Construção de Compiladores 1 - 2018.1 - Profs. Mário César San Felice
(e Helena Caseli, Murilo Naldi, Daniel Lucrédio)
Trabalho 1

Neste primeiro trabalho da disciplina de Construção de Compiladores 1, você irá implementar um analisador léxico + sintático para a linguagem Lua (versão 5.1)[1], uma linguagem de programação de scripts desenvolvida na PUC-RJ. Antes de começar o trabalho, acesse o manual da linguagem diretamente no site[2] ou no material disponível na página da disciplina, e leia atentamente seu conteúdo.

No manual, além da parte léxica, trechos de gramática são apresentados para descrever a sintaxe de todas as construções da linguagem. No final do manual, a gramática é apresentada na íntegra. O manual também descreve a semântica da linguagem, que a princípio não será utilizada, já que neste trabalho a análise semântica não será realizada. Porém, parte do trabalho consiste em realizar testes com a gramática para verificar se programas sintaticamente corretos são analisados completamente, e se erros sintáticos são detectados. Para realizar esses testes, é importante compreender o significado de cada construção da linguagem e portanto recomenda-se a leitura atenta também da semântica.

Para desenvolver o trabalho você deverá utilizar o ANTLR[3], um gerador de analisadores léxicos e sintáticos. Toda a documentação disponível sobre o ANTLR encontra-se em seu próprio website, assim como links para download.

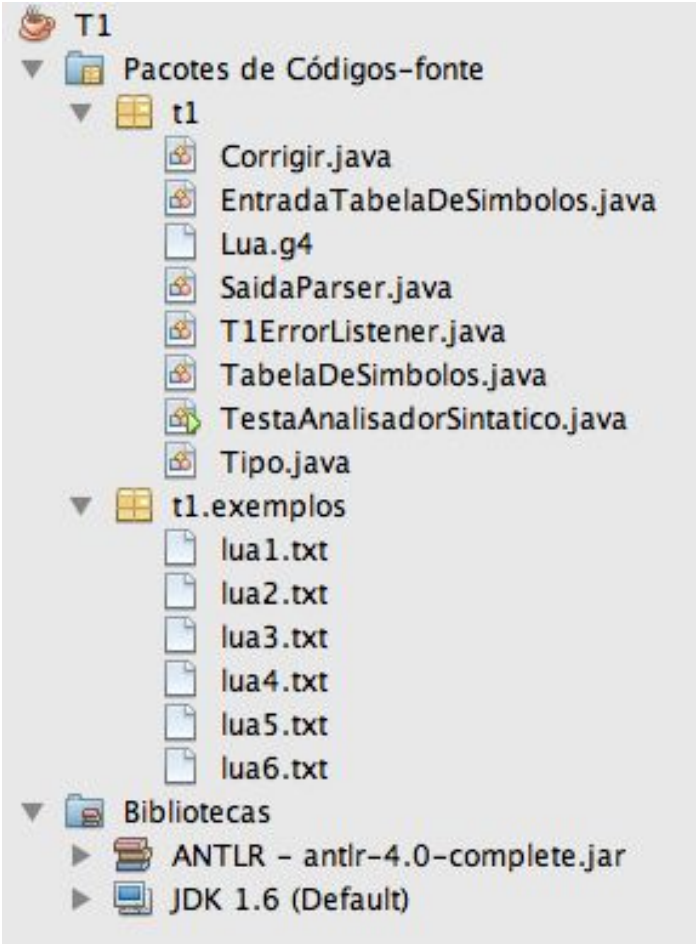
O trabalho deve ser desenvolvido em grupos de até TRÊS alunos. Atente para a política de plágio e atrasos disponível no sistema NEXOS.

A seguir são descritas mais informações de como você deverá fazer seu trabalho. Leia-as atentamente e busque construir uma implementação precisa, pois a avaliação será feita através de scripts automáticos.

[1] www.lua.org
[2] www.lua.org/manual/5.1/pt/
[3] www.antlr.org

1. O projeto a ser utilizado

No ambiente da disciplina foi disponibilizado o arquivo T1.zip, um projeto do NetBeans que deverá ser utilizado no desenvolvimento deste trabalho. Trata-se de um projeto pré-configurado para a execução do trabalho. Para acessá-lo, basta descompactá-lo e abri-lo com o NetBeans[4]. Pode ser necessário especificar o caminho completo da biblioteca/jar do ANTLR standalone. Caso a configuração tenha sido correta, o seguinte conteúdo deve ser exibido no NetBeans.



Analise cuidadosamente esses elementos para identificar sua função.

Lua.g4: arquivo da gramática. Está vazio, você precisa preenchê-lo.

TestaAnalisadorSintatico.java: arquivo de teste. Use-o para testar seu analisador com um dos exemplos fornecidos.

Corrigir.java: arquivo de correção automática. É o mesmo que será utilizado pelos professores na correção. Use-o para saber qual será a sua nota de execução.

