Instituto Tecnológico de Costa Rica

Curso: Compiladores e intérpretes

Profesor: Ing. Allan Rodriguez

Estudiante:	Carnet:
Jordano Escalante López	2018161994
Fredd Badilla Viquez	2022012800

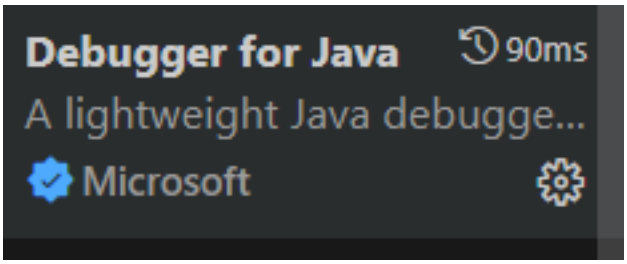
Primer Semestre

Año: 2024

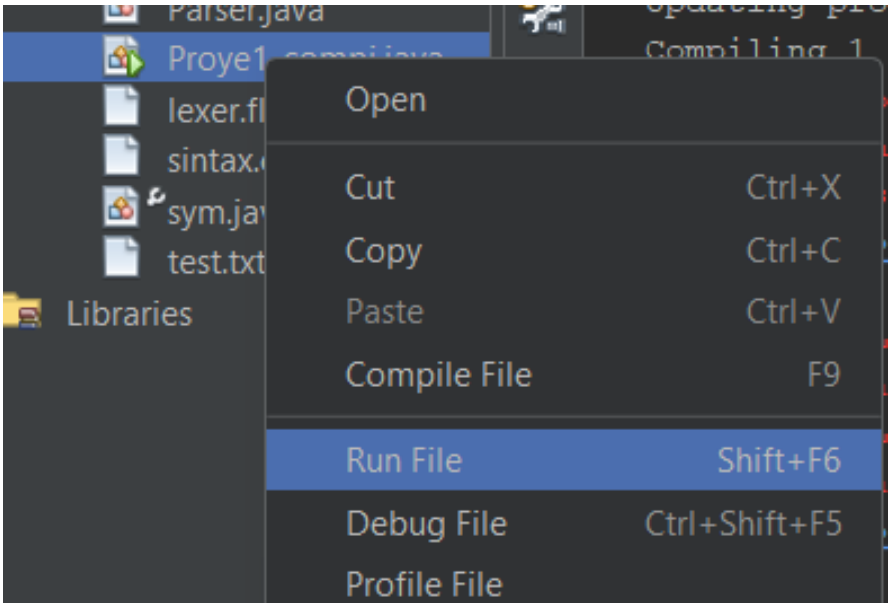
Manual de Usuario

El programa es un programa de Java simple, para asegurarse de poder correrlo asegúrese de contar con el JDK de java en su equipo, puede instalarlo desde la página oficial.

Para ejecutarlo puede hacerlo desde netbeans o VisualStudioCode con la extensión para java de Microsoft que se indica a continuación.

