

Procesadores de Lenguajes

Entrega 1: Completar el conjunto de instrucciones de Tinto



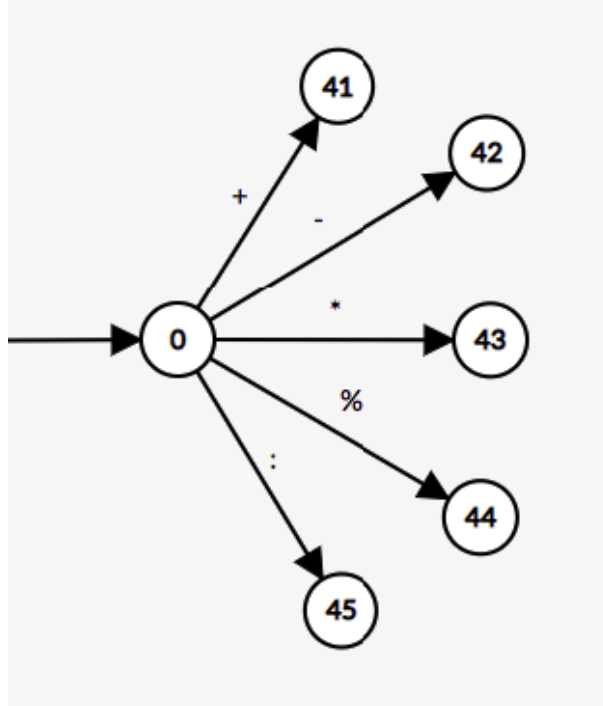
Francisco Jesús Beltrán Moreno

Índice

<i>Descripción de las modificaciones realizadas al autómata</i>	<i>3</i>
<i>Descripción de las modificaciones realizadas a las constantes que describen las categorías léxicas.....</i>	<i>5</i>
<i>Pruebas de funcionamiento</i>	<i>6</i>
<i>Documentación.....</i>	<i>7</i>

Descripción de las modificaciones realizadas al autómata

La única modificación del autómata que tendremos que realizar será la siguiente:



Como podemos apreciar, consiste en añadir el `:` como estado final. (Los nodos del 41 al 45 son finales, es decir, deberían tener doble círculo, pero el simulador usado no disponía de esa opción, es simplemente una limitación de la página, no una errata.)

Continuando con la explicación, tenemos que modificar ahora el fichero [TintoLexer.java](#), en el método `transition()` vamos a modificar lo que hemos mostrado en el grafo anterior:

```
protected int transition(int state, char symbol)
{
    switch(state)
    {
        case 0:
            if(symbol == '/') return 1;
            else if(symbol == ' ' || symbol == '\t') return 7;
            else if(symbol == '\r' || symbol == '\n') return 7;
            else if(symbol >= 'a' && symbol <= 'z') return 8;
            else if(symbol >= 'A' && symbol <= 'Z') return 8;
            else if(symbol == '.') return 8;
            else if(symbol >= '1' && symbol <= '9') return 9;
            else if(symbol == '0') return 10;
            else if(symbol == '\(') return 16;
            else if(symbol == '(') return 22;
            else if(symbol == ')') return 23;
            else if(symbol == '{') return 24;
            else if(symbol == '}') return 25;
            else if(symbol == '[') return 26;
            else if(symbol == ']') return 27;
            else if(symbol == ',') return 28;
            else if(symbol == '=') return 29;
            else if(symbol == '<') return 31;
            else if(symbol == '>') return 33;
            else if(symbol == '!') return 35;
            else if(symbol == '|') return 37;
            else if(symbol == '&') return 39;
            else if(symbol == '+') return 41;
            else if(symbol == '-') return 42;
            else if(symbol == '*') return 43;
            else if(symbol == '%') return 44;
            else if(symbol == ':') return 45;
            else return -1;
        case 1:
            if(symbol == '*') return 2;
            else if(symbol == '/') return 5;
            else return -1;
        case 2:
            if(symbol == '*') return 3;
            else return 2;
        case 3:
            if(symbol == '*') return 3;
            else if(symbol == '/') return 4;
```

