UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
FACOM - FACULDADE DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

COMPILADORES I (2016/2)
PROFA. BIANCA DE ALMEIDA DANTAS

Trabalho Prático - Análise Sintática e Interface Gráfica

1 DESCRIÇÃO

A segunda parte do trabalho prático de nossa disciplina consiste na implementação do analisador sintático descendente preditivo para a linguagem MiniJava modificada, cuja gramática se encontra ao final desse texto, e de uma interface gráfica para o nosso ambiente de compilação.

O analisador sintático deve ser capaz de percorrer o programa fonte, detectar e reportar erros. Seu programa não precisa utilizar técnicas de recuperação de erros.

A interface gráfica desenvolvida servirá para facilitar o processo de edição e compilação do arquivo fonte (o qual deve ter a extensão `.mj`). Ela deverá fornecer, pelo menos, as seguintes funcionalidades:

- Uma área de texto para exibir o código fonte do programa;
- Uma área de texto para exibir mensagens do compilador;
- Opções para abrir e salvar arquivos, bem como para criar um novo;
- Opções para compilar o arquivo aberto.

2 ESPECIFICAÇÕES

- O trabalho prático poderá ser realizado em grupos de, no máximo, 3 alunos **sem exceções**.
- A linguagem Java deverá ser utilizada na implementação do trabalho.
- A entrega de todas as etapas deve ser realizada até o dia: **23/03/2017**.
- Qualquer parte do trabalho copiada ou "fortemente inspirada" nos trabalhos de outros grupos receberão nota **zero**.
- Entrevistas **podem** ser realizadas com todos os grupos.

3 GRAMÁTICA

A gramática seguinte utiliza as notações **(N)*** para representar 0 ou mais repetições de **N** e a notação **(N)?** para representar 0 ou 1 repetição de **N**.

1. *Program* → *MainClass* (*ClassDeclaration*)*

2. *MainClass* → **class ID { public static void main (String[ ] ID){** *Statement* **} }**

3. *ClassDeclaration* → **class ID** (**extends ID**)? **{** (*VarDeclaration*)* (*MethodDeclaration*)* **}**

4. *VarDeclaration* → *Type* **ID ;**

5. *MethodDeclaration* → **public** *Type* **ID (** (*Type* **ID** (**,** *Type* **ID**)*)? **) {** (*VarDeclaration*)* (*Statement*)* **return** *Expression* **; }**

6. *Type* → **int[]** | **boolean** | **int** | **ID**

7. *Statement* → **{** (*Statement*)* **}**
   | **if (** *Expression* **)** *Statement* **else** *Statement*
   | **while (** *Expression* **)** *Statement*
   | **System.out.println (** *Expression* **) ;**
   | **ID =** *Expression* **;**
   | **ID [** *Expression* **] =** *Expression* **;**

8. *Expression* → *Expression Op Expression*
   | *Expression* **[** *Expression* **]**
   | *Expression* **. length**
   | *Expression* **. ID (** (*Expression* (**,** *Expression*)*)? **)**
   | **INTEGER_LITERAL**
   | **true**
   | **false**
   | **ID**
   | **this**
   | **new int [** *Expression* **]**
   | **new ID ( )**
   | **!** *Expression*
   | **(** *Expression* **)**

9. *Op* → **&&** | **<** | **>** | **==** | **!=** | **+** | **-** | ***** | **/**