Proyecto Programado del Curso IC-5701

**Compiladores e Intérpretes**

**Unidad de Computación**

**Carrera de Ingeniería de Computación.**

**Andrés García y Kenneth Alvarez**

**Sede San Carlos, 24 de Marzo del 2018**

[**Introducción**](#_lzkzccphy66y) **3**

[**Análisis del lenguaje**](#_19osmqiqwxvp) **4**

[**Solución e implementación**](#_qx0nplv26jl4) **5**

[**Análisis de Resultados**](#_um03tbtg4ogn) **6**

[Resultados Obtenidos](#_e3pk49m5zobq) 6

[**Conclusiones**](#_k87iag51tt8x) **8**

[**Bibliografia**](#_xwukc4btw9uf) **8**

# Introducción

Las herramientas de compilación son de alta importancia para los desarrolladores debido a que les permite agilizar el proceso en diagnóstico de errores en el código,estas usualmente son utilizadas sin medir el alcance y funcionalidades con las que ellas se deben de regir de manera interna, por lo que es de alta importancia que el estudiante comprenda de qué manera distingue los errores, las reglas sintácticas y gramaticales un programa. De esta manera conocer y graficar los nodos que utiliza un programa en su compilación. Esto va a beneficiar al estudiante no solo a conocer su funcionalidad interna sino que además le brinda la posibilidad de analizar profundamente los algoritmos que utiliza cotidianamente.

El sistema utilizará programación orientada a objetos con librerías de ANTLR4 para el manejo de parser. Además el sistemas permite mostrar los errores en los que se está incurriendo en el código y mostrar el árbol que lo compone.

# Análisis del lenguaje

El proyecto a desarrollar constituye una herramienta en ANTLR4 con el lenguaje de programación Java en el cual se creó un compilador en “Monkey language”. Para su desarrollo es importante denotar que Java en un lenguaje orientado a objetos por lo que se cuenta con una vista agradable para el usuario la cual cuenta con una pantalla para ingresar el código y una pantalla donde muestra los errores y árbol del programa a compilar.

La importancia del proyecto radica en que se conozcan los procesos que conllevan al compilar un código y que se visualice desde un ambito mas analitico de los procesos internos de Java.

La estructura del compilador está compuesta por:

**Scanner**: en él se encuentran los símbolos, operadores y palabras reservadas con las que el sistema de Java trabaja. Debe contener los tokens válidos( letras[a-z A-Z \_ ] , dígitos [0-9] y id{ letra|dígito}.Además contiene los caracteres especiales y comentarios(modalidades(// y /\* \*/)).

**Parser:** son las reglas de sintaxis del sistema, estas reglas serán ingresada al sistema para poder medir y mostrar si el recorrido del árbol de ANTLR está siendo recorrido correctamente. Esto desde un conjunto de reglas del programa.

# Solución e implementación

Como herramienta de traducción y generador de código a partir de una sintaxis y un lexer se implementó ANTLR4 G4, el cual permite generar un parser y ejecutarlo de tal manera que facilite el proceso de creación de un compilador y la ejecución de recorrido en árboles.

Para la identificación y notificación de errores se implementó la librería BaseErrorListener la cual por medio de las reglas revisa el código línea por línea para identificar los posibles errores e imprimirlos según los criterios evaluados y sus errores.

Creación de árbol de nodos: el parser.program es una herramienta que genera el árbol de manera automática a partir del parser el cual se le envía por parámetro, luego para mostrar en pantalla se implementa la librería gui.trees.

El editor de texto se generó por medio un editor encontrado en la web el cual fue modificado y adaptado al proyecto actual para que este tuviera la posibilidad de compilar y mejoras generales de la GUI.

# Análisis de Resultados

## Resultados Obtenidos

|  |  |  |
| --- | --- | --- |
| Tarea | Estado | Observaciones |
| ***Interfaz de Usuario*** | | |
| Editor Amigable. | Listo. |  |
| Opciones de Compilación funcionales. | Listo. |  |
| Identificación de código por línea y columna. | Listo. | jTextField abajo del textArea donde se carga/escribe código. |
| Manejo de Archivos. | Listo. | Al cargar/guardar archivo. |
| ***Scanner*** | | |
| Tokens válidos, Comentarios y caracteres especiales | Listo. |  |
| Detección de errores léxicos (Claridad y Ubicación) | Listo. |  |
| ***Parser*** | | |
| Declaraciones. | Listo. |  |
| Statements. | Listo. |  |
| Expresiones (de todos los tipos permitidos aunque no  sean contextualmente correctos). | Listo. |  |
| Detección de errores sintácticos (Claridad y Ubicación) | Listo. |  |
| ***Documentación*** | | |
| Formato y Contenido | Listo. |  |
| Ortografía y Gramática | Listo. | Revisar. ( Según el equipo de trabajo,todo se encuentra bien escrito). |
|  |  |  |

# 

# Conclusiones

Gracias a la herramienta de ANTLR, la creación de un parser para reconocimiento sintáctico de un lenguaje en específico se facilita de gran manera, sin embargo, aun con la simplicidad que ofrece la herramienta, hay que seguir teniendo mucho cuidado con la escritura de las reglas del parser.g4, verificando que su forma de ejecución sea la deseada en función del lenguaje que se desea desarrollar.

# Bibliografia

Fila y Columna del JTextArea. (2015, September 24). Retrieved from http://lineadecodigo.com/java/fila-y-columna-del-jtextarea/

How to Use File Choosers. (n.d.). Retrieved from https://docs.oracle.com/javase/tutorial/uiswing/components/filechooser.html

Text Editor in Java with Source Code. (2017, February 20). Retrieved from http://www.genuinecoder.com/text-editor-java-with-source-notepad-html/

Antlr/antlr4. (n.d.). Retrieved from https://github.com/antlr/antlr4/blob/master/doc/index.md