PSEUDO PARSER – PROYECTO 1

MANUAL TÉCNICO

Allen RománPROGRAMA REALIZADO POR EL ALUMNO ALLEN GIANKARLO ROMÁN VÁSQUEZ CARNET 202004745 PARA EL CURSO ORGANIZACIÓN DE LENGUAJES Y COMPILADORES 1 DE LA ESCUELA DE CIENCIAS Y SISTEMAS DE LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD SAN CARLOS DE GUATEMALA

Programa Utilizado

IntelliJ IDEA es un entorno de desarrollo integrado (IDE) para el desarrollo de programas informáticos. Es desarrollado por JetBrains (anteriormente conocido como IntelliJ), y está disponible en dos ediciones: edición para la comunidad y edición comercial.

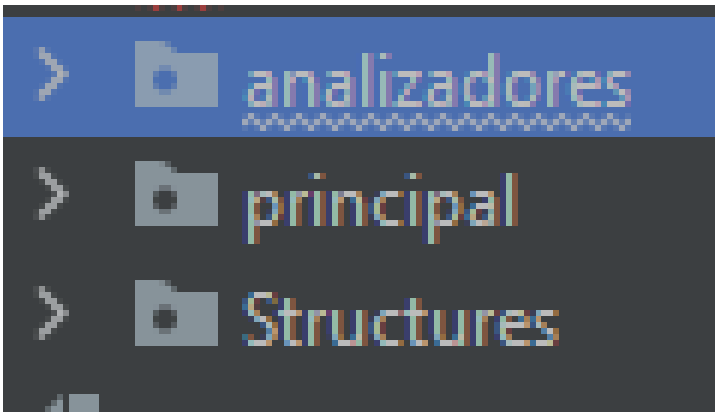
Lenguaje Utilizado

Java es un lenguaje de programación y una plataforma informática que fue comercializada por primera vez en 1995 por Sun Microsystems. Hay muchas aplicaciones y sitios web que no funcionarán, probablemente, a menos que tengan Java instalado, y cada día se crean más. Java es rápido, seguro y fiable. Desde ordenadores portátiles hasta centros de datos, desde consolas para juegos hasta computadoras avanzadas, desde teléfonos móviles hasta Internet, Java está en todas partes. Si es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos diez millones de usuarios reportados.

Código

Clases

El programa cuenta con tres packages en los que se encuentran analizadores que contiene principalmente el analizador léxico y sintáctico con la ayuda de jlex y cup; el principal que contiene el main y la interfaz gráfica; Structures que contiene el árbol, nodo, producciones y otro package que cuenta con todas las instrucciones que básicamente son las traducciones.



Interfaz gráfica

La interfaz gráfica es lo primero que se ejecuta en la aplicación, la cual es realizada con poo.

```
1 usage
public Principal() {
    this.setTitle("Proyecto 1 OLC1");
    this.setBounds( x: 50,  y: 50,  width: 1300,  height: 750);
    this.setLocationRelativeTo(null);
    this.getContentPane().setBackground(new Color( r: 174,  g: 214,  b: 241));
    this.setLayout(null);

    JLabel titulo = new JLabel();
    titulo.setText("Pseudo-Parser");
    titulo.setBounds( x: 500,  y: 10,  width: 500,  height: 35);
    titulo.setFont(new Font( name: "Century Gothic",  style: 1,  size: 35));
    this.add(titulo);

    JLabel tituloEntrada = new JLabel();
    tituloEntrada.setText("Archivo de entrada");
    tituloEntrada.setBounds( x: 10,  y: 100,  width: 200,  height: 35);
    tituloEntrada.setFont(new Font( name: "Century Gothic",  style: 1,  size: 15));
    this.add(tituloEntrada);


    panelEntrada = new JPanel();
    panelEntrada.setLayout(null);
    panelEntrada.setBounds( x: 10,  y: 140,  width: 700, height: 550);

    entrada=new JTextArea();
    entrada.setBounds( x: 0, y: 0, width: 700, height: 550);
    entrada.setFont(new Font( name: "Century Gothic",  style: 1,  size: 12));
    panelEntrada.add(entrada);

    JScrollPane scrollEntrada = new JScrollPane(entrada, JScrollPane.VERTICAL_SCROLLBAR_ALWAYS, JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
    scrollEntrada.setBounds( x: 0,  y: 0,  width: 700, height: 550);
    panelEntrada.add(scrollEntrada);
    this.add(panelEntrada);
```

```
btnLimpiar = new JButton();
btnLimpiar.setText("Limpiar");
btnLimpiar.setBounds( x: 800,  y: 60,  width: 150,  height: 35);
btnLimpiar.setFont(new Font( name: "Century Gothic",  style: 1,  size: 15));
btnLimpiar.addActionListener( l: this);
this.add(btnLimpiar);

btnCompilar = new JButton();
btnCompilar.setText("Compilar");
btnCompilar.setBounds( x: 1000,  y: 60,  width: 150,  height: 35);
btnCompilar.setFont(new Font( name: "Century Gothic",  style: 1,  size: 15));
btnCompilar.addActionListener( l: this);
this.add(btnCompilar);


panelGolang = new JPanel();
panelGolang.setLayout(null);
panelGolang.setBounds( x: 750, y: 140, width: 500, height: 250);

salidaGolang=new JTextArea();
salidaGolang.setBounds( x: 0, y: 0, width: 500, height: 250);
salidaGolang.setFont(new Font( name: "Century Gothic",  style: 1,  size: 12));
salidaGolang.setEditable(false);
panelGolang.add(salidaGolang);

JScrollPane scrollGolang = new JScrollPane(salidaGolang, JScrollPane.VERTICAL_SCROLLBAR_ALWAYS, JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
scrollGolang.setBounds( x: 0,  y: 0,  width: 500,  height: 250);
panelGolang.add(scrollGolang);
this.add(panelGolang);

JLabel tituloSalidaGolang = new JLabel();
tituloSalidaGolang.setText("Salida Golang");
tituloSalidaGolang.setBounds( x: 800,  y: 100,  width: 200,  height: 35);
tituloSalidaGolang.setFont(new Font( name: "Century Gothic",  style: 1,  size: 15));
this.add(tituloSalidaGolang);
```

