Universidade Federal Fluminense

TCC00289 - COMPILADORES - A1

Turma ministrada por Flávia Bernardini

Alunos: Fernando Vieira, Lucas Coutinho, Renan Martins

Mini Java e Calculadora: Seus Scanners, Tokens e Erros

A linguagem selecionada por nosso grupo é o “Mini Java”, uma linguagem simplificada baseada em Java capaz de realizar operações matemáticas, lógicas, definir funções e classes com atributos e métodos, entre outras características básicas.

Um exemplo de função implementada na linguagem é o System.out.println() que, diferente do Java, só pode imprimir números e não é oriundo de uma classe. O termo inteiro é o nome da função, para se assemelhar ao Java em aparência mas não necessariamente implementar todas as suas bibliotecas.

Github

https://github.com/Lucas-Coutinho-Cunha/Trabalho-de-Compiladores-Mini-Java

Calculadora

Pasta: CalculadoraScanner
Casos de teste: 3

Mini Java

Pasta: MiniJavaScanner
Casos de teste: 7

Dificuldades:

Durante a resolução do trabalho encontramos muita dificuldade em rodar o Jflex e entender o seu funcionamento. Porém, depois de fazer o scanner para a calculadora ficamos mais confortáveis em criar o scanner do MiniJava. Além disso, montar a lógica da identificação dos tokens também foi algo que demandou tempo para entender.

Comandos para execução do scanner da Calculadora e do MiniJava:

```
javac nomeScanner.java
java nomeScanner.java nomeTeste.txt
```

Arquivo de definição do scanner da Calculadora e do MiniJava

Vale ressaltar que definimos dois métodos privados para auxiliar no salvamento dos tokens no arquivo de saída.

Calculadora

Os teste 1 e 2 são caminhos felizes. Sem problemas na definição. No teste 3 temos um exemplo de erro com um valor não definido($) o que leva a gerar um token para erro.

```
import java.io.*;

%%
%class CalculadoraScanner
%unicode
%public
%standalone
%line
%column
```

```
%{
    private PrintWriter arquivo;

    private void escreverToken(String conteudo, String tipo) {
        arquivo.println("<" + tipo + ", \"" + conteudo + "\">");
    }

    private void escreverError(String conteudo) {
        arquivo.println("<ERRO, \"" + conteudo + "\", linha=" + (yyline+1) + ",
posição=" + (yycolumn+1) + ">");
    }
%}

%init{
    try {
        arquivo = new PrintWriter(new FileWriter("saida.txt"));
    } catch (IOException e) {
        throw new RuntimeException("Erro ao abrir arquivo de saída", e);
    }
%init}

%eof{
    if (arquivo != null) {
        arquivo.close();
        System.out.println("Tokens salvos em saida.txt");
    }
%eof}

%%

// OPERADORES
"**"    { escreverToken(yytext(), "POTÊNCIA"); }
"//"    { escreverToken(yytext(), "RESTO"); }
"/"     { escreverToken(yytext(), "DIVISÃO"); }
"+"     { escreverToken(yytext(), "MAIS"); }
"-"     { escreverToken(yytext(), "MENOS"); }
"*"     { escreverToken(yytext(), "VEZES"); }
"("     { escreverToken(yytext(), "A-PAREN"); }
")"     { escreverToken(yytext(), "F-PAREN"); }

// NÚMEROS
[0-9]+              { escreverToken(yytext(), "INT"); }
```

```
[0-9]+\.[0-9]+        { escreverToken(yytext(), "FLOAT"); }

// ESPAÇOS
[ \t\r\n\f]+          { /* ignora */ }

// ERRO
.       { escreverError(yytext()); }
```

