

UNIVERSIDADE DE COIMBRA

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

COMPILADORES

---

# Compilador para linguagem iJava

---

*Autor:*

David CARDOSO

Número: 2011164039

*Autor:*

Bruno CACEIRO

Número: 2008107991

May 26, 2014

# 1 Introdução

Este projecto consiste no desenvolvimento de um compilador para a linguagem *iJava* (imperative Java), que consiste num pequeno subconjunto da linguagem Java (versão 5.0). Os programas da linguagem *iJava* são constituídos por uma única classe (a principal), contendo necessariamente um método *main*, e podendo conter outros métodos e atributos, todos eles estáticos e (possivelmente) públicos.

O projecto foi estruturado em 3 fases, primeiramente foi feita a Análise Lexical, implementada na linguagem *C* e utilizando a ferramenta *lex*. A segunda fase consistiu na análise sintática, com a construção da árvore de sintaxe abstrata e análise semântica (tabelas de símbolos, deteção de erros semânticos). No final foi feita a geração de código.

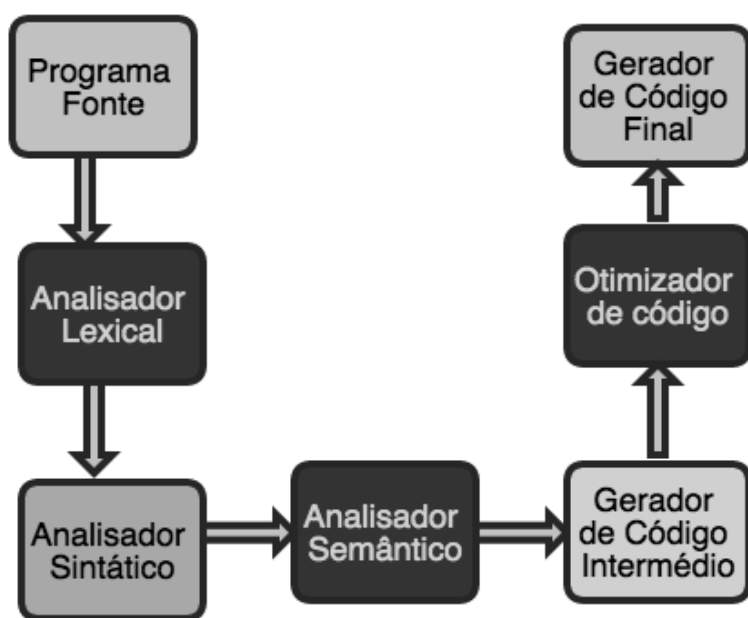


Figure 1: Fases de Compilação

## 2 Análise Lexical

A Análise Lexical consiste em analisar a entrada de linhas de caracteres e produzir uma sequência de símbolos (*tokens*) que podem ser manipulados mais facilmente por um *parser*. É uma forma de verificar um determinado alfabeto, neste caso o alfabeto da linguagem *iJava*. Esta análise pode ser dividida em três fases:

- Extração e classificação de *tokens*;
- Eliminação de delimitadores e comentários;
- Tratamento de erros;

## 2.1 Tokens

- **ID:** Sequências alfanuméricas começadas por uma letra, onde os símbolos "\_" e "\$" contam como letras. Maiúsculas e minúsculas são consideradas letras diferentes
- **INTLIT:** Sequências de dígitos decimais e sequências de dígitos hexadecimais (incluindo a-f e A-F) precedidas de "0x"
- **BOOLLIT:** "true" | "false"
- **INT:** "int"
- **BOOL:** "boolean"
- **NEW:** "new"
- **IF:** "if"
- **ELSE:** "else"
- **WHILE:** "while"
- **PRINT:** "System.out.println"
- **PARSEINT:** "Integer.parseInt"
- **CLASS:** "class"
- **PUBLIC:** "public"
- **STATIC:** "static"
- **VOID:** "void"
- **STRING:** "String"
- **DOTLENGTH:** ".length"
- **RETURN:** "return"
- **OCURV:** "("
- **CCURV:** ")"
- **OBRACE:** "{"
- **CBRACE:** "}"
- **OSQUARE:** "["
- **CSQUARE:** "]"

Foi necessário separar alguns *tokens* devido às diferentes prioridades que cada operador tem.

- **OP1:** "&&"
- **OP1OR:** "|"
- **OP2:** "<" | ">" | "<=" | ">="
- **OP2EQS:** "==" | "!="
- **OP3:** "+" | "-"
- **OP4:** "\*" | "/" | "%"
- **NOT:** "!"
- **ASSIGN:** "="
- **SEMIC:** ";"
- **COMMA:** ","
- **RESERVED:** O *iJava* é um subconjunto da linguagem *Java*, como tal, existe um conjunto de funcionalidades que embora não sejam suportadas, têm de ser consideradas. Assim, foi necessário tratar todo um conjunto de palavras reservadas de forma a permitir que sejam lexicalmente válidas mas não sintaticamente.
  - abstract | assert | break | byte | case | catch | char | const | continue | default | do | double | enum | extends | final | finally | float | for | goto | implements | import | instanceof | interface | long | native | package | private | protected | short | strictfp | super | switch | synchronized | this | throw | throws | transient | try | volatile | null | ++ | -

