Informe Etapa 1
Compiladores e Interpretes

Nahuel Ignacio Fuentes

August 26, 2023

Uso

El proyecto fue desarrollado en InteliJ y compilado con Java 11, para poder compilarlo es necesario ...

Tokens y sus Expresiones regulares asociadas

Por simplificacion asociaremos a estas expresiones regulares las siguientes abreviaciones, con el fin de hacer la expresion resultante más legible :

```
Digito = [0..9]
Letra = [a..z]|[A..Z]
Letra_Mayus = [A..Z]
Letra_Minus = [a..z]
Caracter = [Letra | Digito | , | . | ; | ( | .. | ] ]
```

- Identificador de clase

  ```
  idClase = Letra_Mayus [ Letra | Digito | _ ]*
  ```

- Identificador de metodo y variable

  ```
  idMetVar = Letra_Minus [ Letra | Digito | _ ]*
  ```

- Enteros

  ```
  intLiteral = Digito ^ n , 1 <= n <= 9
  ```

- Floats

  ```
  floatLiteral = intLiteral . intLiteral
  ```

- Caracteres

```
charLiteral = ' [[Character - {\ , '}] | [\ Character] ] '
```

- String

```
stringLiteral = " [ Character - {\ , ' , \n} | [ \" ] ]* "
```

- Puntuación

```
openPar = (
closePar = )
openCurl = {
closeCurl = }
period = .
comma = ,
semiColon = ;
```

- Operadores

```
opAdd = +
opSub = -
opProd = *
opIntDiv = %
opDiv = /
opLess = <
opLessEq = <=
opEq = ==
opGreater = >
opGreaterEq = >=
opAnd = &&
opOr = ||
opNot = !
opNotEq = !=
```

- Asignacion

```
assign = =
assignAdd = +=
assignSub = -=
```

- Palabras reservadas

`resWord` estas no tienen asociadas ninguna Expresion Regular.

Tipos de errores

Se reconocen diversios tipos de errores lexicos, estos son :

- Simbolos que no pertenecen al alfabeto
- Comentarios multilinea sin cerrar
- Cadena de caracteres sin cerrar
- Operadores logicos no validos

Clases utilizadas

Durante el desarrollo se pensón 4 Clases para la totalidad de la etapa, esas clases fueron LexicalAnalyzer, FileManager, Token y Main. Luego tambien se implementó una excepción especifica llamada LexicalException para cuando se detectan excepciones lexicas, nos permite obtener información acerca de donde y porque ocurrió el error.

Además la clase FileManager paso a ser interfaz para poder probar distintas implementaciones, finalmente quedando la clase ImpFileManager que implementa la interfaz previamente mencionada a traves del uso de FileReader y BufferReader.

La clase LexicalAnalyzer es la encargada de intentar reconocer tokens de un archivo y luego retornarlos o informar alguna excepción que haya ocurrido, estas por último las maneja la clase Main que se encarga de mostrar una gráfica para poder informar los tokens o los errores que ocurrieron, adem;ás de inyectar el manejador de archivos con el camino especificado por parametro al analizador.

Logros intentados

Los logros que se intentaron resolver durante el desarrollo de la etapa fueron :

- Imbatibilidad Lexica
- Reporte de Error Elegante
- Columnas