

FrontEnd

Documentação

Desenvolvimento

O objetivo deste FrontEnd é construir um compilador que produza código intermediário (quádruplas) para um dado programa escrito numa linguagem que consiste dos seguintes terminais, não-terminais e produções:

<i>program</i> → <i>block</i>	
<i>block</i> → { <i>decls stmts</i> }	
<i>decls</i> → <i>decls decl</i> ϵ	
<i>decl</i> → <i>type id</i> ;	<i>bool</i> → <i>bool</i> <i>join</i> <i>join</i>
<i>type</i> → <i>type</i> [num] basic	<i>join</i> → <i>join</i> && <i>equality</i> <i>equality</i>
<i>stmts</i> → <i>stmts stmt</i> ϵ	<i>equality</i> → <i>equality</i> == <i>rel</i> <i>equality</i> != <i>rel</i> <i>rel</i>
	<i>rel</i> → <i>expr</i> < <i>expr</i> <i>expr</i> <= <i>expr</i> <i>expr</i> >= <i>expr</i>
	<i>expr</i> > <i>expr</i> <i>expr</i>
<i>stmt</i> → <i>loc</i> = <i>bool</i> ;	<i>expr</i> → <i>expr</i> + <i>term</i> <i>expr</i> - <i>term</i> <i>term</i>
if (<i>bool</i>) <i>stmt</i>	<i>term</i> → <i>term</i> * <i>unary</i> <i>term</i> / <i>unary</i> <i>unary</i>
if (<i>bool</i>) <i>stmt</i> else <i>stmt</i>	<i>unary</i> → ! <i>unary</i> - <i>unary</i> <i>factor</i>
while (<i>bool</i>) <i>stmt</i>	<i>factor</i> → (<i>bool</i>) <i>loc</i> num real true false
do <i>stmt</i> while (<i>bool</i>) ;	
break ;	
<i>block</i>	
<i>loc</i> → <i>loc</i> [<i>bool</i>] id	

Figura1: Produções da Linguagem

O token **basic** representa os tipos básicos: **int**, **char**, **float** e **bool**.

A linguagem Java e a orientação ao objeto foram escolhidos para gerar o código do compilador. A maior parte dele foi extraída do livro do dragão [1]. Em nível de pacotes, temos a seguinte dependência:

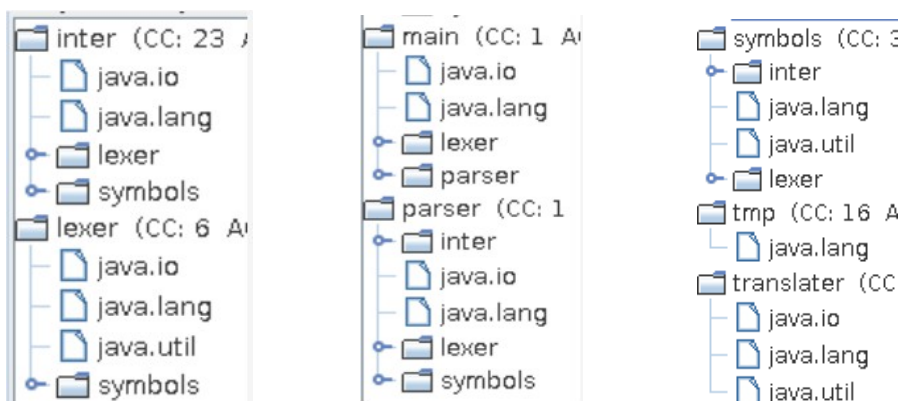


Figura2: Dependência De

Estrutura de dados

A estrutura de dados mais relevante é a tabela de símbolos. Ela foi construída usando a classe `Hashtable` em `src/symbols/Env`, consiste de dois templates: *Token* e *Id*. O primeiro definido em `src/lexer` e o segundo em `src/inter`. *Id* é uma sub-classe de *Expr*, que por sua vez, herda de *Node*.

