UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
LENGUAJES FORMALES Y DE PROGRAMACIÓN
SEGUNDO SEMESTRE

MANUAL

-

TÉCNICO

TABLA DE CONTENIDO

LICENCIA DEL DESARROLLADOR

CHRISTOPHER IVÁN MONTERROSO ALEGRIA.

CARNÉ:201902363

ESTUDIANTE INGENIERIA EN CIENCIAS Y SISTEMAS.

REQUSISITOS DE LA APLICACIÓN:

- **WINDOWS 10,8,7(X64, X32).**
- **PROCESADOR 1.4GHZ O SUPERIOR**
- **2 GB RAM MINIMO, 4 RECOMENDADO**
- **500 MB ESPACIO LIBRE**
- **JAVA 8.**
- **JDK 11 O SUPERIOR.**

VENTANA:

PARA LA VENTANA SE UTILIZÓ JAVA.SWING CON LA CUAL SE CREARON TODOS LOS BOTONES Y ACCIONES DE FORMA MANUAL.

```
import static javax.swing.WindowConstants.EXIT_ON_CLOSE;

public class home extends JFrame implements ActionListener {

    static JTextArea textArchivoEntrada, consola;
    private JButton generarAutomata, analizarEntrada,abrirUbicacion;
    private JMenuItem newA, open, save, saveAs;
    private JComboBox comboBox;
    private String ruta = "";

    public home() {
        this.setTitle("ExRegan USAC");
        this.setSize(900, 700);
        this.setDefaultCloseOperation(EXIT_ON_CLOSE);
        this.setResizable(false);
        this.setLayout(null);
        this.setLocationRelativeTo(null);

        widgets();

    }
```

```
public void widgets() {
    Border borde = BorderFactory.createLineBorder(Color.BLACK, 2, true);
    JMenuBar menuBar = new JMenuBar();

    JMenu archive = new JMenu("Archivo");
    newA = new JMenuItem("Nuevo");
    newA.addActionListener(this);
    open = new JMenuItem("Abrir");
    open.addActionListener(this);
    save = new JMenuItem("Guardar");
    save.addActionListener(this);
    saveAs = new JMenuItem("Guardar como");
    saveAs.addActionListener(this);
    archive.add(newA);
    archive.add(open);
    archive.add(save);
    archive.add(saveAs);

    menuBar.add(archive);
    setJMenuBar(menuBar);

    generarAutomata = new JButton("Generar Autómata");
    generarAutomata.setBounds(30, 400, 150, 30);
    generarAutomata.addActionListener(this);
    add(generarAutomata);
```

AQUÍ SE LE DAN PROPIEDADES A LOS BOTONES PARA LA APLICACIÓN

```
    generarAutomata = new JButton("Generar Autómata");
    generarAutomata.setBounds(30, 400, 150, 30);
    generarAutomata.addActionListener(this);
    add(generarAutomata);

    abrirUbicacion = new JButton("Mostrar archivos");
    abrirUbicacion.setBounds(550, 100, 150, 40);
    abrirUbicacion.addActionListener(this);
    add(abrirUbicacion);

    analizarEntrada = new JButton("Analizar entrada");
    analizarEntrada.setBounds(265, 400, 150, 30);
    analizarEntrada.addActionListener(this);
    add(analizarEntrada);

    JLabel labelArchivo = new JLabel("Archivo");
    labelArchivo.setBounds(25, 10, 150, 30);
    add(labelArchivo);

    textArchivoEntrada = new JTextArea();
    textArchivoEntrada.setBounds(25, 35, 500, 350);
    textArchivoEntrada.setBorder(borde);
    JScrollPane scrollPane = new JScrollPane();
    scrollPane.setBounds(25, 35, 500, 350);
    scrollPane.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);
    scrollPane.getViewport().setBackground(Color.WHITE);
    scrollPane.getViewport().add(textArchivoEntrada);
    add(scrollPane);
    //add(textArchivoEntrada);

    JLabel labelConsola = new JLabel("Consola");
    labelConsola.setBounds(25, 460, 150, 30);
```

```
    consola = new JTextArea();
    consola.setBorder(borde);
    consola.setBounds(25, 490, 825, 125);
    JScrollPane scrollPane2 = new JScrollPane();
    scrollPane2.setBounds(25, 490, 825, 125);
    scrollPane2.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);
    scrollPane2.getViewport().setBackground(Color.WHITE);
    scrollPane2.getViewport().add(consola);
    add(scrollPane2);

    String[] elements = {"Arboles", "Siguientes", "Transiciones", "AFD", "AFND", "Errores", "Salidas"};
    comboBox = new JComboBox(elements);
    comboBox.setBounds(570, 35, 100, 40);

    add(comboBox);
}
```

