UNIVERSIDAD SAN CARLOS DE GUATEMALA
CENTRO UNIVERSITARIO DE OCCIDENTE
DIVISIÓN DE CIENCIAS DE LA INGENIERÍA
Ingeniería en Ciencias y Sistemas.
Ing. Christian Quiroa
Auxiliar: Julio Fernando Ixcoy

Byron Fernando Torres Ajxup 201731523

Manual Técnico Analizador Sintactico

Parser-py

Versión 1.1

Quetzaltenango 16 de Octubre del 20223

Índice

Introducción

Este manual técnico describe el analizador sintáctico realizado por [tu nombre]. El analizador sintáctico es una herramienta que se utiliza para analizar la estructura sintáctica de un lenguaje. En este caso, el analizador sintáctico está diseñado para analizar el lenguaje de programación Python.

Requisitos

El analizador sintáctico tiene los siguientes requisitos:

- Soporta el lenguaje de programación Python.
- Realiza un análisis sintáctico descendente.
- Reconoce los siguientes tipos de expresiones:
- Variables
- Operadores
- Funciones
- Condicionales
- Bucles

Diseño

El analizador sintáctico está diseñado utilizando un enfoque propio, partiendo de la realización utilizando la lógica propia sin enfoques predefinidos puesto que ya se tenia el análisis léxico, el sintáctico solo se encargó de llamar token por token para analizar si estaban en el orden adecuado, utilice una clase para el análisis y dividir en varios métodos para una mejor lectura de código, cada acción de bloque realizaba operaciones diferente por ejemplo para verificar la estructura de asignación, ciclo for, sentencia if, etc.

El analizador utiliza las siguientes estructuras de datos

- Pila: Para almacenar los nodos del árbol sintáctico que aún no se han analizado(ArrayList).
- Tabla de símbolos: Para almacenar las variables y funciones definidas en el programa (Hashmap).

Implementación

Implementación
El analizador sintáctico está implementado utilizando el lenguaje de programación Java. El analizador se divide en una sola clase llamada Sintaxis.
La clase Sintaxis implementa el analizador sintáctico descendente. La clase utiliza las siguientes estructuras de datos:
ArrayList: Para almacenar los tokens en el analizador léxico.
HashMap: Para almacenar las variables y funciones definidas en el programa.
La clase Sintaxis tiene varios métodos que se encargan de verificar la posición de cada token para el análisis sintáctico.
A continuación, se describen los métodos más importantes de la clase Sintaxis:
**public void analiza(): Este método es el punto de entrada del analizador sintáctico. El método analiza el primer token del ArrayList y llama a los métodos apropiados para analizar el resto del programa.
**public void idAnalyzer(): Este método analiza una expresión. El método utiliza la tabla de símbolos para verificar que las variables y funciones utilizadas en la expresión sean válidas.
**public void opAsignacion(): Este método analiza una asignación. El método utiliza la tabla de símbolos para asignar el valor de la expresión a la variable.
**public void sentenciIF(): Este método analiza una instrucción condicional. El método utiliza la tabla de símbolos para verificar que las variables utilizadas en la condición sean válidas.
**public void bubleFor(): Este método analiza una instrucción de bucle. El método utiliza la tabla de símbolos para verificar que las variables utilizadas en el bucle sean válidas.
Entre estas hay más métodos que analizan la posición del token en durante el análisis.

