202002940

Sebastian Edgardo Godoy Salvatierra

Manual tecnico

Interfaz gráfica: mediante la herramienta drag and drop de netbeans se extiende un jframe y se le agregan todos los botones y funciones, debido a la naturaleza del código extendido en la interfaz grafica y su poca relevancia con el resto de la funcionalidad del proyecto se mostarara de una manera muy breve:

```
17      public class UI extends javax.swing.JFrame {
18      File file_current = null;
19
20          public UI() {
21              initComponents();
22          }
23
24          /**
25           * This method is called from within the constructor to initialize the form.
26           * WARNING: Do NOT modify this code. The content of this method is always
27           * regenerated by the Form Editor.
28           */
29          @SuppressWarnings("unchecked")
30          Generated Code
159
160         private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
161             // TODO add your handling code here:
162             JFileChooser fileChooser = new JFileChooser();
163             int seleccion = fileChooser.showSaveDialog(parent: txta_editor);
164
165              if (seleccion == JFileChooser.APPROVE_OPTION){
166                  String content = "";
167                  file_current = fileChooser.getSelectedFile();
168                  try{
169                      Scanner input = new Scanner (source: file_current);
170                      while (input.hasNextLine()){
171                          content += input.nextLine()+"\n";
172                      }
173                      input.close();
174
175                  }
176                  catch (Exception ex){
177                      ex.printStackTrace();
178                  }
179                  txta_editor.setText(t: content);
180              }
181         }
182
183         private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
184             // TODO add your handling code here:
185             File file = file_current;
186             String text = txta_editor.getText();
187             try (PrintWriter out = new PrintWriter(file, charset: StandardCharsets.UTF_8)){
188                 out.print(x: text);
189                 System.out.println(x: "Guardado exitosamente");
190
191             }
192             catch (IOException e){
```

Lexer y parser: la herramienta a utilizar para reconocimiento de caracteres y tokens es el lexer denominado jflex de java, aquí se establecen las expresiones regulares y como entran el juego en el resto del proyecto

```
20    comment="<!" (.|[^.])+ "!>"
21    lnComment = "//" .+
22    llaveA="("|"{"
23    llaveC=")"|"}"
24    CONJ="CONJ"
25    dbPto=":"
26    ID=[a-zA-Z_][0-9a-zA-Z_]+
27    flecha="->"
28    ptoComa=";"
29    minus=[a-z]
30    mayus=[A-Z]
31    nums=[0-9]
32    porcentaje="%"
33    signos=[\ -/:-@\[-`{-}]
34    virgulilla="~"
35    coma=","
36    concatenacion="."
37    disyuncion="|"
38    estrella="*"
39    cerrPositiva="+"
40    cerrBooleana="?"
41    caracter="\""({minus}|{mayus}|{nums}|{signos})"\""
42    input=[^\"\n\r]
43    cadena="\""({input})+"\""
44    espacioBlanco=[\ \r\t]+
45    finDeLinea=[\r\n]
```