SE CREARON LOS METODOS GUARDAR COMO, GUARDAR, NUEVO ARCHIVO Y ABRIR.

```
@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == newA) {
        ruta = "";
        textArchivoEntrada.setText("");
        this.saveAS();

    } else if (e.getSource() == open) {
        JFileChooser fileChooser = new JFileChooser();
        int result = fileChooser.showOpenDialog(this);
        if (result == JFileChooser.APPROVE_OPTION) {
            File selectedFile = fileChooser.getSelectedFile();
            String fileContent = null;
            ruta = selectedFile.getPath();
            System.out.println(ruta);
            try (BufferedReader reader = new BufferedReader(new FileReader(selectedFile))) {
                String line = null;
                StringBuilder stringBuilder = new StringBuilder();
                while ((line = reader.readLine()) != null) {
                    stringBuilder.append(line).append("\n");
                }
                fileContent = stringBuilder.toString();
            } catch (FileNotFoundException ex) {
                Logger.getLogger(home.class.getName()).log(Level.SEVERE, null, ex);
            } catch (IOException ex) {
                Logger.getLogger(home.class.getName()).log(Level.SEVERE, null, ex);
            }
            textArchivoEntrada.setText("");
            textArchivoEntrada.setText(fileContent);


        }
```

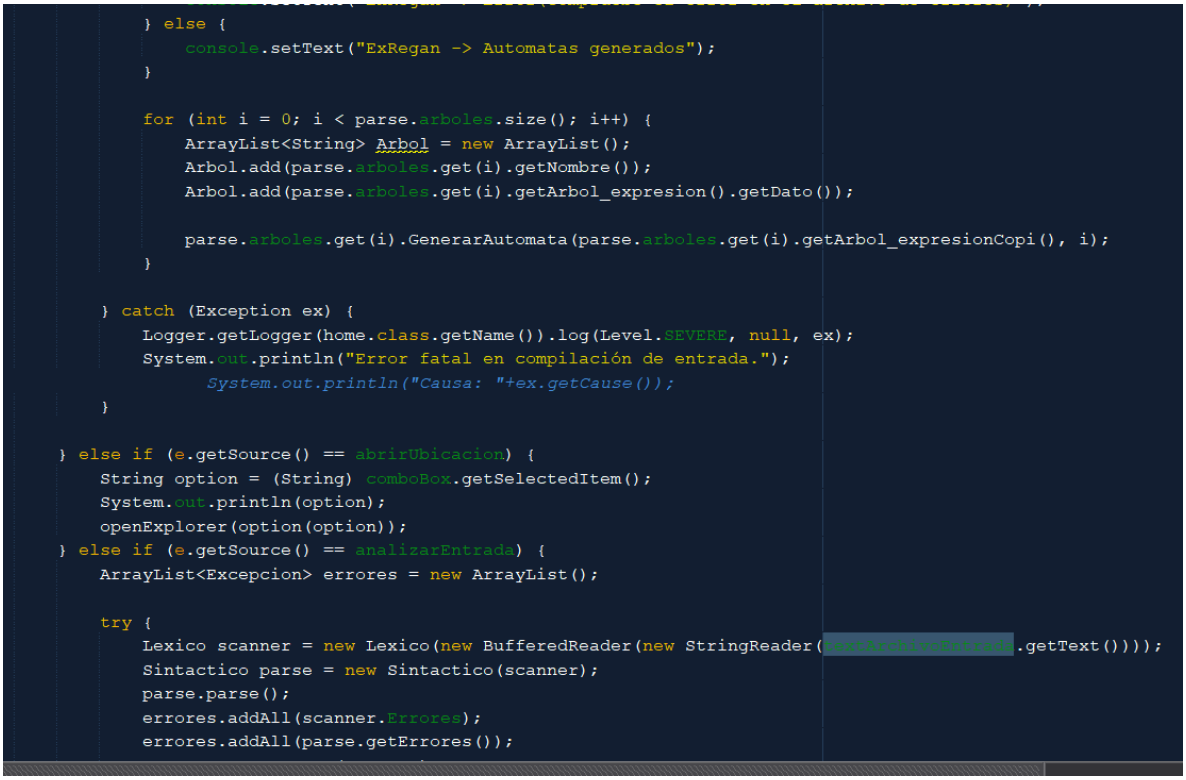
```
    } else if (e.getSource() == save) {
        if ("".equals(ruta)) {
            this.saveAS();
        } else {
            try {
                try (FileWriter writer = new FileWriter(ruta)) {
                    writer.write(textArchivoEntrada.getText());
                }
                JOptionPane.showMessageDialog(null, "Se han guardado los cambios");
                System.out.println("guardado");
            } catch (IOException s) {
                JOptionPane.showMessageDialog(null, "No se han guardado los cambios");
            }
        }

    } else if (e.getSource() == saveAs) {
        this.saveAS();

    } else if (e.getSource() == generarAutomata) {

        ArrayList<Excepcion> errores = new ArrayList();
        try {
            Lexico scanner = new Lexico(new BufferedReader(new StringReader(textArchivoEntrada.getText())));
            Sintactico parse = new Sintactico(scanner);
            parse.parse();
            errores.addAll(scanner.Errores);
            errores.addAll(parse.getErrores());
            generarReporteHTML(errores);
            if (errores.size() >= 1) {
                console.setText("ExRegan -> Error(compruebe el error en el archivo de errores)");
            } else {
```

AQUÍ INICIAMOS EL ANALIZADOR LEXICO Y SINTACTICO PARA POSTERIORMENTE GENERAR LOS RESPECTIVOS ARCHIVOS, TENIENDO EN CUENTA DE QUE SI OCURRE UN ERROR LA SE GENERARÁ EL ARCHIVO.



```
            } else {
                console.setText("ExRegan -> Automatas generados");
            }

            for (int i = 0; i < parse.arboles.size(); i++) {
                ArrayList<String> Arbol = new ArrayList();
                Arbol.add(parse.arboles.get(i).getNombre());
                Arbol.add(parse.arboles.get(i).getArbol_expresion().getDato());

                parse.arboles.get(i).GenerarAutomata(parse.arboles.get(i).getArbol_expresionCopi(), i);
            }

        } catch (Exception ex) {
            Logger.getLogger(home.class.getName()).log(Level.SEVERE, null, ex);
            System.out.println("Error fatal en compilación de entrada.");
                  System.out.println("Causa: "+ex.getCause());
        }

    } else if (e.getSource() == abrirUbicacion) {
        String option = (String) comboBox.getSelectedItem();
        System.out.println(option);
        openExplorer(option(option));
    } else if (e.getSource() == analizarEntrada) {
        ArrayList<Excepcion> errores = new ArrayList();

        try {
            Lexico scanner = new Lexico(new BufferedReader(new StringReader(textArchivoEntrada.getText())));
            Sintactico parse = new Sintactico(scanner);
            parse.parse();
            errores.addAll(scanner.Errores);
            errores.addAll(parse.getErrores());
```

