UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERIA

ESCUELA DE SISTEMAS

ORGANIZACIÓN DE LENGUAJES Y COMPILADORES 1

Ing. Kevin Lajpop

Aux. Moises Gonzales

PROYECTO 1

Pseudocodigo -> Python/Golang

JOSUE ROLANDO

GRAMAJO ROLDAN

202000895

3021021080101

INDICE

Introducción…….……2

Objetivos……2

Conocimientos previos……...……2

Requerimientos……...……2

Descripción de clases……..……...5

Gramática libre de contexto..……..……...6

Introducción

Un cliente ha solicitado a usted un Pseudo-Parser para que el nuevo personal que no conoce los lenguajes de Python y Golang, se solicita que implemente de las primeras 2 fases de un compilador y ejecute una traducción con la entrada de pseudocódigo a Python y Golang.

Objetivos

Guiar al lector en la manipulación del programa, así mismo dar a conocer la lógica empleada en el desarrollo del programa por si es necesario actualizar, mejorar o dar mantenimiento a la aplicación creada y descrita.

Conocimientos previos

Los conocimientos que deberán tener las personas que manejen el programa son:

- Conocimiento del lenguaje JAVA
- Conocimiento de las bibliotecas Jflex y Jcup
- Conocimientos sobre POO
- Conocimientos sobre GUI con JavaSwing
- Manejo del IDE Netbeans.

Requerimientos mínimos para ejecutar en Neatbens 8.2

- Java 1.8
- Procesador a 1.6 GHz o superior
- 512 MB de RAM
- 750 MB de espacio disponible en el disco duro.
- Tarjeta de vídeo compatible con DirectX 9
- windows 7,8,superior

Descripción de clases

Paquete	Clase	Descripción
analizadores	*A_lexico.jflex*	Esta clase es un archivo 'jflex' en el cual se escriben todos los tokens que requiere el lenguaje además de guardar todos los errores léxicos encontrados.
analizadores	*A_sintactico.cup*	Esta clase es un archivo 'cup' en el cual se escribe toda la gramática para aceptar los tokens que requiere el lenguaje, además de ser la clase que utiliza nuestro árbol AST para generasrse.
arbol	*Nodo.java*	Esta clase es un objeto de tipo Nodo que recibe parámetros de tipo token.
arbol	*AST.java*	Esta clase es la generadora del árbol AST y recibe objetos de tipo Nodo, escribe y guarda el archivo '.dot' que utiliza Graphviz para dibujar el árbol.
arbol	*Declarar.java*	Esta clase se llama desde el archivo cup y le envía parámetros de tipo nodo para guardar en memoria la traducción de las **declaraciones** a Python y Golang
arbol	*Asingar.java*	Esta clase se llama desde el archivo cup y le envía parámetros de tipo nodo para guardar en memoria la traducción de las **asignciones** a Python y Golang

arbol	*If.java*	Esta clase se llama desde el archivo cup y le envía parámetros de tipo nodo para guardar en memoria la traducción de las instrucciones **condicionales** a Python y Golang
arbol	*Para.java*	Esta clase se llama desde el archivo cup y le envía parámetros de tipo nodo para guardar en memoria la traducción de la instrucción **cíclica 'For'** a Python y Golang
arbol	*Imprimir.java*	Esta clase se llama desde el archivo cup y le envía parámetros de tipo nodo para guardar en memoria la traducción de **impresión en consola** a Python y Golang
arbol	*Errores.java*	Esta clase se llama desde el archivo jflex y añade objetos de tipo Error léxico en la lista de errores léxicos general.

GRAMÁTICA LIBRE DE CONTEXTO

PROYECTO 1 OLC1 - 202000895

INICIO::= res_INICIO RECURSIVE res_FIN

RECURSIVE::= BEGIN RECURSIVE

| BEGIN

BEGIN::= DECLARA

| ASIGNA

| CONDICION

| SWITCH

| FOR

| WHILE

| HASTA

| METODO

| FUNCION

| PRINT

| error tk_PTCOMA

DECLARA::= res_INGRESAR LISTID res_COMO TIPO res_CONVALOR EXPRESION_NUM tk_PTCOMA

ASIGNA::= LISTID tk_ASIGNAFLECHA EXPRESION_NUM tk_PTCOMA

//---->CONDICIONES<----//

CONDICION::= res_SI EXPRESION_logica res_ENTONCES RECURSIVE_INST res_FINSI

| res_SI EXPRESION_logica res_ENTONCES RECURSIVE_INST res_ELSE RECURSIVE_INST res_FINSI

| res_SI EXPRESION_logica res_ENTONCES RECURSIVE_INST ELSEIF res_FINSI

| res_SI EXPRESION_logica res_ENTONCES RECURSIVE_INST ELSEIF res_ELSE RECURSIVE_INST res_FINSI