Pruebas

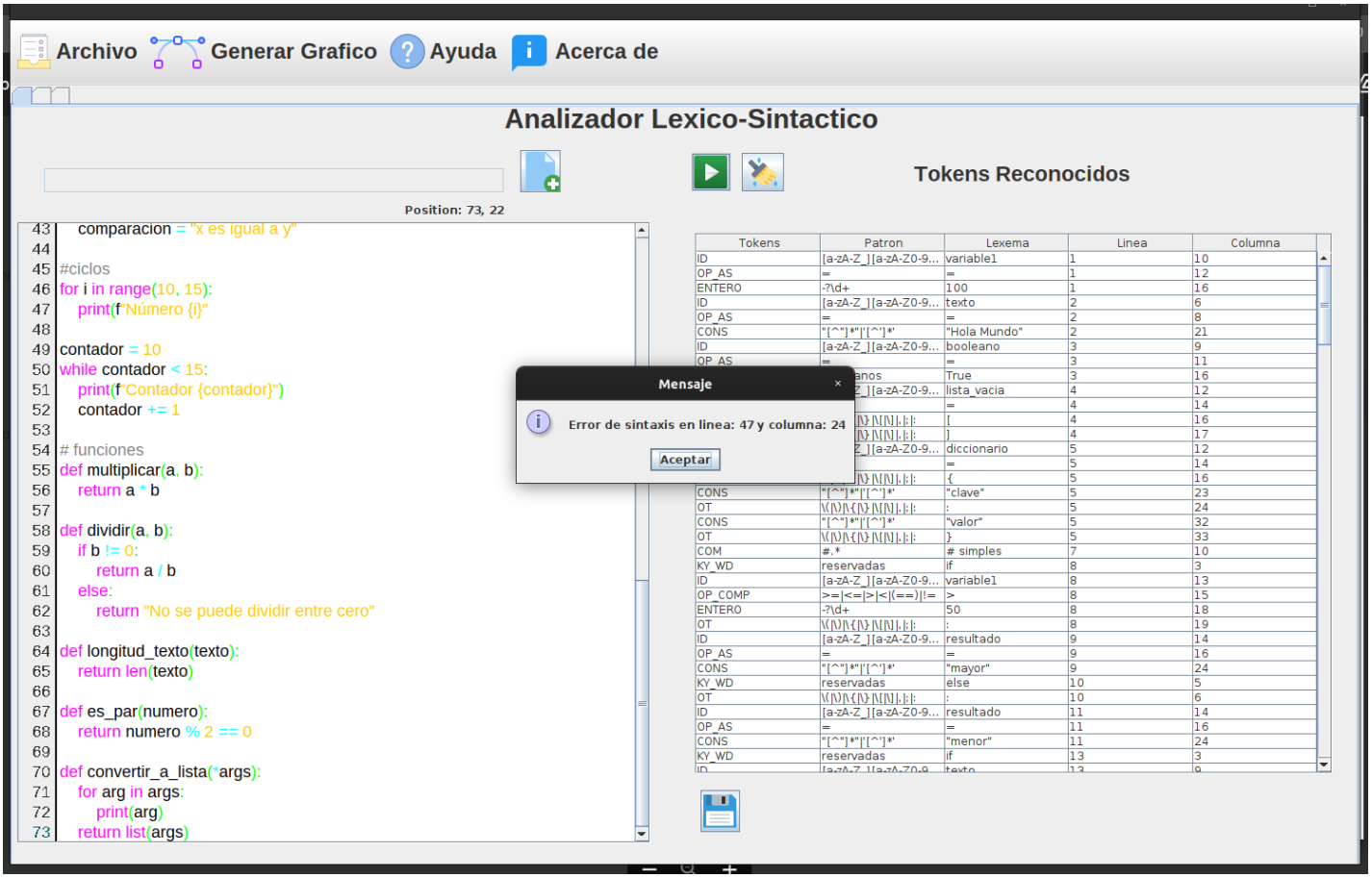
El analizador sintáctico se ha probado con una serie de casos de prueba. Los casos de prueba incluyen expresiones simples y complejas, así como funciones y condicionales.