AQUÍ SE VUELVE A INICIAR EL ANALIZADOR LEXICO Y SINTACTICO PERO UNICAMENTE PARA PODER ANALIZAR LA ENTRADA Y POSTERIORMENTE MOSTRAR LOS RESULTADOS EN CONSOLA Y EN UN ARCHIVO JSON.

```
    } else if (e.getSource() == analizarEntrada) {
        ArrayList<Excepcion> errores = new ArrayList();

        try {
            Lexico scanner = new Lexico(new BufferedReader(new StringReader(textArchivoEntrada.getText())));
            Sintactico parse = new Sintactico(scanner);
            parse.parse();
            errores.addAll(scanner.Errores);
            errores.addAll(parse.getErrores());
            generarReporteHTML(errores);

            ArrayList<ArrayList> Conjuntos_Arreglados = ArreglarConjuntos(parse.conjuntos);

            for (int i = 0; i < parse.arboles.size(); i++) {
                parse.arboles.get(i).Analizar(parse.arboles.get(i).getArbol_expresionCopi(), Conjuntos_Arreglados, (ArrayList<ArrayList
            }

        } catch (Exception ex) {
            Logger.getLogger(home.class.getName()).log(Level.SEVERE, null, ex);
            System.out.println("Error fatal en compilación de entrada.");
                  System.out.println("Causa: "+ex.getCause());
        }
    }
}
```