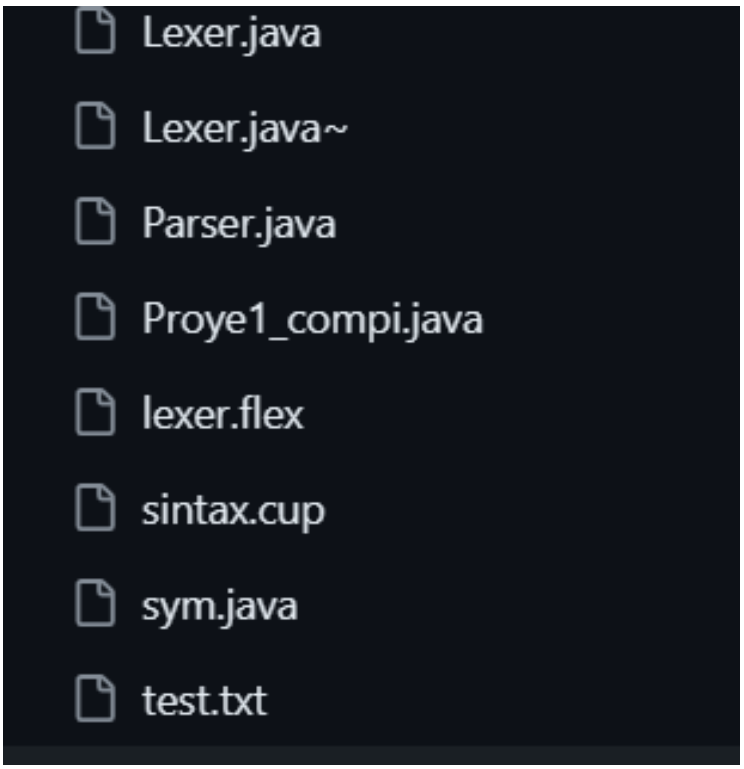


Si cuenta con la extensión y el jdk debería ser capaz de ejecutar el programa, para ello presione click derecho sobre la clase main del proyecto, en este caso se llama Proye1_Compi.java, al dar click derecho, seleccione la opción “Run File” tal como se muestra en la imagen.



Al hacer esto, el programa creará primero el archivo Lexer.java y luego los archivos Parser.java y sym.java.

En la imagen siguiente se pueden ver los archivos mencionados de primero, tercero y séptimo lugar.

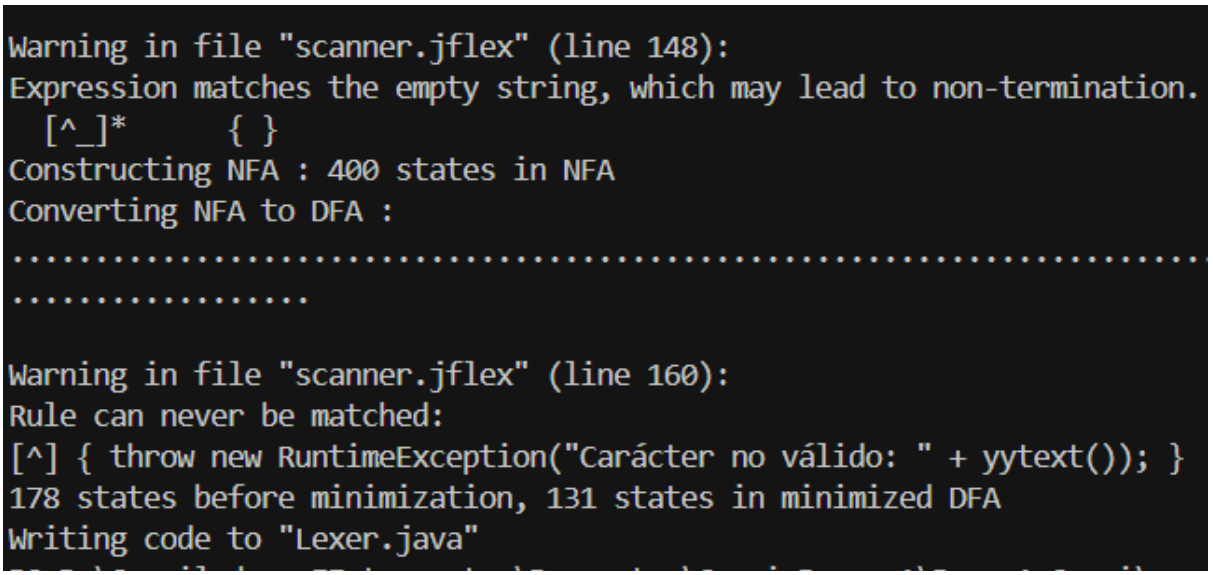


Dependiendo de la máquina en la que se ejecute este proceso puede verse entorpecido si se intenta crear el Parser.java antes de que se complete la creación del Lexer.java, si esto le llega a suceder, vuelva a correr el programa una segunda vez y esto debería solucionar el problema.

Si el proceso se completa con éxito, el programa realizará en análisis del codigo fuente en el archivo test.txt y en el documento TOKENS.txt podrá la lista de los tokens que se encuentren en el archivo test.txt y adicionalmente se van a imprimir en pantalla las tablas de símbolos que se generen.

