UNIVERSIDAD NEBRIJA

Escuela Politécnica Superior

Procesadores de lenguaje

Práctica 2

Modifique el analizador léxico desarrollado en la práctica 1. Esta nueva versión debe utilizar un fichero de configuración que defina los componentes léxicos del lenguaje.

El analizador léxico mejorado cumplir los siguientes requisitos:

- Analizar un programa almacenado en una cadena de caracteres o en un fichero de texto.
- Reconocer las palabras reservadas, los operadores y los caracteres delimitadores del lenguaje. Utilice una tabla Hash para almacenar el fichero `lexico.txt`, que define los componentes léxicos del lenguaje y sus lexemas. Las líneas de comentarios de este fichero comienzan por el carácter '/' y deben ser ignoradas. Las líneas no vacías definen un componente léxico y su lexema, separados por tabuladores.

El fichero de configuración `lexico.txt`.

```
// El fichero lexico.txt define los componentes léxicos del lenguaje,
// definidos por parejas de componente léxicos y lexemas


// Operadores relacionales y de asignación

greater_than                    >
greater_equals                  >=
less_than                       <
less_equals                     <=
equals                          ==
not_equals                      !=
assignment                      =

// Operadores aritméticos

add                             +
subtract                        -
multiply                        *
divide                          /
remainder                       %

// Operadores lógicos

and                             &&
or                              ||
not                             !

// Delimitadores

dot                             .
comma                           ,
semicolon                       ;
open_parenthesis                (
closed_parenthesis              )
open_square_bracket             [
closed_square_bracket           ]
open_bracket                    {
closed_bracket                  }
```

```
// Palabras reservadas

boolean                    boolean
break                      break
do                         do
else                       else
false                      false
float                      float
for                        for
if                         if
int                        int
while                      while
true                       true
```

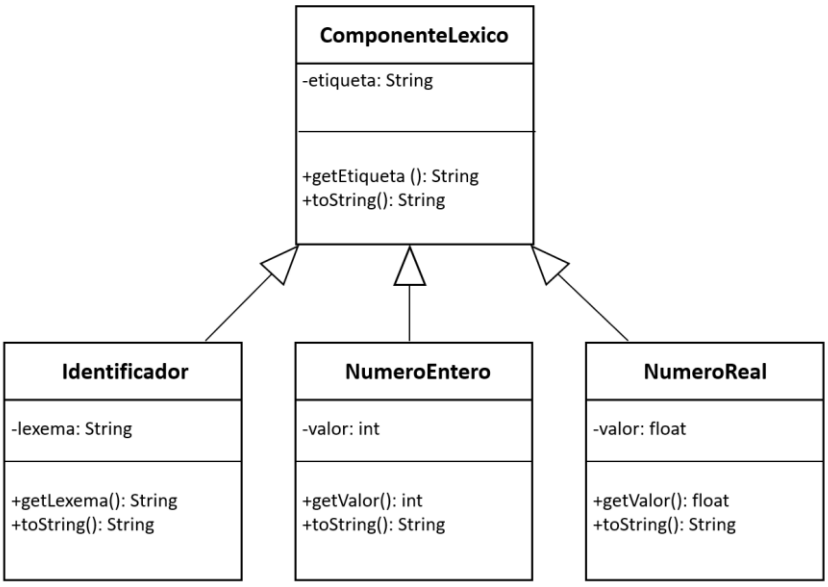
Este fichero debe almacenarse en la carpeta del proyecto de procesadores de lenguaje. Para crear el fichero en Eclipse, seleccione el proyecto y agregue un fichero de texto nuevo.

La clase `ComponentesLexicos`.

```
package practica_2;

import java.io.File;
import java.io.IOException;
import java.util.Hashtable;
import java.util.Map;
import java.util.Scanner;
import java.util.Set;
public class PalabrasReservadas {

   /*
    * ComponentesLexicos es una tabla Hash (String, String) que
    * almacena los componentes léxicos del lenguaje, definidos
    * por parejas <lexema, etiquetaLexica> donde el lexema es
    * la clave de la tabla y la etiqueta léxica el valor
    */

   private Hashtable<String, String> componentesLexicos;

   public ComponentesLexicos(String ficheroComponentesLexicos) {
      this.componentesLexicos = new Hashtable<String, String>();
      leeComponentesLexicos(this.componentesLexicos,
                            ficheroComponentesLexicos);
   }

   public String getEtiqueta(String lexema) {
      return this.componentesLexicos.get(lexema);
   }

   public String getLexema(String etiquetaLexica) {
      String lexema = null;

      Set<Map.Entry<String, String>> s =
         this.componentesLexicos.entrySet();

      for(Map.Entry<String, String> m : s)
         if (m.getValue().equals(etiquetaLexica)) {
            lexema = m.getKey();
            break;
         }

      return lexema;
   }
```

```
    private static boolean existeFichero(String fichero) {
       File ficheroEntrada = new File (fichero);

       return ficheroEntrada.exists();
    }

    private static String etiqueta(String s) {
       return s.substring(0, s.indexOf("\t")).trim();
    }

    private static String lexema(String s) {
       return s.substring(s.lastIndexOf("\t") + 1).trim();
    }

    private static void leeComponentesLexicos(Hashtable<String, String>
            componentesLexicos, String ficheroComponentesLexicos) {

       if (existeFichero(ficheroComponentesLexicos)) {
          try {

             Scanner fichero = new Scanner(new File
                               (ficheroComponentesLexicos), "UTF-8");

             String componenteLexico, lexema, etiquetaLexica;

             while (fichero.hasNext()) {
                componenteLexico = fichero.nextLine();

                if (componenteLexico.length() > 0 &&
                    componenteLexico.charAt(0) != '/') {

                   lexema = lexema(componenteLexico);
                   etiquetaLexica = etiqueta(componenteLexico);

                   componentesLexicos.put(lexema, etiquetaLexica);
                }
             }

             fichero.close();

          } catch (IOException e) {}
       }
    }

}
```

Los componentes léxicos.



