Insper

Lógica da Computação - 2020/1

Roteiro 1 - Simple Calculator **v1.0**

Raul Ikeda - rauligs@insper.edu.br

Entrega: 03/Mar/2020 às 13h30

Nome:

Objetivos

1. Reestruturar o compilador para o modelo visto em aula.
2. Implementar a separação de fases.

GitHub - *Issues*

Antes de começar o desenvolvimento de um novo roteiro, criar sempre uma *branch* a partir do *master*. É sugerido que o nome da branch tenha algo a ver com a versão, mas ela não pode ser o mesmo nome do futuro release. Com isso, vocês conseguem corrigir eventuais erros no roteiro anterior sem afetar o desenvolvimento do novo roteiro. Para isso, crie também uma branch a partir do *release*, corrija as *issues*, faça o *merge* e solte uma nova *release* na *master* do roteiro anterior. Quando retomar o desenvolvimento desse roteiro, ao final faça o merge e corrija os conflitos.

Tarefas:

1. Criar uma *branch* com *checkout*.
2. Colocar o Diagrama Sintático no GitHub.
3. Criar uma Classe **Token** com 2 atributos:
   - type: string. tipo do token
   - value: integer. valor do token
4. Criar uma Classe **Tokenizer** com 3 atributos e 1 método:
   - origin: string. código-fonte que será tokenizado
   - position: integer. posição atual que o Tokenizador está separando
   - actual: token. o último token separando
   - selectNext(): lê o próximo token e atualiza o atributo *atual*
5. Criar uma Classe **Parser** com 1 atributo e 2 métodos (Sugestão: *estáticos*):
   - tokens: Tokenizer. Objeto da classe que irá ler o código fonte e alimentar o Analisador.
   - parseExpression(): consome os tokens do Tokenizer e analisa se a sintaxe está aderente à gramática proposta. retorna o resultado da expressão analisada.
   - run(code): recebe o código fonte como argumento, inicializa um objeto Tokenizador e retorna o resultado do parseExpression(). Esse método será chamado pelo main().
6. Fazer o *merge*, corrigir os conflitos (se houver) e soltar o release na *master*.

Base de Testes:

```
>> 1+2
>> 3-2
>> 1+2-3
>> 11+22-33
>> 789   +345  -    123
```

Questionário

1. Nosso compilador é de 1 passagem ou múltiplas passagens? Justifique.

2. Colocar as operações de MULTIPLICAÇÃO e DIVISÃO e testar:

```
>> 4/2+3
>> 2 + 3 * 5
```

1. Explique por que o resultado está incorreto.

2. Sugira a correção no diagrama sintático (não precisa implementar).