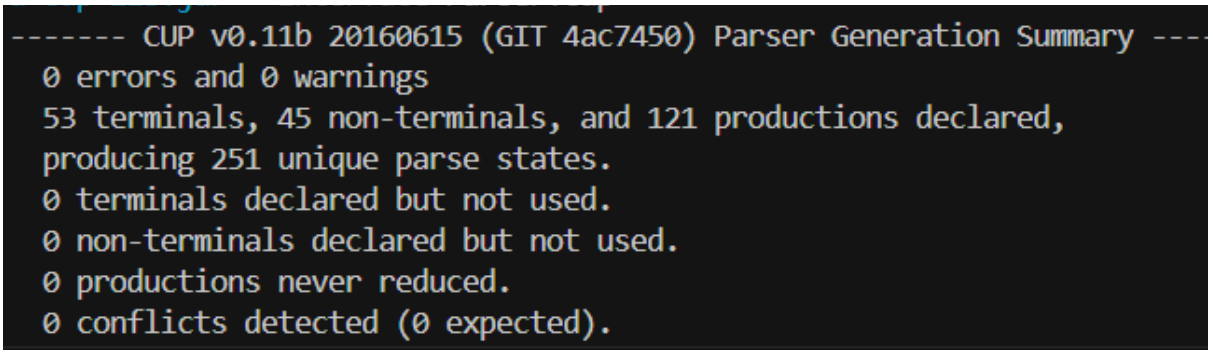
Pruebas de funcionalidad

Proceso de generación del Lexer.java

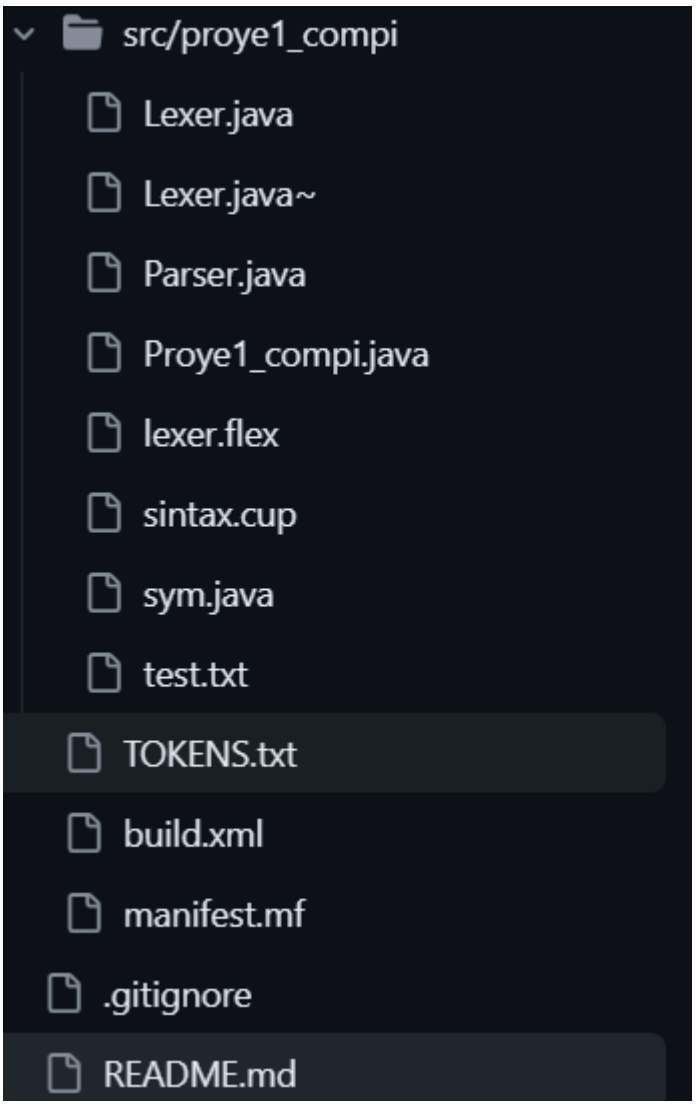


La imagen muestra los mensajes que aparecen durante el proceso de generación del lexer.

A continuación se muestran los mensajes que aparecen en consola al generarse el parser.java



La siguiente imagen muestra el árbol de archivos en el cual ya aparecen el Lexer.java, sym.java y Parser.java generados



Si el proceso continúa como debe ser, debería aparecer en consola tanto los tokens generados, que también se guardarán en un archivo, y las tablas de símbolos.

