Insper

Lógica da Computação - 2020/1

Roteiro 4 - Simple Calculator **v2.0**

Raul Ikeda - rauligs@insper.edu.br

Entrega: 01/Apr/2020 às 13h30

Nome:

Objetivos

1. Implementar a árvore sintática abstrata (AST)

Tarefas:

1. Criar uma classe abstrata (ou interface) chamada *Node*, contendo os seguintes atributos e métodos:

```
value: variant
children: list of nodes
Evaluate(): variant
```

2. Criar as seguintes classes que herdam (ou implementam) a classe *Node*:

```
BinOp - Binary Operation. Contem 2 filhos
UnOp - Unary Operation. Contem um filho
IntVal - Integer value. Não contem filhos
NoOp - No Operation (Dummy). Não contem filhos
```

Todas as classes acima devem retornar no método *Evaluate()* - via *overriding* - o valor correspondentes à operação de cada classe. Por Exemplo: *Evaluate()* do *IntVal* deve retornar o próprio valor inteiro, *Evaluate()* de *BinOp* deve retornar a operação dos seus dois filhos, etc.

3. Modificar o *Parser* para não mais interpretar a entrada e sim retornar a AST montada. Com isso cada função do *Parser* retorna um objeto do tipo *Node* (e não um valor). O responsável por executar cada função tem que se encarregar de montar a árvore e retornar o próprio nó. Por Exemplo: a função *parseExpression* coleta a árvore resultante do *parseTerm* e monta o seu próprio nó conforme realiza o consumo dos tokens, retornando o seu nó ao final.
4. Quando o *main()*, de posse de toda a árvore montada, executar o método *Evaluate()* da raiz, este deverá executar **recursivamente** os seus filhos, obtendo ao final o valor interpretado.
5. Modificar o programa principal para ler um arquivo de extensão *.php* ao invés de ler a entrada do console.

Base de Testes:

```
Todos os testes dos roteiros anteriores
```

Questionário

1. Montar a árvore de derivação e a AST para: (2 + 3) / (5 * 1). Explique por que parênteses não precisa de uma classe *Node* específica.

2. Proponha uma ideia para implementar variáveis no compilador.