Utilização

Em um terminal, de uma máquina *linux* com *java* [4] e *make* instalados, descompacte o arquivo **tpfinal.zip** enviado. Em seguida dentro da pasta **src**, digite ***make compile*** para compilar, ou simplesmente, ***make***, para compilar e realizar alguns testes básicos (além de traduzir as quádruplas para o bytecode java, assunto da próxima seção). Abaixo, na Figura3, o conteúdo do Makefile:

```
build: compile test translate

compile:
    javac lexer/*.java
    javac symbols/*.java
    javac inter/*.java
    javac parser/*.java
    javac main/*.java
    javac translator/Tradutor.java

test:
    @for i in `(cd tests; ls *.txt | sed -e 's/.txt$$//')`; \
    do echo $$i.txt in $$i.i; \
    java main.Main <tests/$$i.txt >tmp/$$i.i; \
    done

translate:
    @for i in `(cd tests; ls *.txt | sed -e 's/.txt$$//')`; \
    do echo $$i.i in $$i.java; \
    java translator.Tradutor tmp/$$i.i; \
    javac $$i.java; \
    mv *.class *.java tmp/; \
    done

clean:
    (cd lexer; rm *.class)
    (cd symbols; rm *.class)
    (cd inter; rm *.class)
    (cd parser; rm *.class)
    (cd main; rm *.class)
    (cd translator; rm *.class)
```

Figura3: Arquivo Makefile

Implementação

Gerando código para as declarações e apresentando a listagem do fonte

O código das declarações pertence ao pacote *Symbols*, em `src/symbols`, temos na classe *Env* a responsabilidade por mapear tokens de palavra a objetos da classe *Id*. *Type* define os quatro tipos básicos da linguagem, utilizando-se da classe *Word*. O tipo arranjo, construído na linguagem fonte, é uma extensão de *Type*.

Gerando código para os comandos e apresentando a listagem do fonte

Os comandos são tratados usando a subclasse de *Node*: *Stmt*. Os campos para os componentes de

uma construção de comando estão na subclasse relacionada: a classe *While* possui campos para uma expressão de teste e um subcomando. O código está em `src/inter/Stmt.java`, bem como as classes componentes estão em `src/inter/`.

Gerando código para as expressões e apresentando a listagem do fonte

As expressões são tratadas usando a subclasse de *Node*: *Expr*. Alguns dos métodos de *Expr* tratam booleanos e o código de desvio. A classe possui os campos *op* e *type*, representando o operador e o tipo de um nó. O código está em `src/inter/Expr.java`.

Resultados

Listagem dos programas testes submetidos ao compilador

Ao todo foram submetidos 15 arquivos para teste do compilador, sendo que todos eles resultaram em código intermediário (e bytecode de java) satisfatório(s). Na Figura4 estão os nomes e no Anexo1 estão os códigos desses programas.

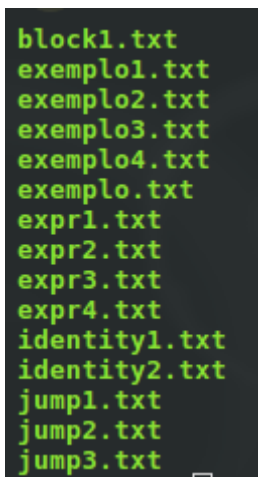
A imagem mostra uma lista de arquivos de texto em uma interface de terminal ou editor de código. Os arquivos listados são: block1.txt, exemplo1.txt, exemplo2.txt, exemplo3.txt, exemplo4.txt, exemplo.txt, expr1.txt, expr2.txt, expr3.txt, expr4.txt, identity1.txt, identity2.txt, jump1.txt, jump2.txt, e jump3.txt. A lista é apresentada em uma única coluna, com cada nome de arquivo em uma nova linha.

Figura4: Listagem dos programas submetidos ao compilador

Apresentar saída das quádruplas geradas

As quádruplas geradas possuem os mesmos nomes dos programas, apenas com a extensão alterada para *i*. Esses códigos intermediários estão no Anexo2.

Tradutor

Documentação

Desenvolvimento

A tradução das quádruplas para o bytecode de java, extensão class, foi feita transformando as quádruplas em código java e executando o comando javac. A implementação do Tradutor foi realizada na linguagem java.