ELSEIF::= ELSEIF res_ELSEIF EXPRESION_logica res_ENTONCES RECURSIVE_INST

| res_ELSEIF EXPRESION_logica res_ENTONCES RECURSIVE_INST

//---->CICLOS<----//

FOR::= res_PARA LISTID tk_ASIGNAFLECHA EXPRESION_NUM res_HASTA EXPRESION_NUM res_HACER RECURSIVE_INST res_FINPARA

| res_PARA ASIGNA res_HASTA EXPRESION_NUM res_CONVALOR RECURSIVE_INST res_FINPARA

WHILE::= res_MIENTRAS EXPRESION_logica res_HACER RECURSIVE_INST res_FINMIENTRAS

HASTA::= res_REPETIR RECURSIVE_INST res_HASTAQ EXPRESION_logica

SWITCH::= res_SEGUN EXPRESION_NUM res_HACER SWITCH_ res_FINSEGUN

SWITCH_::= SWITCH_:a tk_ASKA:b EXPRESION_NUM:c tk_ASKC:d res_ENTONCES:e RECURSIVE_INST:f

//--->METODOS<---//

METODO::= res_METODO IDENTIFICADOR RECURSIVE_INST res_FINMETODO

| res_METODO IDENTIFICADOR res_METODOCONP tk_PARIZQ LISTP tk_PARDER RECURSIVE_INST res_FINMETODO

//--->FUNCION<---//

FUNCION::= res_FUNCION IDENTIFICADOR TIPO RECURSIVE_INST res_FINFUNCION

| res_METODO IDENTIFICADOR res_METODOCONP tk_PARIZQ LISTP tk_PARDER RECURSIVE_INST res_FINFUNCION

//--->EJECUTAR<---//

EJECUTAR::= res_EJECUTAR IDENTIFICADOR tk_PARIZQ tk_PARDER tk_PTCOMA

| res_EJECUTAR IDENTIFICADOR tk_PARIZQ LISTP tk_PARDER tk_PTCOMA

//--->PRINT<---//

PRINT::= res_IMPRIMIR tk_CADENA tk_PTCOMA

| res_IMPRIMIRNL tk_CADENA tk_PTCOMA

// ---------[Utilidades]-------- //

TIPO::= res_CADENA

| res_NUMERO

| res_BOOLEAN

LISTID::= LISTID tk_COMA IDENTIFICADOR

| IDENTIFICADOR

LISTP::= LISTP tk_COMA IDENTIFICADOR TIPO

| IDENTIFICADOR TIPO

EXPRESION_NUM ::= EXPRESION_NUM tk_MAS EXPRESION_NUM

| EXPRESION_NUM tk_MENOS EXPRESION_NUM

| EXPRESION_NUM tk_POR EXPRESION_NUM

| EXPRESION_NUM tk_DIVIDIDO EXPRESION_NUM

| EXPRESION_NUM res_POTENCIA EXPRESION_NUM

| EXPRESION_NUM res_MOD EXPRESION_NUM

| tk_PARIZQ EXPRESION_NUM tk_PARDER

| ENTERO

| DECIMAL

| IDENTIFICADOR

| tk_CADENA

| res_VERDADERO

| res_FALSO

RECURSIVE_INST::= INSTRUCCIONES RECURSIVE_INST

| INSTRUCCIONES

INSTRUCCIONES::= DECLARA

| ASIGNA

| CONDICION

| SWITCH

| FOR

| WHILE

| HASTA

| PRINT

EXPRESION_logica::= EXPRESION_NUM res_MAYOR EXPRESION_NUM

| EXPRESION_NUM res_MENOR EXPRESION_NUM

| EXPRESION_NUM res_MENORIGUAL EXPRESION_NUM

| EXPRESION_NUM res_MAYORIGUAL EXPRESION_NUM

| EXPRESION_NUM res_ESIGUAL EXPRESION_NUM

| EXPRESION_NUM res_ESDIFERENTE EXPRESION_NUM

| EXPRESION_NUM res_AND EXPRESION_NUM

| EXPRESION_NUM res_OR EXPRESION_NUM

| EXPRESION_NUM res_NOT EXPRESION_NUM