Ahora para que el autómata reconozca el nuevo estado 45 como estado final tenemos que añadirlo al método **IsFinal()**, aumentando el rango de **state** de 44 a 45:

```
protected boolean isFinal(int state)
{
    if(state <=0 || state > 45) return false;
    switch(state)
    {
        case 2:
        case 3:
        case 5:
        case 12:
        case 14:
        case 16:
        case 17:
        case 19:
        case 20:
        case 21:
        case 37:
        case 39:
            return false;
        default:
            return true;
    }
}
```

Con esto conseguimos añadir a nuestro autómata el nuevo estado 45, ahora tenemos que modificar nuestro **getToken()** para que reconozca cuando introduzcamos las nuevas cadenas de caracteres (For, while, ...):

```
protected Token getToken(int state, String lexeme, int row, int column)
{
    switch(state)
    {
        case 1: return new Token(DIV, lexeme, row, column);
        case 4: return null;
        case 6: return null;
        case 7: return null;
        case 8: return new Token(getKind(lexeme), lexeme, row, column);
        case 9: return new Token(INTEGER_LITERAL, lexeme, row, column);
        case 10: return new Token(INTEGER_LITERAL, lexeme, row, column);
        case 11: return new Token(INTEGER_LITERAL, lexeme, row, column);
        case 13: return new Token(INTEGER_LITERAL, lexeme, row, column);
        case 15: return new Token(INTEGER_LITERAL, lexeme, row, column);
        case 18: return new Token(CHAR_LITERAL, lexeme, row, column);
        case 22: return new Token(LPAREN, lexeme, row, column);
        case 23: return new Token(RPAREN, lexeme, row, column);
        case 24: return new Token(LBRACE, lexeme, row, column);
        case 25: return new Token(RBRACE, lexeme, row, column);
        case 26: return new Token(SEMICOLON, lexeme, row, column);
        case 27: return new Token(COMMA, lexeme, row, column);
        case 28: return new Token(DOT, lexeme, row, column);
        case 29: return new Token(ASSIGN, lexeme, row, column);
        case 30: return new Token(EQ, lexeme, row, column);
        case 31: return new Token(LT, lexeme, row, column);
        case 32: return new Token(LE, lexeme, row, column);
        case 33: return new Token(GT, lexeme, row, column);
        case 34: return new Token(GE, lexeme, row, column);
        case 35: return new Token(NOT, lexeme, row, column);
        case 36: return new Token(NE, lexeme, row, column);
        case 38: return new Token(OR, lexeme, row, column);
        case 40: return new Token(AND, lexeme, row, column);
        case 41: return new Token(PLUS, lexeme, row, column);
        case 42: return new Token(MINUS, lexeme, row, column);
        case 43: return new Token(PROD, lexeme, row, column);
        case 44: return new Token(MOD, lexeme, row, column);
        case 45: return new Token(COLONS, lexeme, row, column);
        default: return null;
    }
}
```

Descripción de las modificaciones realizadas a las constantes que describen las categorías léxicas

Para ello será necesario modificar el fichero **TokenConstants.java**, en él añadiremos las nuevas palabras claves siguiendo los anteriores valores, tendríamos algo así:

```
// Modificaciones Entrega 1
/**
 * Palabra clave "for"
 */
public int FOR = 41;

/**
 * Palabra clave "do"
 */
public int DO = 42;

/**
 * Palabra clave "switch"
 */
public int SWITCH = 43;

/**
 * Palabra clave "case"
 */
public int CASE = 44;

/**
 * Palabra clave "default"
 */
public int DEFAULT = 45;

/**
 * Palabra clave "break"
 */
public int BREAK = 46;

/**
 * Palabra clave "continue"
 */
public int CONTINUE = 47;
```

