Documentação dos principais métodos das Classes

LÉXICO

`lexical.Scanner.Scanner(source)`

`source`: String do local do arquivo.pas a ser compilado

`lexical.Scanner.nextToken()`

Onde fica o código referente ao autômato da linguagem.

Retorna um token de acordo com a especificação do autômato ou um erro caso o token não esteja dentro das especificações.

`lexical.Scanner.tokenizer()`

Itera o buffer do arquivo chamando o método `nextToken()`.

Retorna um buffer de tokens

SINTÁTICO

`parser.Parser`

Classe onde foram implementados os analisadores sintático e semântico, os atributos dessa classe que são referentes apenas a análise sintática são:

`buffer`: buffer de tokens gerado pelo léxico

`pos`: posição no buffer

`currentToken`: token na posição do buffer

`parser.Parser.parser(buffer)`

Inicia a análise preditiva recursiva pelo buffer de tokens passado por parâmetro

`parser.Parser.advance()`

Avança para o próximo

`parser.Parser.match( tipo_esperado | texto_esperado )`

Verifica se o tipo/string do token atual no buffer é igual ao tipo/string do parâmetro '`token_esperado`'. Se for, chama `advance()` e vai pro próximo token do buffer de tokens, ao contrário, joga erro de token inválido.

`parser.Parser.isCurrentTokenType(tipo_esperado)`

Verifica se o token atual no buffer tem o mesmo tipo de `tipo_esperado`

`parser.Parser.isCurrentTokenText(texto_esperado)`

Verifica se o token atual no buffer tem o mesmo texto de `texto_esperado`

parser.Parser.error()

Erro sintático v1

parser.Parser.errorExpected()

Erro sintático v2 que alerta o valor lido vs valor esperado

parser.Parser.errorMalformedExpression()

Erro sintático v2 que alerta que uma expressão foi mal feita

SEMÂNTICO

parser.Parser

os atributos dessa classe que são referentes apenas a análise sintática são:

scopeStack: Pilha de escopos

PcT: Pilha com tipos

utils.ScopeStack

Pilha de escopo para análise semântica. Essa pilha serve para verificar erros de variáveis já declaradas e garantir a procedência de uso de variáveis de acordo com o escopo.

utils.ScopeStack.push(token)

Adiciona o Token ao topo da pilha

utils.ScopeStack.pop()

Remove e retorna o token do topo da pilha

utils.ScopeStack.getFromTop()

Utilizado para iterar a pilha do índice do topo até a base

utils.ScopeStack.assignTypeWhereIsNull(tipo_de_token)

Itera a pilha de escopo do topo até a base atribuindo `tipo_de_token` aos tokens cujo o tipo é null (serve para a técnica de atribuir um tipo à uma lista de variáveis declaradas)

utils.ScopeStack.scopeContains(texto)

Itera a pilha de escopo do topo até a marcação $ procurando se há algum um token na pilha cujo o texto é igual ao passado no parâmetro (a marcação $ indica início do escopo, esse método será útil para alertar erro de variável já declarada em um escopo)

utils.ScopeStack.cleanScope()

Desempilha a pilha de escopo até a marcação $

parser.Parser.tryToPush(texto, tipo)

Verifica se a variável já foi declarada dentro do escopo para alertar erro ou empilhar um novo token passando o texto, novo tipo, linha atual e coluna atual.

parser.Parser.cleanScope()

Desempilha a pilha de escopo até a marcação $

parser.Parser.assignTypeWhereIsNull(tipo_de_token)

Itera a pilha de escopo do topo até a base atribuindo tipo_de_token aos tokens cujo o tipo é null (serve para a técnica de atribuir um tipo à uma lista de variáveis declaradas)

parser.Parser.findTokenOnScopeStack(texto_token)

Busca e retorna o token na pilha de escopos cujo o texto é igual ao texto passado por parâmetro

parser.Parser.atualizePcT(tipo_resultante)

Atualiza o novo topo da pilha de tipos de acordo com o topo e o subtopo

parser.Parser.checkTypes(tipo_1, tipo_2)

Checa se os tipos do topo e subtopo da pilha de tipos são iguais aos tipos tipo_1 e tipo_2 passados por parâmetro.