### 2.1.1 Tratamento de Erros

Se forem detectados erros lexicais no ficheiro de entrada então é impressa uma mensagem de erro no *stdout*:

- "Line<num linha>,col<num coluna>:illegal character('<c >'\n)"
- "Line<num linha>,col<num coluna>:unterminated comment\n"

## 3 Análise Sintática e Semântica

### 3.1 Gramática

A gramática é a maneira formal de especificar a sintaxe de uma linguagem. Desenvolver uma gramática não ambígua é um dos passos mais importantes para o sucesso do compilador. Para a gramática da linguagem *iJava* usámos a notação **EBNF** (*Extended Backus*

*Naur Form*). A gramática que nos foi dada era ambígua e por isso tivémos de efectuar diversas alterações para permitir a análise sintática ascendente com o *yacc*.

$\text{START} \longrightarrow \text{CLASS ID OBRACE field\_or\_method\_declaration CBRACE}$

$\text{field\_or\_method\_declaration} \longrightarrow \text{FieldDecl field\_or\_method\_declaration}$   
 $\mid \text{MethodDecl field\_or\_method\_declaration}$

$\text{FieldDecl} \longrightarrow \text{STATIC VarDecl VarDecl\_REPETITION}$

$\text{MethodDecl} \longrightarrow \text{PUBLIC STATIC method\_type\_declaration ID OCURV FormalParams CCURV OBRACE VarDecl\_REPETITION statement\_declaration\_REPETITION CBRACE}$

$\text{method\_type\_declaration} \longrightarrow \text{Type}$   
 $\mid \text{VOID}$

$\text{FormalParams} \longrightarrow \text{Type ID several\_FormalParams}$   
 $\mid \text{STRING OSQUARE CSQUARE ID}$

$\text{several\_FormalParams} \longrightarrow \text{COMMA Type ID several\_FormalParams}$

$\text{VarDecl\_REPETITION} \longrightarrow \text{VarDecl VarDecl\_REPETITION}$

$\text{VarDecl} \longrightarrow \text{Type ID several\_var\_decl\_in\_same\_instruction OPTIONAL SEMIC}$

$\text{several\_var\_decl\_in\_same\_instruction OPTIONAL} \longrightarrow \text{COMMA ID several\_var\_decl\_in\_same\_instruction}$

$\text{Type} \longrightarrow \text{INT OSQUARE CSQUARE}$   
 $\mid \text{BOOL OSQUARE CSQUARE}$   
 $\mid \text{INT}$   
 $\mid \text{BOOL}$

$\text{statement\_declaration\_REPETITION} \longrightarrow \text{Statement statement\_declaration\_REPETITION}$

$\text{Statement} \longrightarrow \text{OBRACE several\_statement CBRACE}$

| *IF OCURV* Expr *CCURV* Statement *%prec IFX*  
 | *IF OCURV* Expr *CCURV* Statement *ELSE* Statement  
 | *WHILE OCURV* Expr *CCURV* Statement  
 | *PRINT OCURV* Expr *CCURV SEMIC*  
 | *ID* array\_index *OPTIONAL ASSIGN* Expr *SEMIC*  
 | *RETURN* return\_expression *SEMIC*

several\_statement  $\longrightarrow$  Statement several\_statement

array\_index *OPTIONAL*  $\longrightarrow$  *OSQUARE* Expr *CSQUARE*

return\_expression  $\longrightarrow$  Expr

IndexableExpr  $\longrightarrow$  *ID*

| *INTLIT BOOLLIT ID OCURV* Args\_ *OPTIONAL CCURV*  
 | *OCURV* Expr *CCURV*  
 | Expr *DOTLENGTH*  
 | IndexableExpr *OSQUARE* Expr *CSQUARE*  
 | *PARSEINT OCURV ID OSQUARE* Expr *CSQUARE CCURV*

Expr  $\longrightarrow$  Expr *OP1* Expr *%prec OP1*

| Expr *OP1OR* Expr *%prec OP1OR*  
 | Expr *OP4* Expr *%prec OP4*  
 | Expr *OP3* Expr *%prec OP3*  
 | Expr *OP2* Expr *%prec OP2*  
 | Expr *OP2EQS* Expr *%prec OP2EQS*  
 | *OP3* Expr *%prec NOT*  
 | *NOT* Expr *%prec NOT*  
 | *NEW INT OSQUARE* Expr *CSQUARE*  
 | *NEW BOOL OSQUARE* Expr *CSQUARE*  
 | IndexableExpr

Args\_ *OPTIONAL*  $\longrightarrow$  Args

Args  $\longrightarrow$  Expr comma\_expr

comma\_expr  $\longrightarrow$  *COMMA* Expr comma\_expr

- 3.2 Árvore de Sintaxe Abstrata
- 3.3 Tratamento de Erros Lexicais
- 3.4 Análise Semântica
- 3.5 Tabela de Símbolos
- 3.6 Tratamento de Erros Semânticos