CODIGO PARA GENERAR LOS ARBOLES CUMPLIENDO LAS REGLAS RESPECTIVAS, Y GENERACIÓN DEL PNG.

```
public class Automata {

    private Nodo_binario arbol_expresion, arbol_expresionCopi;;
    private int num_nodo = 1;
    private String nombre;
    ArrayList<ArrayList> table = new ArrayList();

    ArrayList<Nodo_binario> leaves = new ArrayList();

     public Automata(Nodo_binario arbol_expresion,String nombre){
        this.arbol_expresion=arbol_expresion;
        this.arbol_expresionCopi=arbol_expresion;
        this.nombre=nombre;
    }
    public void Analizar(Nodo_binario arbol_expresion,ArrayList<ArrayList> conjuntos,ArrayList<ArrayList> declaraciones){
        Nodo_binario raiz = new Nodo_binario(".");
        Nodo_binario aceptacion = new Nodo_binario("#");
        aceptacion.setHoja(true);
        aceptacion.setAnulable(false);
        raiz.setHijo_derecho(aceptacion);
        leave hoja = new leave();

        hoja.addLeave(aceptacion, leaves);
        raiz.setHijo_izquierdo(arbol_expresion);
        this.arbol_expresion = raiz;
        asignar_numeros(this.arbol_expresion);
        num_nodo = 0;
        metodo_arbol(this.arbol_expresion);


        graficar_arbol(this.arbol_expresion, num_nodo);
```

```
    }
    public void GenerarAutomata(Nodo_binario arbol_expresion,int i) throws IOException {
        Nodo_binario raiz = new Nodo_binario(".");
        Nodo_binario aceptacion = new Nodo_binario("#");
        aceptacion.setHoja(true);
        aceptacion.setAnulable(false);
        raiz.setHijo_derecho(aceptacion);
        leave hoja = new leave();

        hoja.addLeave(aceptacion, leaves);
        raiz.setHijo_izquierdo(arbol_expresion);
        this.arbol_expresion = raiz;
        asignar_numeros(this.arbol_expresion);
        num_nodo = 0;
        metodo_arbol(this.arbol_expresion);

        GenerarDot(graficar_arbol(this.arbol_expresion, num_nodo),nombre);
        follows(this.arbol_expresion);
        tablaSiguientes ft = new tablaSiguientes();

        ft.printTable(table,nombre);
        tablaTransicion tran = new tablaTransicion(raiz, table, leaves); // bug

        tran.impTable(nombre);

        tran.impGraph(nombre);
    }
```

```
public void asignar_numeros(Nodo_binario actual) {
    if (actual == null) {
        return;
    }
    if (actual.isHoja()) {
        actual.setIdentificador(num_nodo);
        leave hoja = new leave();

        hoja.addLeave(actual, leaves); //Tabla de siguientes o transiciones
        num_nodo++;
        return;
    }
    asignar_numeros(actual.getHijo_izquierdo());
    asignar_numeros(actual.getHijo_derecho());
}

public void metodo_arbol(Nodo_binario actual) {
    if (actual == null) {
        return;
    }
    metodo_arbol(actual.getHijo_izquierdo());
    metodo_arbol(actual.getHijo_derecho());

    if (actual.isHoja()) {
        actual.getPrimeros().add(actual.getIdentificador());
        actual.getUltimos().add(actual.getIdentificador());
        return;
    }
    if (actual.getDato().equals("*")) {
```

```
        actual.setAnulable(actual.getHijo_izquierdo().isAnulable());
        actual.getPrimeros().addAll(actual.getHijo_izquierdo().getPrimeros());
        actual.getUltimos().addAll(actual.getHijo_izquierdo().getUltimos());
    } else if (actual.getDato().equals("?")) {
        actual.setAnulable(true);
        actual.getPrimeros().addAll(actual.getHijo_izquierdo().getPrimeros());
