Universidade de São Paulo

Teoria da Computação e Compiladores

Trabalho 2
Parser em JASON

Felipe Aparecido Garcia 9368751
Marcelo Bertoldi Diani 9313291
Marco Adriano Tette Schaefer 9266302
Pedro Morello Abbud 8058718

Professora Dra.
Sandra Aluisio

8 de Maio de 2018

Conteúdo

1 Introdução

Neste trabalho, foi implementado um *parser* para a linguagem Jason, utilizando a ferramenta JavaCC. Nele, definiram-se as regras da linguagem, com a notação EBNF (*Extended Backus–Naur Form*), e os *tokens* da mesma, a serem gerados pelo *parser*.

Além disso, foram desenvolvidos casos de teste para diversas funcionalidades da linguagem, utilizando o *framework* de teste JUnit para executar todos os testes, apoiado na *Build Tool* chamada *Gradle*. Os testes foram divididos em testes que deveriam ser aceitos pelo *parser* e testes onde o *parser* deveria gerar erros.

2 Questões

Antes de iniciar a construção do analisador léxico, algumas decisões de projeto devem ser tomadas. Com este propósito, propôs-se perguntas a serem respondidas sobre a linguagem:

1. A linguagem permite comentário dentro de comentário? São sensíveis à linha? Qual o marcador de comentário?

   Resposta: A linguagem não permite comentários dentro de comentário, sendo que o comentários são sensíveis à linha e possuem como marcador o símbolo '#'.

2. Os identificadores e as palavras-reservadas serão *case-sensitive*? Ou seja, haverá diferenciação entre letras minúsculas e maiúsculas?

   Resposta: Sim, os identificadores e as palavras reservadas serão *case-sensitive*.

3. Os identificadores terão tamanho máximo de caracteres? Se não, há número de caracteres para diferenciação?

   Resposta:Os identificadores não terão um tamanho máximo de caracteres e não haverá número de caracteres para diferenciação.

4. Conjunto de palavras reservadas:

Grupo de símbolos 1	Palavras-Chave da linguagem
V_t = (‘begin’, ‘end’, ‘program’, ‘types’, ‘array’, ‘record’, ‘variables’, ‘real’, ‘integer’, ‘string’, ‘procedure’, ‘parameters’, ‘var’, ‘function’, ‘returns’, ‘read’, ‘set’, ‘write’, ‘if’, ‘then’, ‘while’, ‘do’, ‘endwhile’, ‘untill’, ‘enduntil’, ‘call’, ‘else’, ‘endif’, ‘return’)	

5. Constantes permitidas: real, integer e string.

6. Caracteres não imprimíveis e símbolos especiais: a linguagem não os aceita.

3 Utilzando e construindo a aplicação

3.1 Rodando o Parser

Utilize o arquivo *.jar* fornecido, ou construído nas subseções seguintes, da seguinte forma:

Programa 1: Executando o Parser

```
#!/bin/bash
java -jar nomeDoJar.jar
```

O programa espera como entrada um caminho para um arquivo ou caso não haja argumento de entrada, este rodará iterativamente pelo *STDIN* (*standart input*). Como definido no projeto, caso aconteça algum erro, o parser parará imediatamente e informará qual era *Token* esperada e qual foi a *Token* recebida.

3.2 Construindo o Parser

Pode-se construir o *Parser* através dos códigos fontes. Para isto algumas dependências são necessárias:

- Certifique-se de ter o Java Development Kit 8 ou superior instalado. Caso não o tenha, siga estes passos ou utilze seu gerenciador de pacotes.
- Instale o Gradle para automatização da *build*. Encontre os passos para sua distribuição aqui. Alternativamente, use seu gerenciador de pacotes.

Descompacte os arquivos fontes do zip ou clone o repositório:

Programa 2: Obtendo os arquivos fontes

```
#!/bin/bash
git clone https://github.com/abbudao/SCC0605
```

Na pasta raiz do projeto, construa uma *build*:

Programa 3: Construção automatizada

```
#!/bin/bash
gradle build
```

Os arquivos gerados pelo *JavaCC* estarão no caminho *"build/generated"* e o jar automaticamente gerado no caminho *"build/libs"*. Para a construção a suíte de testes já foi rodada e um report detalhado pode ser encontrado em *"build/reports"*.

3.3 Testes

Para executar-se os testes, o segundo comando pode ser usado:

Programa 4: Rodando a suíte de testes

```
#!/bin/bash
gradle test
```

O resultado da suite será mostrado no *STDOUT* e um resultado detalhado também pode ser encontrado em *"build/reports"*.