UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
SEGUNDO SEMESTRE 202

C

MANUAL TÉCNICO
PROYECTO 1

Nombre: Angel Francisco Sique Santos
Carné: 202012039

INTRODUCCIÓN

Un cliente ha solicitado a usted un Pseudo-Parser para que el nuevo personal que no conoce los lenguajes de Python y Golang, pueda aplicar sus conocimientos en pseudocódigo y utilizando una aplicación, traducir este código y ver cómo se comportan las diferentes sintaxis de cada uno de los lenguajes ya que para cada uno existen diferentes características, por lo cual, se le solicita a usted que a partir de sus conocimientos en compiladores haga una implementación de las primeras 2 fases de un compilador y ejecute una traducción con la entrada de pseudocódigo a Python y Golang. Además que se pueda visualizar el diagrama de flujo resultante de la entrada, para esto debe utilizar las herramientas de JFLEX y CUP e implementar su solución en el lenguaje de programación JAVA.

REQUISITOS MÍNIMOS

500mb de disco duro

2gb de RAM

Windows 7

Equipo Intel Pentium o superior

Herramientas de análisis léxico y sintáctico: JFlex/CUP

Java 14

Graphviz

OPCIONES DEL PROGRAMA

1. En esta parte creamos la interfaz gráfica, en este caso se hizo uso de la herramienta de drag and drop incuida en netbeans para realizarla. Esta herramienta crea los componentes de manera más sencilla e intuitiva.

```
    public static void main(String args[]) {…

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton Clean;
    private javax.swing.JMenuItem ErrorsOption;
    private javax.swing.JButton GolangView;
    private javax.swing.JMenuItem OpenFile;
    private javax.swing.JButton PythonView;
    private javax.swing.JButton Run;
    private javax.swing.JMenuItem SaveFile;
    private javax.swing.JTextArea TextArea;
    private javax.swing.JLabel errorNumber;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JMenu jMenu1;
    private javax.swing.JMenu jMenu2;
    private javax.swing.JMenu jMenu3;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JMenuItem jMenuItem3;
    private javax.swing.JMenuItem jMenuItem4;
    private javax.swing.JMenuItem jMenuItem5;
    private javax.swing.JScrollPane jScrollPane1;
    // End of variables declaration//GEN-END:variables
}
```

2. En el botón Run leemos el archivo de entrada a traves de un BufferReader que obtiene todo el texto del TextArea y lo envía al anaalizador para obtener los datos, luego se envia la petición para que el analizador analice el archivo, creamos el AST además del diagrama de flujo del código ingresado y también creamos un HTML con los errores sintáctocos y léxicos encontrados.

```
public class Ventana extends javax.swing.JFrame {
    private void RunActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:eve
        // TODO add your handling code here:    TODO add your handling code here:
        try {
            String dato= TextArea.getText();
            System.out.println(dato);
            Analizador_Lexico lexico = new Analizador_Lexico(new BufferedReader(new
            Analizador_sintactico sintactico = new Analizador_sintactico(lexico);
            sintactico.parse();
            errorNumber.setText(Integer.toString(sintactico.erroresSintacticos));

            Nodo raiz = sintactico.padre;    The static field Analizador_sintactico.
            //Crear flowchart
            getInicio(raiz);
            diamond = getIF(raiz);
            rectangle = getProcedimientos(raiz);
            parallelogram = getVariables(raiz);
            invhouse = getSalidas(raiz);
            crearFlowchart(recorrido(raiz));
            //Recorreo raiz
            crearAST(recorrido(raiz));

            //Crar HTML de errores
            String htmlstyle = "<!DOCTYPE html>"+
            "<html>"+

                "<head>" +
                "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=iso-
```

3. El botón clean limpia todo el cuadro de texto y lo remplaza por una String vacia.

```
    private void CleanActionPerformed(java.awt.event.ActionEvent evt) {//
        // TODO add your handling code here:    TODO add your handling co
        TextArea.setText("");

    }//GEN-LAST:event_CleanActionPerformed
```

