Compiladores e Intérpretes

Informe etapa 1

Analizador léxico

Bonanno, Pablo Ariel

Licenciatura en Ciencias de la Computación

Universidad Nacional del Sur

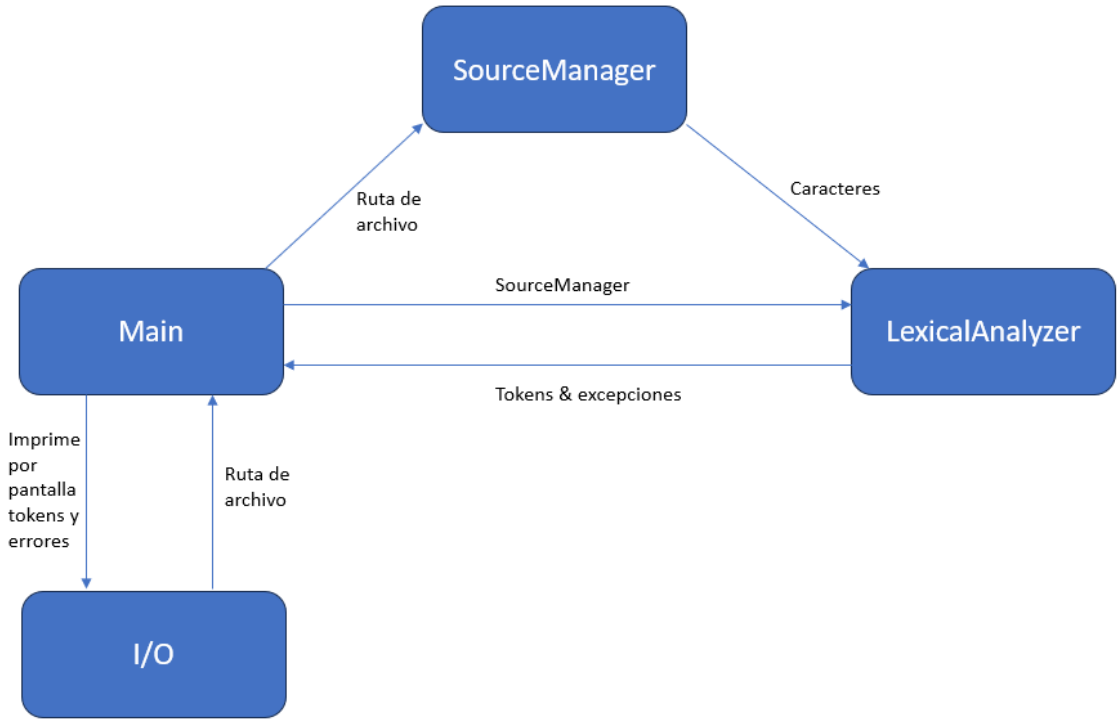


Diagrama general del funcionamiento del programa

Módulo principal

Consiste en la clase Main. Es la encargada de inicializar el analizador léxico y el manejador de archivos. Una vez creados, solicita de forma iterada al analizador léxico los *tokens* del archivo pasado como parámetro al ejecutar el programa y los imprime por pantalla. También se encarga de manejar las excepciones generadas por el analizador léxico.

Módulo manejador de archivos

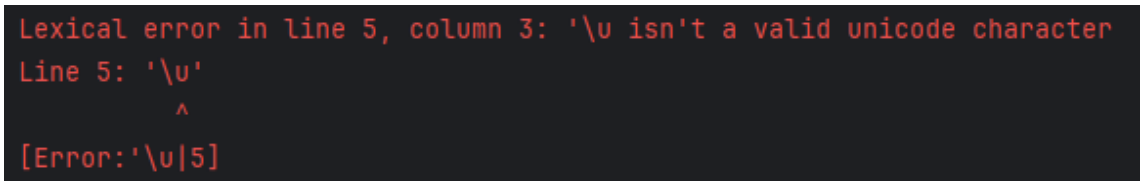
Consiste en la clase EfficientSourceManager y su interfaz. Recibe en su constructor una ruta a un archivo. Brinda servicios para leer dicho archivo, ya sea carácter por carácter o una línea completa.

Por defecto, el manejador de archivos lee carácter por carácter utilizando un BufferedReader sobre el contenido del archivo. Si se solicita la línea actual, devuelve tanto lo ya leído como lo que resta para completarla. El fragmento pendiente se almacena, y posteriormente, hasta completar dicha línea, la lectura se realiza carácter por carácter a partir de ese *string* guardado.

Módulo analizador léxico

Consiste en la clase LexicalAnalyzer y su interfaz. Utiliza el manejador de archivos para analizar el contenido de este e implementa el autómata que transforma el flujo de caracteres en tokens reconocidos o errores léxicos.

Al detectar un error léxico, el programa lo muestra por pantalla, indicando la línea y columna donde se encuentra el error, imprimiendo dicha línea por pantalla e indicando dónde se encuentra el lexema que contiene el error.



El programa reconoce los siguientes tipos de errores léxicos:

Error	Ejemplos
Literal char vacío	''
Unicode ilegal	\u - \u123- \u12J3
Longitud del literal int mayor a 9	1234567890
Símbolo inválido	# - $ - \
Salto de linea en string	"error error"
Demasiados caracteres en un literal char	ab' - '\u12345'
Literal char sin cerrar	'
Comentario sin cerrar	/* comentario
Literal string sin cerrar	"String sin cerrar

Logros intentados

Se intentaron cumplir todos los logros posibles:

- Unicodes
- Reporte de error elegante
- Columnas
- Multi-detección de Errores Léxicos
- Manejador de Archivos Eficiente

Tokens reconocidos

Token	Expresión regular	Observaciones
METVARID	minusc (minusc | mayusc | _ | digito)*	
CLASSID	mayusc (minusc | mayusc | _ | digito)*	
INTLITERAL	digito {1,9}	
CHARLITERAL	' (char | u | \ (\ | ' | char | (u u* (a..f | A..F | digito){4})))	char es cualquier caracter, exceptuando salto de linea, EOF, ', \ y u
STRINGLITERAL	" (char | \ (\ | " | char))* "	char es cualquier caracter, exceptuando salto de linea, EOF, " y \
CLASS_WORD	class	
EXTENDS_WORD	extends	
PUBLIC_WORD	public	
STATIC_WORD	static	
VOID_WORD	void	
BOOLEAN_WORD	boolean	
CHAR_WORD	char	
INT_WORD	int	
ABSTRACT_WORD	abstract	
FINAL_WORD	final	
IF_WORD	if	
ELSE_WORD	else	
WHILE_WORD	while	
RETURN_WORD	return	
VAR_WORD	var	
THIS_WORD	this	
NEW_WORD	new	
NULL_WORD	null	
TRUE_WORD	true	
FALSE_WORD	false	
OPENING_PAREN	(	
CLOSING_PAREN	)	
OPENING_BRACE	{	
CLOSING_BRACE	}	
SEMICOLON	;	
COMMA	,	
PERIOD	.	
COLON	,	
LESS_THAN	<	
EQUAL_LESS_THAN	<=	
GREATER_THAN	>	
EQUAL_GREATER_THAN	>=	
EXCLAMATION_POINT	!	
DIFERENT	!=	
EQUAL	=	
EQUALS_COMPARISON	==	
AND	&&	
OR	||	
PERCENT	%	
PLUS	+	
PLUS1	++	
MINUS	-	
MINUS1	--	
MULTIPLY	*	
SLASH	/	
EOF	charEOF	charEOF corresponde con el carácter codificado en java por el valor 26