo ____");
13        lenguaje obt = new lenguaje();
14        obt.validad(cadena);
15    }
16 }
```

practica\_4.Practica\_4 > main > cadena >

Output - Practica\_4 (run) X

Compiling 1 source file to C:\Users\georg\Desktop\ESCOM\7mo. Semestre\Compis\Practicas\Practica4\Practica\_4\build\classes  
warning: [options] bootstrap class path not set in conjunction with -source 8  
1 warning  
compile:  
run:  
\_\_\_\_ Analizador Sintactico de Descenso Recursivo \_\_\_\_  
La cadena a validar es: a-a+a-a+a-a+a-a+ae  
Si pertenece a la gramatica  
BUILD SUCCESSFUL (total time: 1 second)

## CONCLUSIÓN

### ¿Qué aprendí de esta práctica?

Claramente, aprendí muchas cosas, entre ellas está:

- Aprender a programar de forma orientada a objetos.
- Aprender a programar en Java.
- Aprender a programar de forma recursiva.
- Comprender de mejor forma las gramáticas.
- Aprender la diferencia entre un terminal y un no terminal

### ¿En qué me ayudó esta práctica?

Claramente, me ayudo a comprender de mejor forma la utilización y las ventajas de programar en Java y el gran poder que posee y nos otorga la programación orientada a objetos, así como, la programación recursiva

También, expandió mi conocimiento sobre la materia de Compiladores y el abanico de posibilidades que esta herramienta nos otorga.

## REFERENCIAS

- [1] BB, D., 2021. *Análisis sintáctico descendente recursivo*. [online] prezi.com. Available at: <https://prezi.com/tevrccpxr-w7/analisis-sintactico-descendente-recursivo/> [Accessed 4 May 2021].
- [2] Águila, J., 2004. *kataix.umag*. [online] Kataix.umag.cl. Available at: [https://kataix.umag.cl/~jaguila/Compilers/T10\\_AS\\_Desc\\_Recursivo.pdf](https://kataix.umag.cl/~jaguila/Compilers/T10_AS_Desc_Recursivo.pdf) [Accessed 4 May 2021].
- [3] Aho, A.V., Sethi, R., Ullman, J.D. (1990), *Compiladores: principios, técnicas y herramientas* ´, Tema 4, páginas: 186-200