Relatório Compiladores –

Geração e Interpretação de Código

**Acadêmicos:**

Darlan Nakamura

Rafael Rodrigues

# Introdução

A compilação da LALG foi realizada utilizando as ferramentas JFlex e Java CUP. A fim de que se faça a análise semântica e a geração de código, o CUP exige a criação de classes para cada não terminal da gramática. Estas classes possuem um método, chamado .run(), que engloba a análise semântica e a geração de código, com exceção da classe Programa, que possui o método .exec().

Após a realização da análise sintática, o método .exec() inicia a análise semântica concomitante à geração de código.

Este relatório apresenta os pontos principais do fluxo da geração e interpretação do código, passando pelas classes principais e detalhes importantes do funcionamento.

# Geração de Código

## Iniciando o Processo

A classe Programa possui um bloco, o nome do programa, e um objeto GeracaoDeCodigo que se trata da lista de instruções que compõem o código intermediário. Após ser instanciada com seus componentes durante a análise sintática, executa-se o comando .exec(), mostrado na Figura 1.

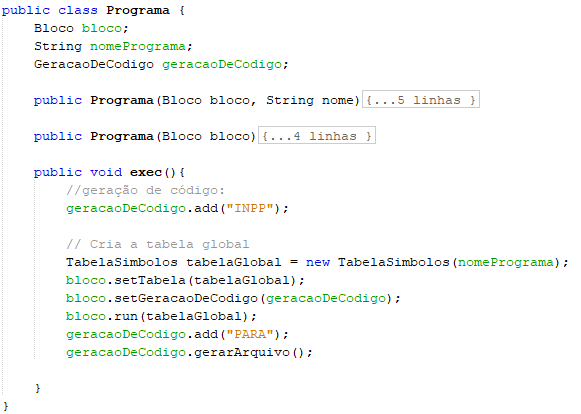


Figura 1: classe Programa

É possível observar o funcionamento da geração de código na Figura 1. A instrução INPP é a primeira a ser adicionada no código, então cria-se a tabela global (adicionando-a ao bloco do programa), e então o método .run() do bloco é chamado. Somente após a execução dos métodos .run() de todos os componentes do bloco, será adicionado a instrução PARA, e então o *bytecode* é gerado.

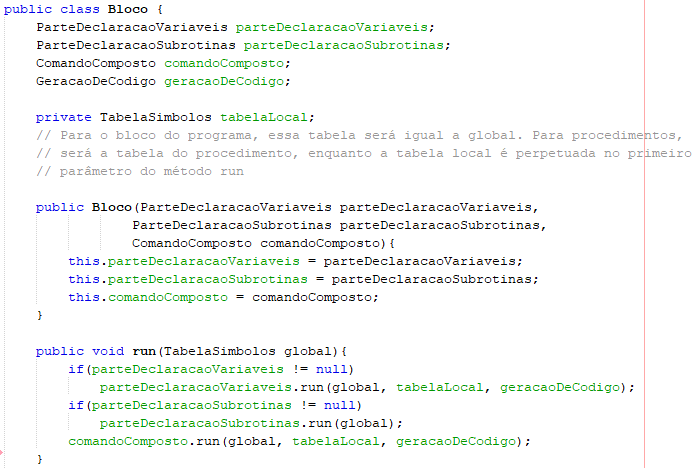


Figura 2: início da classe Bloco

A Figura 2 apresenta o método .run(), que executa o método .run() da parte de declaração de variáveis e da parte de declaração de sub-rotinas, caso existam, seguido pelo comando composto. Na classe Bloco nenhum código é gerado.

## Declaração de Variáveis

A classe ParteDeclaracaoVariaveis é composta por um ArrayList de DeclaracaoVariavel. Cada declaração é uma lista de identificadores com o mesmo tipo.

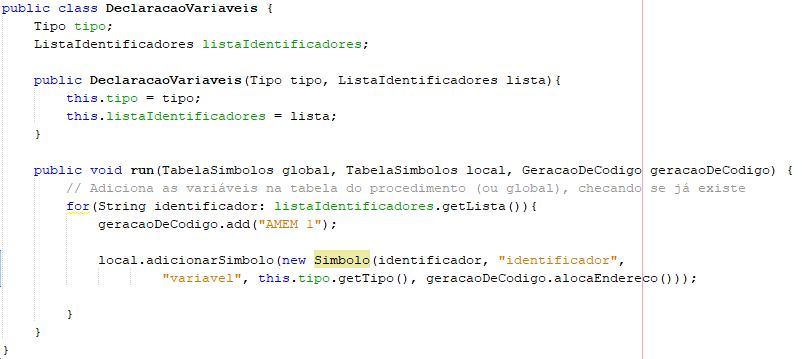


Figura 3 – Classe DeclaracaoVariaveis

No método .run() desta classe gera-se o código “AMEM 1”. A classe GeracaoDeCodigo, por meio do método .alocaEndereco(), é responsável por determinar o endereço de cada variável na área de dados do programa, como mostrado na última linha escrita da Figura 3.