Excepciones y errores

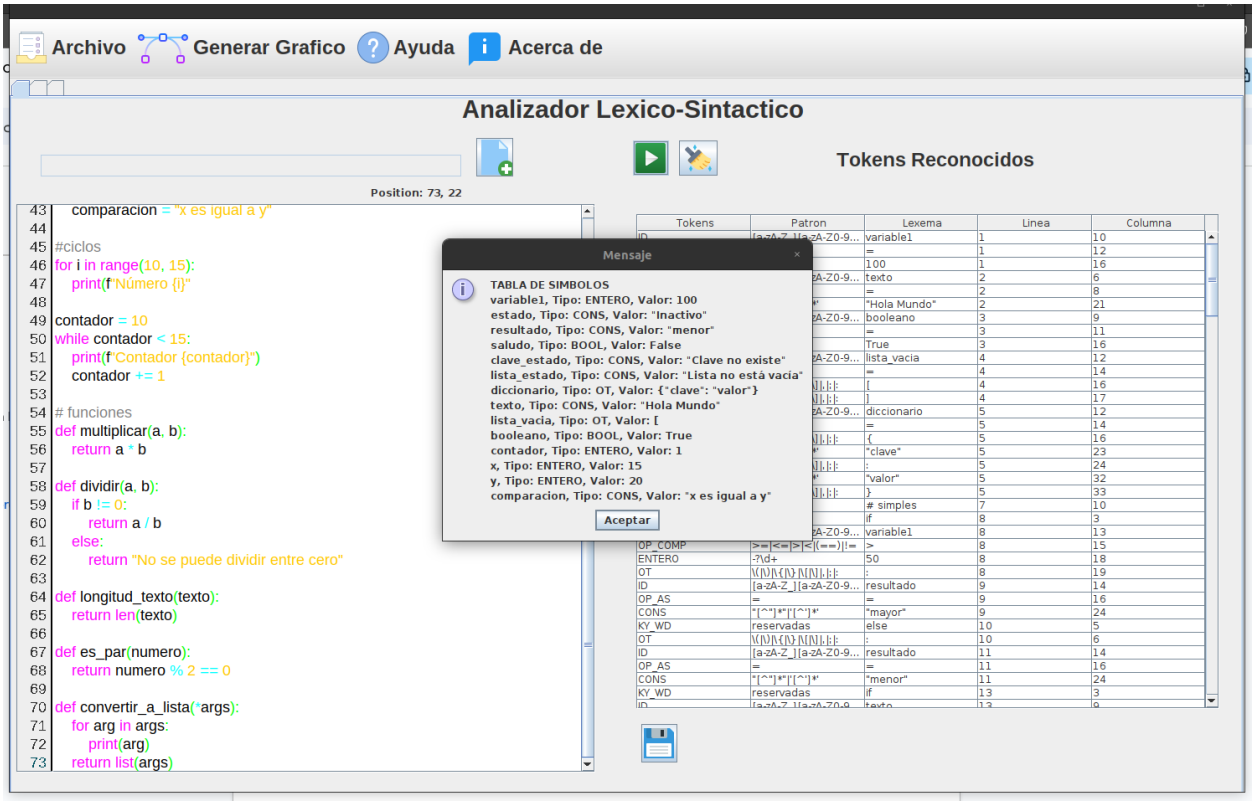
El analizador sintáctico puede generar las siguientes excepciones y errores:

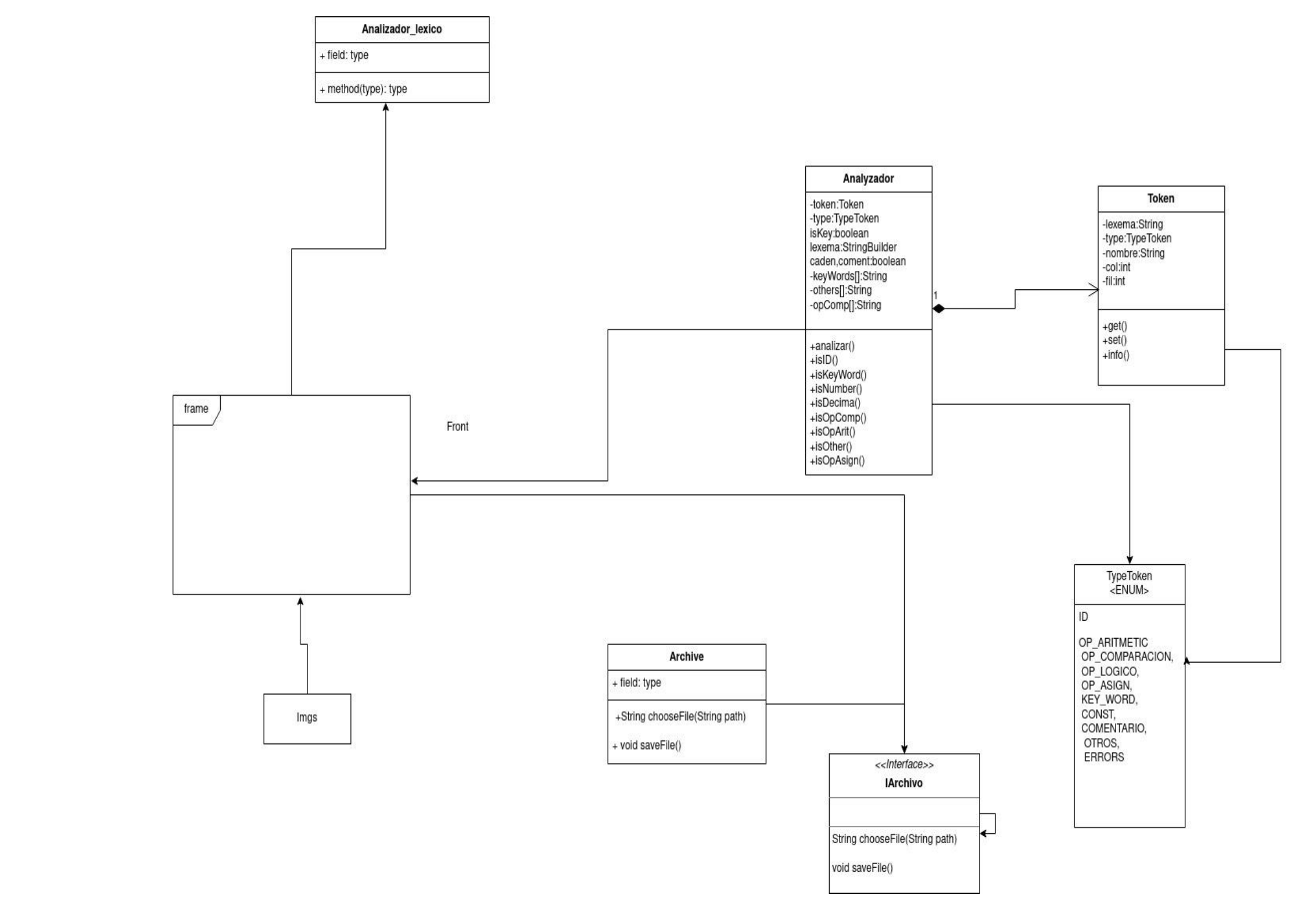
Error de sintaxis: Se genera cuando el analizador encuentra un error en la sintaxis del programa.
Error Léxico: Se genera cuando el analizador no reconoce los tokens previamente personalizados usando la librería regex de java.

Imagen de un error de sintaxis en la línea y columna provocado



Ejemplo de tabla de símbolos




Analizador_lexico
+ field: type
+ method(type): type
Analyzador
-token:Token
-type:TypeToken
isKey:boolean
lexema:StringBuilder
caden,coment:boolean
-keyWords[]:String
-others[]:String
-opComp[]:String
+analizar()
+isID()
+isKeyWord()
+isNumber()
+isDecima()
+isOpComp()
+isOpArit()
+isOther()
+isOpAsign()
1
Token
-lexema:String
-type:TypeToken
-nombre:String
-col:int
-fil:int
+get()
+set()
+info()
frame
Front
Imgs
Archive
+ field: type
+String chooseFile(String path)
+ void saveFile()
<<Interface>>
IArchivo
String chooseFile(String path)
void saveFile()
TypeToken
<ENUM>
ID
OP_ARITMETIC
OP_COMPARACION,
OP_LOGICO,
OP_ASIGN,
KEY_WORD,
CONST,
COMENTARIO,
OTROS,
ERRORS