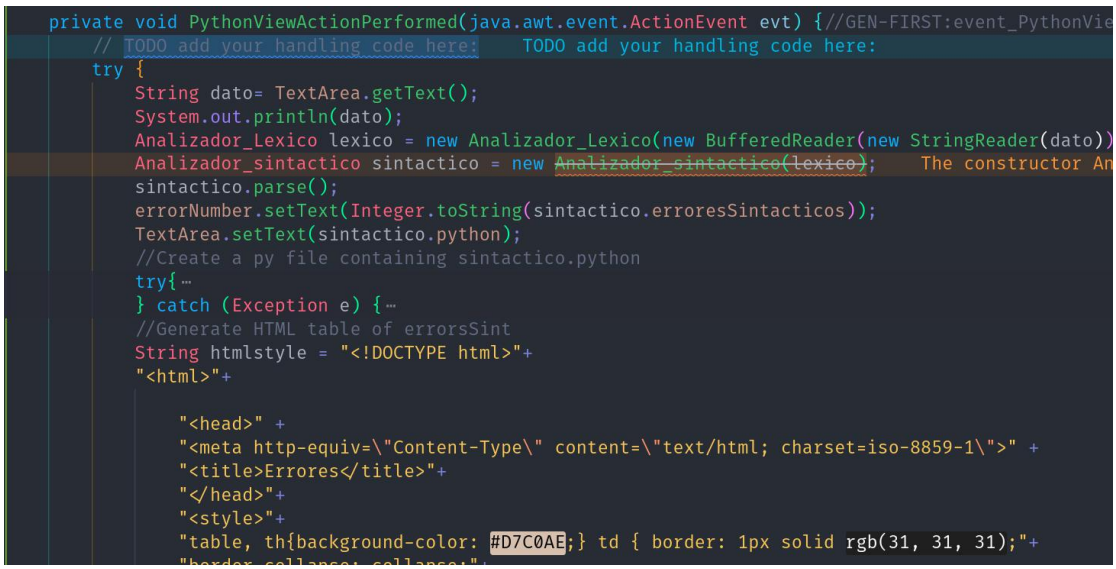
4. La opción Open File crea un FikeChooser con un filtro para que solo muestre los archivos con extension .olc luego obtiene el texto del archivo y lo muestra en el TextArea. La opción Save File guarda el contenido del cuadro de texto en un archivo con el nombre que el usuario elija.

```
private void OpenFileActionPerformed(java.awt.event.ActionEvent evt) {//GE
    // TODO add your handling code here:    TODO add your handling code he
    //Open only .olc files
    FileNameExtensionFilter filter = new FileNameExtensionFilter("OLC File

    //Clean TextArea
    TextArea.setText("");

    //Open File
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setFileFilter(filter);
    fileChooser.showOpenDialog(this);
    File file = fileChooser.getSelectedFile();
    if (file != null) {
        try {
            BufferedReader br = new BufferedReader(new FileReader(file));
            String linea;
            while ((linea = br.readLine()) != null) {…
            br.close();
        } catch (Exception e) {…
    }

}//GEN-LAST:event_OpenFileActionPerformed
```

5. La opción Save File guarda el contenido del cuadro de texto en un archivo con el nombre que el usuario elija.

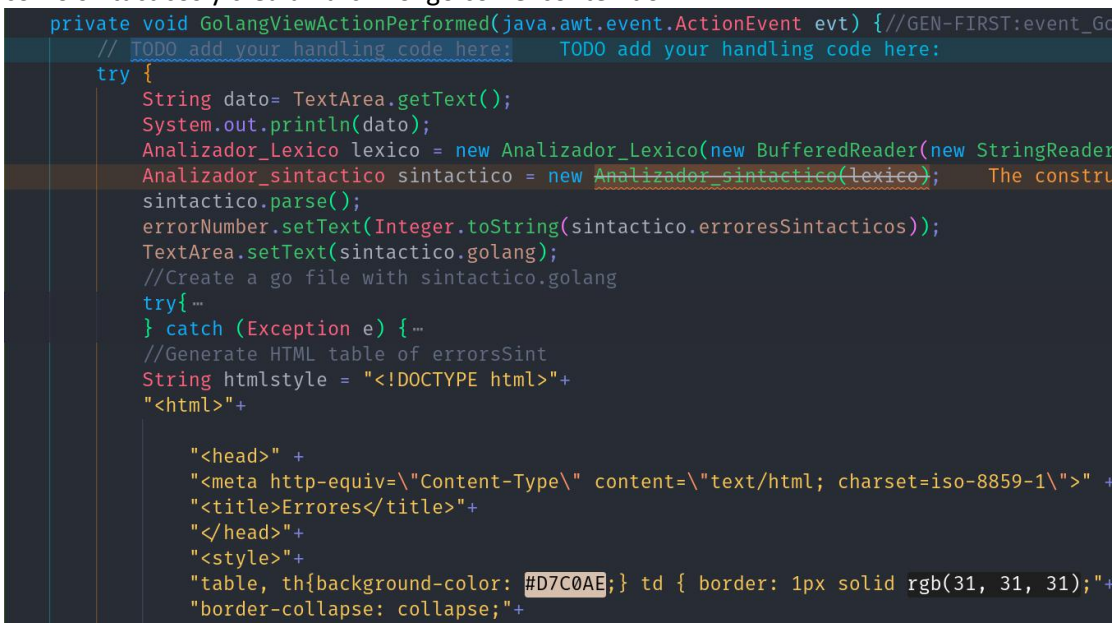
```
    private void SaveFileActionPerformed(java.awt.event.ActionEvent evt) {//GEN
        // TODO add your handling code here:    TODO add your handling code her
        JFileChooser fileChooser = new JFileChooser();
        fileChooser.showSaveDialog(this);
        File file = fileChooser.getSelectedFile();
        if (file != null) {
            try {
                BufferedWriter bw = new BufferedWriter(new FileWriter(file));
                bw.write(TextArea.getText());
                bw.close();
            } catch (Exception e) {
                System.out.println(e);
            }
        }
```