        actual.getUltimos().addAll(actual.getHijo_izquierdo().getUltimos());
    } else if (actual.getDato().equals("|")) {
        actual.setAnulable(actual.getHijo_izquierdo().isAnulable() || actual.getHijo_derecho().isAnulable());
        actual.getPrimeros().addAll(actual.getHijo_izquierdo().getPrimeros());
        actual.getPrimeros().addAll(actual.getHijo_derecho().getPrimeros());
        actual.getUltimos().addAll(actual.getHijo_izquierdo().getUltimos());
        actual.getUltimos().addAll(actual.getHijo_derecho().getUltimos());
    } else if (actual.getDato().equals(".")) {
        actual.setAnulable(actual.getHijo_izquierdo().isAnulable() && actual.getHijo_derecho().isAnulable());
        if (actual.getHijo_izquierdo().isAnulable()) {
            actual.getPrimeros().addAll(actual.getHijo_izquierdo().getPrimeros());
            actual.getPrimeros().addAll(actual.getHijo_derecho().getPrimeros());
        } else {
            actual.getPrimeros().addAll(actual.getHijo_izquierdo().getPrimeros());
        }
        if (actual.getHijo_derecho().isAnulable()) {
            actual.getUltimos().addAll(actual.getHijo_izquierdo().getUltimos());
            actual.getUltimos().addAll(actual.getHijo_derecho().getUltimos());
        } else {
            actual.getUltimos().addAll(actual.getHijo_derecho().getUltimos());
        }
```

```
public String graficar_arbol(Nodo_binario nodo, int padre) {
    String cadena = "";
    num_nodo += 1;
    int actual = num_nodo;
    if (nodo == null) {
        num_nodo -= 1;
        return cadena;
    }

    if (nodo.isHoja()) {
        String anulable = "A";
        if (nodo.isAnulable() == false) {
            anulable = "N";
        }
        cadena += "n_" + actual + "[shape=none label=<\n"
                + "<table border =\"1\" cellspacing=\"2\" cellpadding=\"10\" >\n"
                + " <tr>\n"
                + " <td colspan=\"3\">" + anulable + "</td>\n"
                + " </tr>\n"
                + " <tr>\n"
                + " <td>" + nodo.getPrimeros() + "</td>\n"
                + " <td>" + nodo.getDato() + "</td>\n"
                + " <td>" + nodo.getUltimos() + "</td>\n"
                + " </tr>\n"
                + " <tr>\n"
                + " <td colspan=\"3\">" + nodo.getIdentificador() + "</td>\n"
                + " </tr>\n"
                + " </table>>];";

    } else {
```

```
        } else {
            String anulable = "A";
            if (nodo.isAnulable() == false) {
                anulable = "N";
            }
            cadena += "n_" + actual + "[shape=none label=<\n"
                    + "<table border =\"1\" cellspacing=\"2\" cellpadding=\"10\" >\n"
                    + " <tr>\n"
                    + " <td colspan=\"3\">" + anulable + "</td>\n"
                    + " </tr>\n"
                    + " <tr>\n"
                    + " <td>" + nodo.getPrimeros() + "</td>\n"
                    + " <td>" + nodo.getDato() + "</td>\n"
                    + " <td>" + nodo.getUltimos() + "</td>\n"
                    + " </tr>\n"
                    + " </table>>];";

        }
        if (padre != 0) {
            cadena += "n_" + padre + " -> n_" + actual + ";\n";
        }
        cadena += graficar_arbol(nodo.getHijo_izquierdo(), actual);
        cadena += graficar_arbol(nodo.getHijo_derecho(), actual);

        return cadena;
    }

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