A continuación se muestran los tokens generados.

```
Token: 62 glob
Token: 48 :
Token: 24 char
Token: 48 :
Token: 2 pi
Token: 72 ;
Token: 62 glob
Token: 48 :
Token: 23 int
Token: 48 :
Token: 2 pi
Token: 72 ;
Token: 62 glob
Token: 48 :
Token: 26 bool
Token: 48 :
Token: 2 pi
Token: 72 ;
Token: 62 glob
Token: 48 :
Token: 27 string
Token: 48 :
Token: 2 pi
Token: 72 ;
Token: 62 glob
Token: 48 :
Token: 24 char
Token: 48 :
Token: 2 pi
Token: 72 ;
Token: 62 glob
Token: 48 :
Token: 23 int
```

Las tablas de símbolos se muestran en la consola.

```
Tabla de simbolo : fun1 : valores: tipo:func:int

Tabla de simbolo : globalTS : valores: variableGLob: pi:char
variableGLob: pi:int
variableGLob: pi:bool
variableGLob: pi:String
variableGLob: pi:char
variableGLob: pi:int
variableGLob: pi:bool
variableGLob: pi:String
variableGLob: pi:int
variableGLob: pi:char
variableGLob: pi:int
variableGLob: pi:bool
variableGLob: pi:String
variableGLob: pi:float
variableGLob: y:int

Tabla de simbolo : fun3 : valores: tipo:func:int

Tabla de simbolo : func1 : valores: tipo:func:float
parametro : x22:char
parametro : x22:char
parametro : x22:int
variableLoc: x22:char
variableLoc: ch33:char
variableLoc: y:int
```

Descripción del problema

Se requiere desarrollar una aplicación de java utilizando las herramientas jflex y java cup para generar y lexer y un parser basándose en la gramática de la tarea 1 del curso compiladores e intérpretes del año 2024.

Es importante que el estudiante aprenda a profundidad cómo funcionan estas herramientas para que pueda desenvolverse en el mercado ante una situación en la cual deba desarrollar gramáticas o compiladores para un lenguaje en específico, esto también le va a permitir al profesional en formación como identificar posibles soluciones a problemas de un contexto similar y a aprovechar las herramientas con las que se cuenta en la actualidad.

Debemos recordar que conforme aparecen nuevas tecnologías sigue siendo relevante contar con el conocimiento para desarrollar lenguajes para las nuevas arquitecturas que aparecen y poder aprovechar las mismas.

Diseño del programa

Para este programa se optó inicialmente por un programa con patrón de diseño monolítico, esto para tener un mejor control de las rutinas que se llevaban a cabo, probablemente para la segunda entrega del mismo se cambie esto a un patrón distribuido de clases para cumplir con el principio de responsabilidad única de los principios SOLID y para un mejor control de las partes del programa en cada etapa específica.

Este acercamiento modular lo intentamos implementar desde la primera etapa del proyecto, sin embargo no fue posible por motivos de tiempo.

Librerías usadas

En la parte superior de cada clase de java pueden verse las librerías utilizadas, entre las mas importantes a mencionar son:

- import java_cup.runtime.*: Una librería de CUP para el parseo de los elementos de la gramatica
- import java.util.HashMap; Para la creación de arreglos de tipo Hash que pueden ser consultado con los identificadores o índices de los objetos en la colección.
- import java.util.List; Para la creación de listas
- import java.util.ArrayList; Para la creación de colecciones
- import java.io.BufferedReader;
- import java.io.File;
- import java.io.FileNotFoundException;
- import java.io.FileReader;
- import java.io.FileWriter;
- import java.io.IOException;
- import java.io.Reader;
- import java.nio.file.Files;
- import java.nio.file.Path;
- import java.nio.file.Paths;
- import java.nio.file.StandardCopyOption;
- import java_cup.runtime.Symbol;
- import jflex.exceptions.SilentExit;
- import static jflex.logging.Out.println;

Análisis de resultados

El siguiente cuadro muestra cada uno de los objetivos y su nivel de aceptación o aprobación.

Requerimiento	Alcanzado
Desarrollo de la gramática BNF	100%
Desarrollo del analizador léxico	100%
Desarrollo de Parser	100%
Anílisis de codigo fuente	100%
Identificación de toques	100%
Desarrollo de tablas de símbolos	100%
Manejo de errores	100%

Bitácora

La bitácora del proyecto puede ser consultada en el repositorio en línea de github del siguiente enlace: https://github.com/FR3DD221/Compi-Proye-1.git