6. La opción View Python Code traduce el código ingresado a código Python y lo muestra en el cuadro de texto. Además de que crea el HTML con los errores encontrados, tanto léxicos como sintácticos y crea un archivo .py con el código.



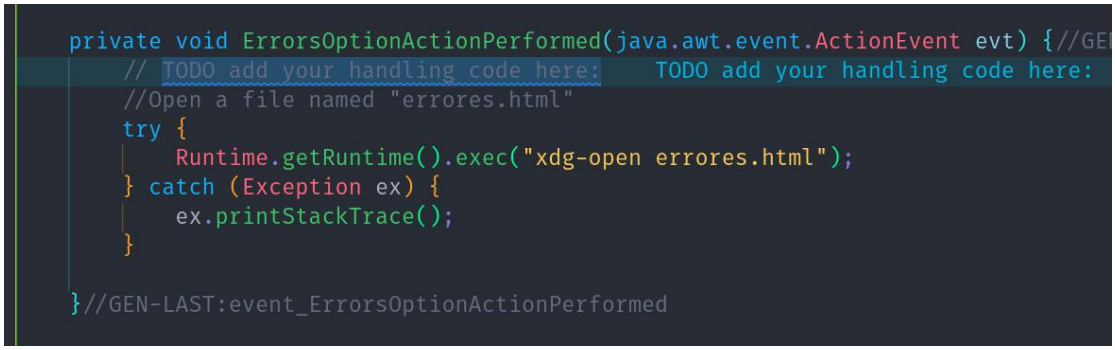
```
private void PythonViewActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_PythonVie
    // TODO add your handling code here:    TODO add your handling code here:
    try {
        String dato= TextArea.getText();
        System.out.println(dato);
        Analizador_Lexico lexico = new Analizador_Lexico(new BufferedReader(new StringReader(dato))
        Analizador_sintactico sintactico = new Analizador_sintactico(lexico);    The constructor An
        sintactico.parse();
        errorNumber.setText(Integer.toString(sintactico.erroresSintacticos));
        TextArea.setText(sintactico.python);
        //Create a py file containing sintactico.python
        try{…
        } catch (Exception e) {…
        //Generate HTML table of errorsSint
        String htmlstyle = "<!DOCTYPE html>"+
        "<html>"+

            "<head>" +
            "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=iso-8859-1\">" +
            "<title>Errores</title>"+
            "</head>"+
            "<style>"+
            "table, th{background-color: #D7C0AE;} td { border: 1px solid rgb(31, 31, 31);"+
```

7. La opción View Golang Code traduce el código ingresado a código Golangos, tanto léxicos como sintácticos y crea un archivo .go con el contenido.



```
private void GolangViewActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_Go
    // TODO add your handling code here:    TODO add your handling code here:
    try {
        String dato= TextArea.getText();
        System.out.println(dato);
        Analizador_Lexico lexico = new Analizador_Lexico(new BufferedReader(new StringReader
        Analizador_sintactico sintactico = new Analizador_sintactico(lexico);    The constru
        sintactico.parse();
        errorNumber.setText(Integer.toString(sintactico.erroresSintacticos));
        TextArea.setText(sintactico.golang);
        //Create a go file with sintactico.golang
        try{…
        } catch (Exception e) {…
        //Generate HTML table of errorsSint
        String htmlstyle = "<!DOCTYPE html>"+
        "<html>"+

            "<head>" +
            "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=iso-8859-1\">" +
            "<title>Errores</title>"+
            "</head>"+
            "<style>"+
            "table, th{background-color: #D7C0AE;} td { border: 1px solid rgb(31, 31, 31);"+
            "border-collapse: collapse;"+
```

8. La opción Errors abre el HTML generado con la lista de todos los erroes léxicos y sintácticos.



```
private void ErrorsOptionActionPerformed(java.awt.event.ActionEvent evt) {//GE
    // TODO add your handling code here:    TODO add your handling code here:
    //Open a file named "errores.html"
    try {
        Runtime.getRuntime().exec("xdg-open errores.html");
    } catch (Exception ex) {
        ex.printStackTrace();
    }

}//GEN-LAST:event_ErrorsOptionActionPerformed
```