Durante o processo de tradução encontrou-se a dificuldade de o tipo das variáveis não ser declarado nas quádruplas. A solução encontrada foi uma detecção dinâmica (não muito sofisticada) do tipo. Para os casos em que o tipo não é determinado por essa estratégia assumiu-se como padrão o tipo básico `int`.

Estrutura de dados

Para realizar a tradução foi necessário construir um conjunto HashSet com o nome das *variáveis*, um dicionário HashMap para os *tipos* da variável e um outro dicionário para os *vínculos* entre as variáveis. Essas três estruturas permitiram definir e declarar as variáveis do código intermediário.

Utilização

Para utilizar o Tradutor, existe uma instrução no arquivo Makefile: *translate*, que gera o arquivo java usando a classe Tradutor e que em seguida compila esse java com o comando javac, gerando assim, o bytecode class.

Implementação e Resultados

Tradução de todas as quadruplas?

Para realizar o objetivo de *traduzir todas as quádruplas* foi feito um mapeamento do código intermediário em um “**while** com **switch**” da linguagem java. Para cada **Label** traduziu-se um **case**.

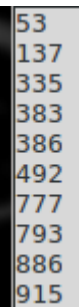
Ainda introduziu-se um variável booleana **stop** de controle do while, que se torna **true** com o término do programa original. Para realizar os efeitos do **goto** adaptou-se uma variável **gotoL** que recebe o valor do próximo **case** em questão.

Gerando código para os comandos e apresentando a listagem do fonte

A classe *Tradutor* opera sobre cada linha do código intermediário produzindo uma equivalente para a linguagem java. Ao final tendo esse código traduzido para java. Utiliza-se o *javac* para gerar os bytecodes.

Listagem dos programas testes submetidos ao compilador

Os programas do Anexo2, no formato **i** foram submetidos a tradução, no Anexo3 encontram-se os arquivos **java** gerados e o código hexadecimal dos bytecodes **class** de apenas alguns deles. Ao lado, a saída correta para o programa Merge.class, executado com **java Merge**. É correta no sentido de colocar em ordem crescente os números do vetor fornecido. Observação: a função para imprimir foi inserida no Merge.java para conseguirmos verificar tal resultado. Além disso inserimos também a capacidade de imprimir os **cases** que foram executados e a ordem em que aconteceram.



53
137
335
383
386
492
777
793
886
915

Compilador Integrado

Documentação

Desenvolvimento e Conclusão

O FrontEnd e o Tradutor permitiram:

1. compreender as fases lógicas de compilação.
2. entender a possibilidade de gerar código para diferentes BackEnds a partir do código intermediário.
3. fazer uso da organização modular e orientada a objeto do código com o objetivo de torná-lo mais organizado e eficiente.

Esse compilador integrado concretizou-se um ótimo exercício para plicar os conhecimentos adquiridos no curso de Compiladores1.

Utilização

Para utilizar o compilador integrado vale o **Makefile** exibido na seção 1.1.3, sendo necessário apenas executar **make** para gerar as quadruplas e os bytecodes class dos arquivos com extensão **txt** que estão dentro da pasta **src/tests**. Os arquivos produzidos, com extensões **i**, **java** e **class**, serão armazenados na pasta **src/tmp**.

Implementação e Resultados

Programas testes convincentes

Para exemplificar o processo completo do compilador integrado selecionou-se dois dos programas de testes: **exemplo** e **merge**.

```
{
    int i; int j; float v; float x; float[100] a;
    while( true ) {
        do i = i+1; while( a[i] < v);
        do j = j-1; while( a[j] > v);
        if( i >= j ) break;
        x = a[i]; a[i] = a[j]; a[j] = x;
    }
}
```