## If e Else

Na Figura 4 é mostrado o funcionamento dos rótulos para o comando if e else.

Primeiro criam-se dois rótulos com auxílio da classe GeracaoDeCodigo, então todo o código da expressão é gerado por meio da chamada de seu método .run. Segue-se a realização da condição gerada pela expressão (0 ou != 0). No caso de ser verdadeiro (!= 0), o programa é desviado para a execução do comandoFalse (rotuloElse), caso contrário a interpretação segue seu fluxo normal, sendo desviada para após o código do comandoFalse (rotuloDoFim).

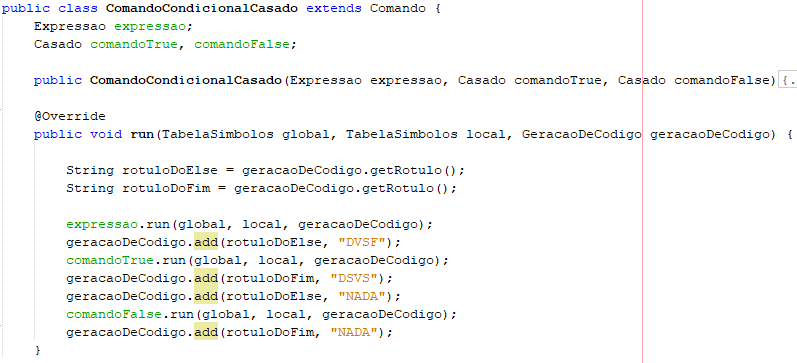


Figura 4: if e else casado

## Operações matemáticas e lógicas

Na classe Expressao são realizadas as operações de soma, divisão, etc. Na Figura 5 encontram-se os exemplos da comparação se é igual e se é maior ou igual.

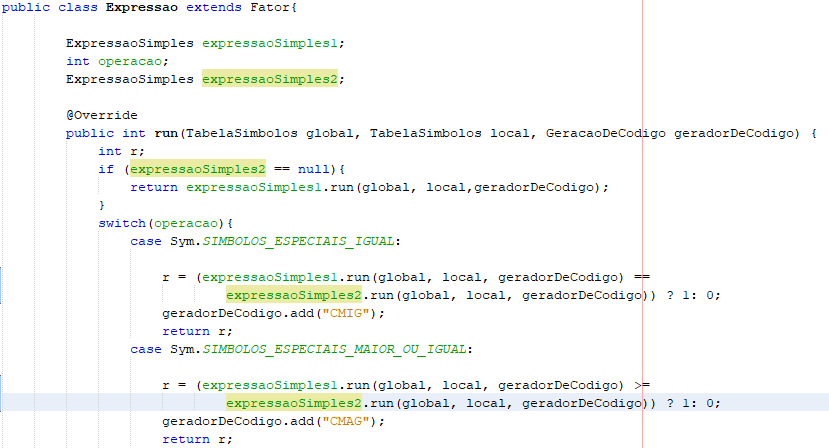


Figura 5: início da classe Expressao

Caso a Expressao não seja uma operação, apenas adiciona-se o código da expressão na tabela, que resultará em um inteiro no fim de sua interpretação (ou boolean representado por 0 ou 1). Caso a Expressao possua uma operação, então o código de ambas é adicionado, seguido pela instrução do operador correspondente.

# Interpretação do Código

Após a geração do *bytecode*, a máquina virtual MEPA interpreta o código.

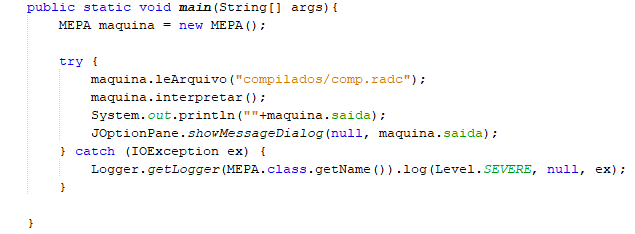


Figura 6: função main da MEPA.

A única diferença da máquina virtual implementada, quando comparada à apresentada em aula, é que a nossa não apresenta um ponteiro para o topo da pilha, pois a classe Stack do Java realiza esse controle automaticamente.