```
public void follows(Nodo_binario actual) {
    if (actual == null) {
        return;
    }
    follows(actual.getHijo_izquierdo());
    follows(actual.getHijo_derecho());

    if (actual.getDato().equals(".")) {
        for (int item : ((Nodo_binario) actual.getHijo_izquierdo()).getUltimos()) {
            leave hoja = new leave();

            Nodo_binario nodo = hoja.getLeave(item, leaves);
            tablaSiguientes tabla = new tablaSiguientes();
            tabla.append(nodo.getIdentificador(), nodo.getDato(), ((Nodo_binario) actual.getHijo_derecho()).getPrimeros(), table);

        }

    } else if (actual.getDato().equals("*")) {
        for (int item : ((Nodo_binario) actual.getHijo_izquierdo()).getUltimos()) {

            leave hoja = new leave();
            Nodo_binario nodo = hoja.getLeave(item, leaves);
            tablaSiguientes tabla = new tablaSiguientes();
            tabla.append(nodo.getIdentificador(), nodo.getDato(), ((Nodo_binario) actual.getHijo_izquierdo()).getPrimeros(), table);

        }

    } else if (actual.getDato().equals("+")) {
        for (int item : ((Nodo_binario) actual.getHijo_izquierdo()).getUltimos()) {

            leave hoja = new leave();
```

```
    private void GenerarDot(String cadena, String nombre) {
        FileWriter fichero = null;
        PrintWriter escritor = null;
        String s="digraph G {\n ";
        s+=cadena;
        s+="\n }";
        try {
            fichero = new FileWriter("Arboles_201902363/Arbol_Sintactico" + nombre + ".dot");
            escritor = new PrintWriter(fichero);
            escritor.println(s);
            escritor.close();
            fichero.close();
            reportar(nombre);
            System.out.println("aqui");
        } catch (Exception e) {
            System.out.println("error en generar dot");
            e.printStackTrace();
        }
    }

    public void reportar(String nombre) throws IOException {

        String file_input_path = "Arboles_201902363/Arbol_Sintactico" + nombre + ".dot";
        String do_path = "C:\\Program Files (x86)\\Graphviz\\bin\\dot.exe";

        String file_get_path = "Arboles_201902363/Arbol_Sintactico_" + nombre + ".png";
        try {
            ProcessBuilder pBuilder;
            pBuilder = new ProcessBuilder(do_path, "-Tpng", "-o", file_get_path, file_input_path);
            pBuilder.redirectErrorStream(true);
```

```
    public void reportar(String nombre) throws IOException {

        String file_input_path = "Arboles_201902363/Arbol_Sintactico" + nombre + ".dot";
        String do_path = "C:\\Program Files (x86)\\Graphviz\\bin\\dot.exe";

        String file_get_path = "Arboles_201902363/Arbol_Sintactico_" + nombre + ".png";
        try {
            ProcessBuilder pBuilder;
            pBuilder = new ProcessBuilder(do_path, "-Tpng", "-o", file_get_path, file_input_path
            pBuilder.redirectErrorStream(true);
            pBuilder.start();
        } catch (Exception ex) {
            ex.printStackTrace();
        }

    }

    public Nodo_binario getArbol_expresion() {
        return arbol_expresion;
    }
    public Nodo_binario getArbol_expresionCopi() {

        return arbol_expresionCopi;
    }

    public String getNombre() {
        return nombre;
    }

}
```