UNIVERSIDADE DO VALE DO RIO DOS SINOS

Exercício 2

Nomes: Bruno da Siqueira Hoffmann e Rafael Hansen Klauck.
Atividade Acadêmica: Compiladores.
Ano/Semestre: 2024/02.
Professor: Flávio Oliveira

Exercício 2 - Analisador Sintático Para Uma Linguagem de Programação

Informações Gerais: Para criação do analisador léxico foi utilizado a ferramenta ANTLR, onde foi feito a migração dos tokens léxicos antes criados para essa nova ferramenta, além da criação das regras de produção para o analisador sintático. Com relação aos arquivos presentes na entrega é possível visualizar:

- **Exercício 2 - Compiladores - Bruno Hoffmann e Rafael Klauck.pdf**: contendo o resultado do teste executando o comando "grun FoolGrammar programa -gui ../tests/arquivoErro.txt";
- **FoolGrammar.g4:** contendo a especificação da gramática;
- **arquivoSucesso.txt**: contendo o código teste;
- **arvoreCasoSucesso.png**: contendo uma imagem da árvore gerada.

Teste: Para testar a gramática fool desenvolvida foram utilizados dois arquivos, um contendo exemplos de IFs, IFs Elses, IFs Elses encadeados, além de expressões booleanas, expressões aritméticas, métodos com parâmetros e métodos sem parâmetros.

O arquivo refere a caso de sucesso se chama "**arquivoSucesso.txt**", além disso adicionamos a foto da árvore gerada por ele, podendo ser observada no arquivo "**arvoreCasoSucesso.png**". Abaixo é possível verificar o output quando executado o comando "tree".

(programa (declaracaoClasse class Main { (declaracaoAtributo (tipo int) precoProduto ;) (declaracaoAtributo (tipo int) valorInsento ;) (declaracaoAtributo (tipo bool) ehCaro ;) (declaracaoAtributo (tipo bool) mostrarOutputEmConsole ;) (declaracaoMetodo (tipoMetodos void) main ( ) { (comandos (comando (atribuicao valorFinal = (expressao (expressaoAritmetica (termo (chamadaMetodo calcularTaxa ((expressao (expressaoAritmetica (termo precoProduto)))))))))) ; (comando (chamadaMetodo mostrarValor ((expressao (expressaoAritmetica (termo

True))) , (expressao (expressaoAritmetica (termo valorFinal)))))) ;) }) (declaracaoMetodo (tipoMetodos void) calcularTaxa ((argumentos (tipo int) preco)) { (comandos (comando (condicional if ((expressaoBooleana (fatorBoole (termoBoole valorInsento) > (termoBoole preco)))) (comando return (expressao (expressaoAritmetica (termo 0)))) ; else (comando return (expressao (expressaoAritmetica (termo preco) * (termo (chamadaMetodo obterTaxa ( )))))) ;)) ;) }) (declaracaoMetodo (tipoMetodos (tipo int)) obterTaxa ( ) { (comandos (comando return (expressao (expressaoAritmetica (termo 2)))) ;) }) (declaracaoMetodo (tipoMetodos (tipo bool)) mostrarValor ((argumentos (tipo bool) mostrarValor , (tipo bool) consoleAtivo , (tipo int) valorFinal)) { (comandos (comando (atribuicao mostrarOutputEmConsole = (expressao (expressaoBooleana (fatorBoole (termoBoole deveMostrar) && (termoBoole consoleAtivo)))))) ; (comando (condicional if ((expressaoBooleana (fatorBoole (termoBoole mostrarOutputEmConsole)))) (comando (condicional if ((expressaoBooleana (fatorBoole (termoBoole ehCaro)))) (comando (chamadaMetodo mostrarValorEmVermelho ((expressao (expressaoAritmetica (termo valorFinal)))))) ;)) ; else (comando (chamadaMetodo mostrarValorEmVerde ((expressao (expressaoAritmetica (termo valorFinal)))))) ;)) ; (comando (condicional if ((expressaoBooleana (fatorBoole (termoBoole ! (termoBoole mostrarOutputEmConsole))))) (comando (chamadaMetodo mostrarValorUI ( ))) ;)) ;) }) (declaracaoMetodo (tipoMetodos void) mostrarValorEmVerde ((argumentos (tipo int) valor)) { comandos }) (declaracaoMetodo (tipoMetodos void) mostrarValorEmVermelho ((argumentos (tipo int) valor)) { comandos }) (declaracaoMetodo (tipoMetodos void) mostrarValorUI ((argumentos (tipo int) valor)) { comandos }) (declaracaoMetodo (tipoMetodos void) mostrarValorEmVerde ( ) { comandos }) (declaracaoMetodo (tipoMetodos void) mostrarValorEmVermelho ( ) { comandos }) }) <EOF>)