Se imprime cada coincidencia encontrada para poder ser mostrada en consola como parte del reporte de tokens

```
47    <YYINITIAL> {espacioBlanco} {}
48    <YYINITIAL> {finDeLinea} {}
49    <YYINITIAL> {comment} { System.out.println(yytext());}
50    <YYINITIAL> {lnComment} { System.out.println(yytext());}
51    <YYINITIAL> {CONJ} { System.out.println("CONJ " + yytext());return new Symbol(sym.CONJ, yyline, yycolumn, yytext());}
52    <YYINITIAL> {concatenacion} { System.out.println("Concatenacion " + yytext());return new Symbol(sym.concatenacion, yyline, yycolumn, yytext());}
53    <YYINITIAL> {disyuncion} { System.out.println("Disyuncion " + yytext());return new Symbol(sym.disyuncion, yyline, yycolumn, yytext());}
54    <YYINITIAL> {estrella} { System.out.println("Estrella " + yytext());return new Symbol(sym.estrella, yyline, yycolumn, yytext());}
55    <YYINITIAL> {cerrPositiva} { System.out.println("cerrPositiva " + yytext());return new Symbol(sym.cerrPositiva, yyline, yycolumn, yytext());}
56    <YYINITIAL> {cerrBooleana} { System.out.println("cerrBooleana " + yytext());return new Symbol(sym.cerrBooleana, yyline, yycolumn, yytext());}
57    <YYINITIAL> {llaveA} { System.out.println("llaveA " + yytext());return new Symbol(sym.llaveA, yyline, yycolumn, yytext());}
58    <YYINITIAL> {llaveC} { System.out.println("llaveC " + yytext());return new Symbol(sym.llaveC, yyline, yycolumn, yytext());}
59    <YYINITIAL> {dbPto} { System.out.println("dbPto " + yytext());return new Symbol(sym.dbPto, yyline, yycolumn, yytext());}
60    <YYINITIAL> {ID} { System.out.println("ID " + yytext());return new Symbol(sym.ID, yyline, yycolumn, yytext());}
61    <YYINITIAL> {flecha} { System.out.println("Flecha " + yytext());return new Symbol(sym.flecha, yyline, yycolumn, yytext());}
62    <YYINITIAL> {ptoComa} { System.out.println("ptoComa " + yytext());return new Symbol(sym.ptoComa, yyline, yycolumn, yytext());}
63    <YYINITIAL> {minus} { System.out.println("minus " + yytext());return new Symbol(sym.minus, yyline, yycolumn, yytext());}
64    <YYINITIAL> {mayus} { System.out.println("mayus " + yytext());return new Symbol(sym.mayus, yyline, yycolumn, yytext());}
65    <YYINITIAL> {nums} { System.out.println("nums " + yytext());return new Symbol(sym.nums, yyline, yycolumn, yytext());}
66    <YYINITIAL> {virgulilla} { System.out.println("Virgulilla " + yytext());return new Symbol(sym.virgulilla, yyline, yycolumn, yytext());}
67    <YYINITIAL> {coma} { System.out.println("coma " + yytext());return new Symbol(sym.coma, yyline, yycolumn, yytext());}
68    <YYINITIAL> {caracter} { System.out.println("caracter " + yytext());return new Symbol(sym.caracter, yyline, yycolumn, yytext());}
69    <YYINITIAL> {cadena} { System.out.println("cadena " + yytext());return new Symbol(sym.cadena, yyline, yycolumn, yytext());}
70    <YYINITIAL> {porcentaje} { System.out.println("porcentaje " + yytext());return new Symbol(sym.porcentaje, yyline, yycolumn, yytext());}
71    <YYINITIAL> {signos} { System.out.println("Signos " + yytext());return new Symbol(sym.signos, yyline, yycolumn, yytext());}
72
```

JCUP: aquí si establecen las producciones y gramática, trabaja en conjunto con jflex para poder utilizar los tokens validados y poder empezar el árbol binario.

```
43
44    start with programa;
45
46    programa ::= llaveA segmentoA_varios llaveC;
47    segmentoA_varios::= segmentoA_varios segmentoA | segmentoA;
48    segmentoA::= CONJ dbPto ID flecha notacion ptoComa|CONJ dbPto ID flecha notacionA ptoComa|
49    ID flecha exReg:a ptoComa {:
50        arboles.add(new Automata((Nodo_binario) a));
51    :};
52
53    notacion::= minus virgulilla minus |mayus virgulilla mayus
54    |nums virgulilla nums | signos virgulilla signos;
55    notacionA::=notacionA coma notacionB | notacionB;
56    notacionB::=signos|nums|minus|mayus;
57
58
59    exReg::= concatenacion:a exReg:b exReg:c{:
60    Nodo_binario padre = new Nodo_binario(a);
61    padre.hijo_izq = (Nodo_binario) b;
62    padre.hijo_der = (Nodo_binario) c;
63    RESULT = padre;
64    :}
65    |disyuncion:a exReg:b exReg:c {:
66    Nodo_binario padre = new Nodo_binario(a);
67    padre.hijo_izq = (Nodo_binario) b;
68    padre.hijo_der = (Nodo_binario) c;
69    RESULT = padre;
70    :}
71    |estrella:a exReg:b{:
72    Nodo_binario padre = new Nodo_binario(a);
73    padre.hijo_der = (Nodo_binario) b;
74    RESULT = padre;
75    :}
76    |cerrPositiva:a exReg:b{:
77    Nodo_binario padre = new Nodo_binario(a);
```