EntradaTabelaDeSimbolos.java, TabelaDeSimbolos.java, Tipo.java: implementação da tabela de símbolos. Mais detalhes na seção 4.

SaidaParser.java, T1ErrorListener.java: classes auxiliares. Leia-as e entenda, mas não precisará mexer em nada.

t1.exemplos: exemplos de entrada

[4] O projeto foi criado e testado no NetBeans 7.3. Caso opte por utilizar outra versão, ou outro ambiente, como Eclipse, você precisará realizar a configuração do projeto novamente, importando o código-fonte e recursos manualmente.

2. Casos de teste

Na página da disciplina foi disponibilizado o arquivo casosDeTesteT1.zip, um conjunto de casos de teste, contendo a entrada/saída esperados. Esse é o conjunto que será utilizado na avaliação. Use-o como referência. Se algo não está nos casos de teste, não precisa se preocupar, pois não será cobrado.

3. A linguagem Lua

Com relação ao analisador léxico, você não precisará implementar todas as convenções léxicas da linguagem Lua, apenas o subconjunto descrito a seguir.

a) Palavras reservadas (todas)
b) Símbolos reservados (todos)
c) Nomes
d) Cadeias de caracteres (apenas as versões curtas, sem sequência de escape, quebras de linha não permitidas)
e) Constantes numéricas (apenas decimais, sem sinal, com dígitos antes e depois do ponto decimal opcionais)

Espaços em branco e comentários curtos deverão ser IGNORADOS pelo analisador léxico.

Com relação ao analisador sintático, o manual da Lua já contém a gramática da linguagem, no final. Para transformá-la para a sintaxe do ANTLR, as seguintes traduções são necessárias:

```
"::="   => ":"
{regra} => (regra)*
[regra] => (regra)?
'cadeias' devem ser envoltas por aspas simples
toda regra deve terminar com ";"
```

Algumas dicas:
- Crie uma regra nova, no início da gramática (depois dos padrões léxicos). Ela será necessária nos testes:

```
programa : trecho;
```

- É necessário alterar um pouco a gramática do manual da Lua, pois o ANTLR não tolera certos tipos de recursividade. Utilize as técnicas vistas em sala de aula para fazer a gramática funcionar. Analise a gramática com cuidado, faça testes e tente encontrar a solução. Utilize também o mecanismo de desenho dos diagramas para ajudar (Menu Janela -> Syntax Diagram).

Obs: Os seis exemplos fornecidos devem compilar sem erros. Utilize-os como base para o desenvolvimento. Mas também procure testar os casos de teste "oficiais", para verificar o bom funcionamento do seu analisador.

4. Tabela de símbolos

Além de fazer a análise sintática, o seu analisador deverá montar uma tabela de símbolos simplificada. A estrutura da tabela já está pronta, assim como o código que a imprime na saída. Basta portanto inserir as chamadas corretamente, dentro das regras sintáticas, e gerar o analisador novamente.

Para inserir um elemento na tabela de símbolos, insira uma ação semântica chamando o método "adicionarSimbolo". Ex:

```
regra : TK { TabelaDeSimbolos.adicionarSimbolo($TK.text,Tipo.VARIAVEL); }
      ;
```

O método é público e "static" , portanto nada mais é necessário. Consulte o roteiro da aula 4 para mais detalhes sobre ações semânticas, tipos de retorno, etc. Esses conceitos serão necessários para a realização do trabalho. O manual do ANTLR também pode ser consultado (acesse-o no site do antlr).

Há dois tipos de símbolos a serem inseridos na tabela: VARIAVEL ou FUNCAO. Toda variável usada deve aparecer na tabela, na ordem em que aparece, sem duplicação. Toda função, usada ou declarada, também deve aparecer na tabela, na ordem em que aparece, sem duplicação. Utilize os casos de teste como referência para entender o que deve ser feito. Analise a gramática cuidadosamente e decida onde inserir as ações semânticas para produzir o resultado correto. Não é necessário tratar duplicatas, detalhes de instanciação, etc, pois as classes fornecidas já fazem esse trabalho. Você só precisará mexer no arquivo Lua.g4.

5. Entrega do trabalho e avaliação

Você deverá entregar somente o arquivo Lua.g4, compactado em um arquivo .zip ou .rar renomeado com o RA dos alunos (ex: 176168_e_187273.zip). Insira também, no local indicado no arquivo Lua.g4, o RA dos alunos. A entrega será feita exclusivamente via Moodle, até a data indicada no ambiente.

A nota do seu trabalho será composta de 2 parcelas:

- 70% da nota será calculada automaticamente, através da classe "Corrigir.java" aplicada aos casos de teste fornecidos (casosDeTesteT1.zip).

- 30% da nota corresponde à qualidade do arquivo da gramática. Serão considerados aspectos como indentação, nome das regras, e o principal - COMENTÁRIOS EXPLICATIVOS em todas as regras, inclusive explicando aquelas onde foi necessário realizar alguma alteração. A falta de um desses elementos irá ocasionar decréscimo na nota.

Bom trabalho!