Figura4: exemplo.txt

```
{
    int i; int t; int k; int l; int r; int u; int im;
    int jm; int km; int[100000] a; int[100000] b; int n;

    a[1]=886; a[2]=777; a[3]=915; a[4]=793; a[5]=335;
    a[6]=386; a[7]=492; a[8]=137; a[9]=53; a[10]=383; n=10; k=1;

    while (k<n) {
        i=1;
        while (i+k<=n) {
            u=i+k*2;
            if (u>n) u=n+1;
            l=i; r=i+k; im=l; jm=r; km=l;
            while (im<r && jm<u) {
                if (a[im]<=a[jm]) {b[km]=a[im]; im=im+1;}
                else {b[km]=a[jm]; jm=jm+1;}
                km=km+1;
            }
            while (im<r) {
                b[km]=a[im]; im=im+1; km=km+1;
            }
            while (jm<u) {
                b[km]=a[jm]; jm=jm+1; km=km+1;
            }
            km=l;
            while (km<u) {
                a[km]=b[km];
                km=km+1;
            }
            i=i+k*2;
        }
        k=k*2;
    }
}
```

Figura5: merge.txt

Listagem dos programas testes submetidos ao compilador

A lista completa dos programas de teste é a seguinte:

```
mfer@hobbit:~/Documentos/bcc/cl/tpfinal/src$ make
javac lexer/*.java
javac symbols/*.java
javac inter/*.java
javac parser/*.java
javac main/*.java
javac translator/Tradutor.java
block1.txt in block1.i
exemplo1.txt in exemplo1.i
exemplo2.txt in exemplo2.i
exemplo3.txt in exemplo3.i
exemplo4.txt in exemplo4.i
exemplo.txt in exemplo.i
expr1.txt in expr1.i
expr2.txt in expr2.i
expr3.txt in expr3.i
expr4.txt in expr4.i
identity1.txt in identity1.i
identity2.txt in identity2.i
jump1.txt in jump1.i
jump2.txt in jump2.i
jump3.txt in jump3.i
merge.txt in merge.i
block1.i in block1.java
exemplo1.i in exemplo1.java
exemplo2.i in exemplo2.java
exemplo3.i in exemplo3.java
exemplo4.i in exemplo4.java
exemplo.i in exemplo.java
expr1.i in expr1.java
expr2.i in expr2.java
expr3.i in expr3.java
expr4.i in expr4.java
identity1.i in identity1.java
identity2.i in identity2.java
jump1.i in jump1.java
jump2.i in jump2.java
jump3.i in jump3.java
merge.i in merge.java
mfer@hobbit:~/Documentos/bcc/cl/tpfinal/src$ ls tmp/
Block1.class      exemplo2.i      exemplo4.java  Expr2.class     expr4.i         identity2.java  Jump3.class
block1.i          exemplo2.java   Exemplo.class  expr2.i         expr4.java      Jump1.class     jump3.i
block1.java       Exemplo3.class  exemplo.i      expr2.java      Identity1.class  jump1.i         jump3.java
Exemplo1.class    exemplo3.i      exemplo.java   Expr3.class     identity1.i     jump1.java      Merge.class
exemplo1.i        exemplo3.java   Expr1.class    expr3.i         identity1.java  Jump2.class     merge.i
exemplo1.java     Exemplo4.class  expr1.i        expr3.java      Identity2.class  jump2.i         merge.java
Exemplo2.class    exemplo4.i      expr1.java     Expr4.class     identity2.i     jump2.java
```

Figura9: merge.java

Abaixo seguem: o código intermediário **i**, o código **java** e o bytecode **class**, arquivos gerados para os dois exemplos da seção anterior.

```
L1:L3:  i = i + 1
L5:    t1 = i * 8
        t2 = a [ t1 ]
        if t2 < v goto L3
L4:    j = j - 1
L7:    t3 = j * 8
        t4 = a [ t3 ]
        if t4 > v goto L4
L6:    iffalse i >= j goto L8
L9:    goto L2
L8:    t5 = i * 8
        x = a [ t5 ]
L10:   t6 = i * 8
        t7 = j * 8
        t8 = a [ t7 ]
        a [ t6 ] = t8
L11:   t9 = j * 8
        a [ t9 ] = x
        goto L1
L2:
```