La clase `ComponenteLexico`.

```
package practica_2;

public class ComponenteLexico {
   private String etiqueta;

   public ComponenteLexico(String etiqueta) {
      this.etiqueta = etiqueta;
   }

   public String getEtiqueta() {
      return this.etiqueta;
   }

   public String toString() {
      return this.etiqueta;
   }

}
```

La clase `Identificador`.

```
package practica_2;

public class Identificador extends ComponenteLexico{
   private String lexema;

   public Identificador(String lexema) {
   }

   public String getLexema() {
   }

   public String toString() {
   }

}
```

La clase `NumeroEntero`.

```
package practica_2;

public class NumeroEntero extends ComponenteLexico{
   private int valor;

   public NumeroEntero(int valor) {
   }

   public int getValor() {
   }

   public String toString() {
   }

}
```

La clase `NumeroReal`.

```
package practica_2;

public class NumeroReal extends ComponenteLexico {
   private float valor;

   public NumeroReal(float valor) {
   }

   public float getValor() {
   }

   public String toString() {
   }

}
```

Esta nueva versión del analizador utiliza la tabla Hash `componentesLexicos` para buscar las etiquetas léxicas correspondientes a los operadores aritméticos, relacionales, lógicos y caracteres delimitadores. Si se hace una búsqueda de un símbolo que no está en la tabla Hash, como '@', se devuelve el componente léxico `invalid_char`. Si el símbolo existe, por ejemplo '=', es necesario comprobar que el siguiente carácter de la entrada, concatenado con '=', no produce también un componente léxico válido. Por ejemplo, si se tiene '=', es necesario comprobar que el siguiente carácter es distinto de '=', ya que en este caso se debe devolver el componente léxico `equals` en vez de `assignment`.

```
// operadores aritméticos, relacionales, lógicos
// y caracteres delimitadores

   String lexema = "", lexemaAlternativo, etiquetaAlternativa;

   do  {
      lexema = lexema + this.caracter;
      etiquetaLexica = componentesLexicos.getEtiqueta(lexema);

      if (etiquetaLexica == null)
         return new ComponenteLexico("invalid_char");
      else {
         lexemaAlternativo = lexema;
         this.caracter = extraeCaracter();

         lexemaAlternativo = lexemaAlternativo + this.caracter;
         etiquetaAlternativa =
            componentesLexicos.getEtiqueta(lexemaAlternativo);

         if (etiquetaAlternativa != null)
            etiquetaLexica = etiquetaAlternativa;
      }

   } while (etiquetaAlternativa != null);

   devuelveCaracter();

   return new ComponenteLexico(etiquetaLexica);
```

El programa de prueba.

```
package practica_2;

import java.nio.charset.StandardCharsets;

public class TestLexico {

   public static void main(String[] args) {

      String ficheroEntrada = "programa.txt";

      ComponenteLexico componenteLexico;

      Lexico lexico = new Lexico(ficheroEntrada,
                          StandardCharsets.UTF_8);

      int c = 0;

      do {
         componenteLexico = lexico.getComponenteLexico();

         System.out.println("<" + componenteLexico.toString() + ">");

         c++;
      } while (!componenteLexico.getEtiqueta().equals("end_program"));

      System.out.println("\nComponentes léxicos: " + c +
                         ", lineas: " + lexico.getLineas() + "\n\n");

   }

}
```

El código fuente del programa de prueba.

```
{
   float suma = 0.0;

   int [10] v;

   for (int k=0; k<10; k=k+1) {
      if (k % 2 == 0)
         suma = suma + k*10.5;
      else
         suma = suma + k*15.75;

      v[i] = suma;
   }

   if (suma <= 25.0)
      suma = suma / 2.5;
   else
      suma = suma * 4.5;

}
```

La salida por la consola:

```
<open_bracket>
<float>
<id, suma>
<assignment>
<float, 0.0>
<semicolon>
<int>
<open_square_bracket>
<int, 10>
<closed_square_bracket>
<id, v>
<semicolon>
<id, for>
<open_parenthesis>
<int>
<id, k>
<assignment>
<int, 0>
<semicolon>
<id, k>
<less_than>
<int, 10>
<semicolon>
<id, k>
<assignment>
<id, k>
<add>
<int, 1>
<closed_parenthesis>
<open_bracket>
<if>
<open_parenthesis>
<id, k>
<remainder>
<int, 2>
<equals>
<int, 0>
<closed_parenthesis>
<id, suma>
<assignment>
<id, suma>
<add>
<id, k>
<multiply>
```

```
<float, 10.5>
<semicolon>
<else>
<id, suma>
<assignment>
<id, suma>
<add>
<id, k>
<multiply>
<float, 15.75>
<semicolon>
<id, v>
<open_square_bracket>
<id, i>
<closed_square_bracket>
<assignment>
<id, suma>
<semicolon>
<closed_bracket>
<if>
<open_parenthesis>
<id, suma>
<less_equals>
<float, 25.0>
<closed_parenthesis>
<id, suma>
<assignment>
<id, suma>
<divide>
<float, 2.5>
<semicolon>
<else>
<id, suma>
<assignment>
<id, suma>
<multiply>
<float, 4.5>
<semicolon>
<closed_bracket>
<end_program>

Componentes léxicos: 84, lineas: 20
```