Automata: esta clase contiene la generación del metodo del árbol y el metodo de Thompson, dichos metodos utilizan funciones recursivas para poder generar sus outputs esperados.

```
public void metodo_arbol(Nodo_binario actual) {
    if (actual == null) {
        return;
    }

    if (actual.isHoja()) {
        // Agregar ids propios
        actual.getPrimeros().add(e: actual.getIdentificador());
        // El ultimo es el num de hoja
        actual.getUltimos().add(e: actual.getIdentificador());
        return;
    }

    metodo_arbol(actual: actual.getHijo_izquierdo());
    metodo_arbol(actual: actual.getHijo_derecho());

    if(actual.getDato().equals(anObject: "*")) {
        actual.setAnulable(anulable: true);
        actual.getPrimeros().addAll(e: actual.getHijo_derecho().getPrimeros());
        // Ultimos del primer hijo
        actual.getUltimos().addAll(e: actual.getHijo_derecho().getPrimeros());
        for (int i = 0; i < actual.getHijo_derecho().getPrimeros().size(); i++) {
            for (int j = 0; j < actual.getHijo_derecho().getUltimos().size(); j++) {
                siguientes.get(actual.getHijo_derecho().getPrimeros().get(index: i)-1).getSiguientes().add(e: actual.getHijo_derecho().getUltimos().get(index: j));
            }
        }
    }
    else if(actual.getDato().equals(anObject: "+")) {
        actual.setAnulable(anulable: actual.getHijo_derecho().isAnulable());
        actual.getPrimeros().addAll(e: actual.getHijo_derecho().getPrimeros());
        for (int i = 0; i < actual.getHijo_derecho().getPrimeros().size(); i++) {
            for (int j = 0; j < actual.getHijo_derecho().getUltimos().size(); j++) {
                siguientes.get(actual.getHijo_derecho().getPrimeros().get(index: i)-1).getSiguientes().add(e: actual.getHijo_derecho().getUltimos().get(index: j));
            }
        }
    }
    else if(actual.getDato().equals(anObject: "?")) {
        // Si es anulable
        actual.setAnulable(anulable: true);
        actual.getPrimeros().addAll(e: actual.getHijo_derecho().getPrimeros());
        actual.getUltimos().addAll(e: actual.getHijo_derecho().getPrimeros());
```

Metodo de thomspon:

```
public String r_crear_afnd(int inicio,int fin,Nodo_binario actual) {
    String resultado = "";
    if (actual == null) {
        return resultado;
    }
    if(actual.hoja){
        System.out.println(x: actual.dato);
        resultado = "N"+inicio+"->N"+fin+"[label=\""+actual.dato.replaceAll(regex: "\"", replacement: "")+"\"];\n";
}
    else{
        switch(actual.dato){
            case "." -> {
                nodos_afd += 1;
                int anterior = nodos_afd;
                resultado += "N"+nodos_afd+"[label=\"\"];\n";
                resultado += r_crear_afnd(inicio, fin: anterior, actual: actual.getHijo_izquierdo());
                resultado += r_crear_afnd(inicio: anterior,fin, actual: actual.getHijo_derecho());
            }
            case "|" ->{
                nodos_afd += 1;
                resultado += "N"+nodos_afd+"[label=\"\"];\n";
                resultado += "N"+inicio+"->N"+nodos_afd+"[label=\"ε\"];\n";
                nodos_afd += 1;
                resultado += "N"+nodos_afd+"[label=\"\"];\n";
                resultado += "N"+nodos_afd+"->N"+fin+"[label=\"ε\"];\n";
                resultado += r_crear_afnd(nodos_afd-1, fin: nodos_afd, actual: actual.getHijo_izquierdo());
                nodos_afd += 1;
                resultado += "N"+nodos_afd+"[label=\"\"];\n";
                resultado += "N"+inicio+"->N"+nodos_afd+"[label=\"ε\"];\n";
                nodos_afd += 1;
                resultado += "N"+nodos_afd+"[label=\"\"];\n";
                resultado += "N"+nodos_afd+"->N"+fin+"[label=\"ε\"];\n";
                resultado += r_crear_afnd(nodos_afd-1, fin: nodos_afd, actual: actual.getHijo_derecho());
            }
            case "+" ->{
                nodos_afd += 1;
                resultado += "N"+nodos_afd+"[label=\"\"];\n";
                resultado += "N"+inicio+"->N"+nodos_afd+"[label=\"ε\"];\n";
                nodos_afd += 1;
```