MiniJava

Vale ressaltar que estávamos em dúvida sobre como definir o “System.out.println”. Em alguns artigos definiram como palavra reservada e em outros sites definiram como identificador. Seguindo o material de referência, decidimos definir como palavra reservada.
Nos testes 2, 3, 6 e 7 temos exemplos de caminhos felizes. Nos testes 1, 4 e 5 definimos alguns casos para ocorrer o erro.
No teste 1 retornam dois tokens com erro na linha 16 porque nesse caso definimos o caractere “>” que não está contido na linguagem do MiniJava.
No teste 4 retorna um token com erro porque o scanner identifica que uma string foi aberta mas não foi fechada.
No teste 5 retorna um token com erro porque o scanner identifica que um comentário foi aberto mas não foi fechado.

```
import java.io.*;

%%

%class MiniJavaScanner
%unicode
%public
%standalone
%line
%column

%{
    private PrintWriter arquivo;

    private void escreverToken(String tipo, String conteudo) {
        arquivo.println("<" + tipo + ", \"" + conteudo + "\">");
    }

    private void escreverError(String conteudo) {
        arquivo.println("<ERROR, \"" + conteudo + "\", linha=" + (yyline + 1) + ",
```

```
coluna=" + (yycolumn + 1) + ">");
    }
%}

%init{
    try {
        arquivo = new PrintWriter(new FileWriter("saida.txt"));
    } catch (IOException e) {
        throw new RuntimeException("Erro ao abrir arquivo de saída", e);
    }
%init}

%eof{
      if (yystate() == IN_COMMENT) {
        escreverError("Comentario de bloco nao fechado");
    }
    if (arquivo != null) {
        arquivo.close();
        System.out.println("Tokens salvos em saida.txt");
    }
%eof}

%state IN_COMMENT

%%

// ---------- PALAVRAS-RESERVADAS ----------
"class"             { escreverToken("CLASS", yytext()); }
"public"            { escreverToken("PUBLIC", yytext()); }
"static"            { escreverToken("STATIC", yytext()); }
"void"              { escreverToken("VOID", yytext()); }
"main"              { escreverToken("MAIN", yytext()); }
"String"            { escreverToken("STRING", yytext()); }
"extends"           { escreverToken("EXTENDS", yytext()); }
"return"            { escreverToken("RETURN", yytext()); }
"int"               { escreverToken("INT", yytext()); }
"boolean"           { escreverToken("BOOLEAN", yytext()); }
"if"                { escreverToken("IF", yytext()); }
"else"              { escreverToken("ELSE", yytext()); }
"while"             { escreverToken("WHILE", yytext()); }
"System.out.println" { escreverToken("PRINT", yytext()); }
"true"              { escreverToken("TRUE", yytext()); }
"false"             { escreverToken("FALSE", yytext()); }
"this"              { escreverToken("THIS", yytext()); }
"new"               { escreverToken("NEW", yytext()); }
"length"            { escreverToken("LENGTH", yytext()); }
```

```
// ---------- OPERADORES E SÍMBOLOS ----------
"="                 { escreverToken("ATRIB", yytext()); }
"=="                { escreverToken("IGUAL", yytext()); }
"!="                { escreverToken("DIFERENTE", yytext()); }
"<="                { escreverToken("MENOR-IGUAL", yytext()); }
"<"                 { escreverToken("MENORQ", yytext()); }
"&&"                { escreverToken("E", yytext()); }
"||"                { escreverToken("OU", yytext()); }
"+"                 { escreverToken("MAIS", yytext()); }
"-"                 { escreverToken("MENOS", yytext()); }
"*"                 { escreverToken("VEZES", yytext()); }
"/"                 { escreverToken("DIV", yytext()); }
"%"                 { escreverToken("MOD", yytext()); }
"!"                 { escreverToken("NEGACAO", yytext()); }
"{"                 { escreverToken("A-CHAVE", yytext()); }
"}"                 { escreverToken("F-CHAVE", yytext()); }
"("                 { escreverToken("A-PAREN", yytext()); }
")"                 { escreverToken("F-PAREN", yytext()); }
"["                 { escreverToken("A-COLC", yytext()); }
"]"                 { escreverToken("F-COLC", yytext()); }
";"                 { escreverToken("PONTO-VIRGULA", yytext()); }
"."                 { escreverToken("PONTO", yytext()); }

// ---------- IDENTIFICADORES E NÚMEROS ----------
[0-9]+              { escreverToken("NUM", yytext()); }
[a-zA-Z][a-zA-Z0-9_]* { escreverToken("ID", yytext());}

// ---------- STRINGS ----------
\"([^\"\\\n]|\\.)*\"   { escreverToken("TEXTO", yytext()); }

\"([^\"\\\n]|\\.)*     { escreverError("String não fechada: " + yytext()); }

// ---------- ESPAÇOS E COMENTÁRIOS ----------
[ \t\r\n\f]+        { /* ignora */ }
"//" [^\n]*         { /* ignora */ }
"/*"                { yybegin(IN_COMMENT); }

<IN_COMMENT> {
    "*/"            { yybegin(YYINITIAL); }
    [^*]+           { /* ignora */ }
    "*"             { /* ignora */ }
}

// ---------- ERROS ----------
. { escreverError(yytext()); }
```

Referências:

ic.ufrj.br/~fabiom/comp/minijava.html