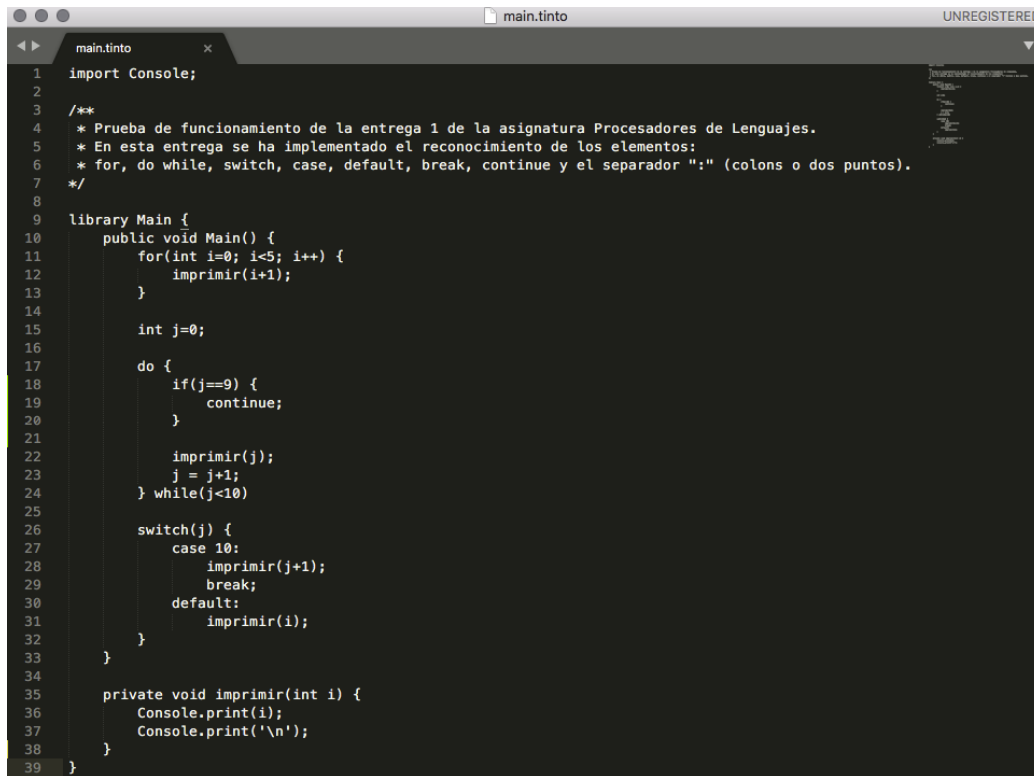
Palabras Clave

```
// Modificacion
/*
 * Dos puntos ":"
 */
public int COLONS = 48;
```

Separadores

Pruebas de funcionamiento

Para comprobar si nuestras modificaciones son correctas, se va a utilizar el siguiente código donde integraremos los nuevos elementos añadidos, para ello recompilaremos el archivo **tinto.jar** y lo ejecutaremos junto al siguiente archivo de código:



```
1 import Console;
2
3 /**
4  * Prueba de funcionamiento de la entrega 1 de la asignatura Procesadores de Lenguajes.
5  * En esta entrega se ha implementado el reconocimiento de los elementos:
6  * for, do while, switch, case, default, break, continue y el separador ":" (colons o dos puntos).
7  */
8
9 library Main {
10     public void Main() {
11         for(int i=0; i<5; i++) {
12             imprimir(i+1);
13         }
14
15         int j=0;
16
17         do {
18             if(j==9) {
19                 continue;
20             }
21
22             imprimir(j);
23             j = j+1;
24         } while(j<10)
25
26         switch(j) {
27             case 10:
28                 imprimir(j+1);
29                 break;
30             default:
31                 imprimir(i);
32         }
33     }
34
35     private void imprimir(int i) {
36         Console.print(i);
37         Console.print('\n');
38     }
39 }
```

En el fichero de salida **TintocOutput.txt**, podemos observar la siguiente salida:

- [Row: 10][Column: 2][Kind: 41] for
- [Row: 16][Column: 2][Kind: 42] do
- [Row: 18][Column: 4][Kind: 47] continue
- [Row: 25][Column: 2][Kind: 43] switch
- [Row: 26][Column: 3][Kind: 44] case
- [Row: 28][Column: 4][Kind: 46] break
- [Row: 29][Column: 3][Kind: 45] default
- [Row: 29][Column: 10][Kind: 48] :

Como podemos apreciar ha reconocido todo correctamente siguiendo los identificadores asignados, por lo que podemos dar por concluido esta práctica.

Documentación

Página usada para realizar el grafo: https://csacademy.com/app/graph_editor/