Main

Contiene varios métodos y funciones entre ellos analizar que analiza la entrada, ejecutarAST que retorna la traducción de Python y Golang, guardar Archivo que guarda el cuadro de texto de Python y guardarArchivoGo que guarda el de Golang, errores genera html con los erros y EscribirArchivo escribe el archivo.

```
    public static void main(String[] args) {
        Principal interfaz = new Principal();
        interfaz.setVisible(true);
    }

    1 usage
    public static String analizar(){...}



    public Arbol getArbol() {
        return this.arbol;
    }

    1 usage
    public static String ejecutarAST(LinkedList<Instruccion> ast) {...}

    1 usage
    public static void guardarArchivo(String texto, String nombreArchivo) {...}

    1 usage
    public static void guardarArchivoGo(String texto, String nombreArchivo) {...}

    1 usage
    public static void Errores(){...}

    1 usage
    public static void EscribirArchivo(String contenido, String ruta){...}
}
```

léxico

En Léxico del package Analizadores se encuentra el archivo de jlex con el análisis léxico que contiene las expresiones regulares, palabras reservadas y retorna un new Symbol que posteriormente será utilizado en el análisis sintáctico.

```
package analizadores;
import java_cup.runtime.Symbol;

%%
%class Lexico
%public
%line
%char
%cup
%unicode
%ignorecase

%init{
    yyline = 1;
    yychar = 1;
%init}

BLANCOS=[ \r\t]+
entero=[0-9]+
decimal=[0-9]+("."[0-9]+)?
valorcadena=([\"][^\n\"]+[\"])
valorcaracter=[\'][A-Za-z][\']
caracter_ascii=("'${"(((6[5-9])|([7-8][0-9])|(90))|((9[7-9])|(1[0-1][0-9])|(12[0-2]))|(164)|(165)|(32))"}'")
comentario_linea=[\/][\/][^\n]+
comentario_multilinea=\/\*[^"\\/"]+\*\/
variable="_"[A-Za-z0-9]+"_"
asignacionSimbolo="->"

%%
```

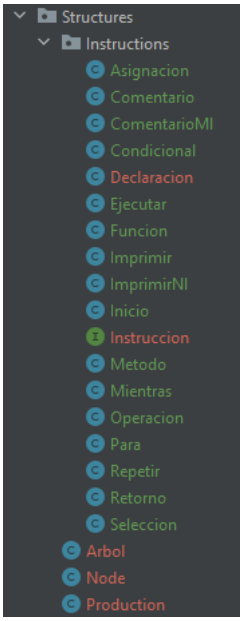
Sintáctico

En Sintactico del package Analizadores se encuentra el archivo de cup que contiene la gramática libre de contexto que a su vez genera el AST con todos los terminales, no terminales, producciones y símbolo inicial.

```
INICIO::= MAIN:a {:System.out.println("Fin del análisis");
                      parser.AST = a.getInstructions();
                      Node NODERAIZ = new Node("INIT");
                      NODERAIZ.addSon(a.getNode());
                      parser.arbol=new Arbol(NODERAIZ);
                    :}
;

MAIN::= inicio INSTRUCCIONES:a fin {:RESULT=new Production(new Node("MAIN"), new Inicio(a.getInstructions()));
                                   :}
;

INSTRUCCIONES::= INSTRUCCIONES:b INSTRUCCION:a  {:
                                        RESULT=new Production(new Node("Instrucciones"), a.getInstructions());
                                        RESULT.addIns(b.getIns());
                                        RESULT.addSon(a);
                                        RESULT.addSon(b);
                                      :}
| INSTRUCCION:a {:
               RESULT=new Production(new Node("Instrucciones"), new LinkedList<>());
               RESULT.addIns(a.getIns());
               RESULT.addSon(a);

               :}
;
```

Structures

En el package Structures se encuentra todas las Instrucciones que realiza el programa (traducciones) así como el Árbol, nodo y Production.



Repositorio: https://github.com/Allenrovas/OLC1-202004745