Figura6: exemplo.i

```

L1: t1 = 1 * 4
    a [ t1 ] = 886
L3: t2 = 2 * 4
    a [ t2 ] = 777
L4: t3 = 3 * 4
    a [ t3 ] = 915
L5: t4 = 4 * 4
    a [ t4 ] = 793
L6: t5 = 5 * 4
    a [ t5 ] = 335
L7: t6 = 6 * 4
    a [ t6 ] = 386
L8: t7 = 7 * 4
    a [ t7 ] = 492
L9: t8 = 8 * 4
    a [ t8 ] = 137
L10: t9 = 9 * 4
    a [ t9 ] = 53
L11: t10 = 10 * 4
    a [ t10 ] = 383
L12: n = 10
L13: k = 1
L14: iffalse k < n goto L2
L15: i = 1
L16: t11 = i + k
    iffalse t11 <= n goto L17
L18: t12 = k * 2
    u = i + t12
L19: iffalse u > n goto L20
L21: u = n + 1
L20: l = i
L22: r = i + k
L23: im = l
L24: jm = r
L25: km = l
L26: iffalse im < r goto L27
    iffalse jm < u goto L27
L28: t13 = im * 4
    t14 = a [ t13 ]
    t15 = jm * 4
    t16 = a [ t15 ]
    iffalse t14 <= t16 goto L31
L30: t17 = km * 4
    t18 = im * 4
    t19 = a [ t18 ]
    b [ t17 ] = t19

```

```

L32: im = im + 1
    goto L29
L31: t20 = km * 4
    t21 = jm * 4
    t22 = a [ t21 ]
    b [ t20 ] = t22
L33: jm = jm + 1
L29: km = km + 1
    goto L26
L27: iffalse im < r goto L34
L35: t23 = km * 4
    t24 = im * 4
    t25 = a [ t24 ]
    b [ t23 ] = t25
L36: im = im + 1
L37: km = km + 1
    goto L27
L34: iffalse jm < u goto L38
L39: t26 = km * 4
    t27 = jm * 4
    t28 = a [ t27 ]
    b [ t26 ] = t28
L40: jm = jm + 1
L41: km = km + 1
    goto L34
L38: km = l
L42: iffalse km < u goto L43
L44: t29 = km * 4
    t30 = km * 4
    t31 = b [ t30 ]
    a [ t29 ] = t31
L45: km = km + 1
    goto L42
L43: t32 = k * 2
    i = i + t32
    goto L16
L17: k = k * 2
    goto L14
L2:

```

Figura7: merge.i


```

import java.util.Set;
import java.util.HashSet;
import java.util.Iterator;
class Exemplo{public static void main(String[] args) {boolean stop=false;
while(!stop){System.out.println(gotoL);
switch(gotoL){case 1:case 3:i=i+1;
case 5:t1=i*8;
t2=a[t1];
if ( t2 < v ) { gotoL = 3;
break;
}case 4:j=j-1;
case 7:t3=j*8;
t4=a[t3];
if ( t4 > v ) { gotoL = 4;
break;
}case 6:if ( !(i >= j) ) { gotoL = 8;
break;
}case 9:gotoL = 2;
break;
case 8:t5=i*8;
x=a[t5];
case 10:t6=i*8;
t7=j*8;
t8=a[t7];
a[t6]=t8;
case 11:t9=j*8;
a[t9]=x;
gotoL = 1;
break;
case 2:stop = true;
}}}public static int gotoL=1;
public static int t3;
public static int t2;
public static int a[] = new int[10000];
public static int t1;
public static int j;
public static int i;
public static int v;
public static int t4;
public static int t5;
public static int t6;
public static int t7;
public static int t8;
public static int t9;
public static int x;
}

```

Figura8: exemplo.java


```

import java.util.Set;
import java.util.HashSet;
import java.util.Iterator;
class Merge{public static void main(String[] args) {boolean stop=false;
while(!stop){System.out.println(gotoL);
switch(gotoL){case 1:t1=1*4;
a[t1]=886;
case 3:t2=2*4;
a[t2]=777;
case 4:t3=3*4;
a[t3]=915;
case 5:t4=4*4;
a[t4]=793;
case 6:t5=5*4;
a[t5]=335;
case 7:t6=6*4;
a[t6]=386;
case 8:t7=7*4;
a[t7]=492;
case 9:t8=8*4;
a[t8]=137;
case 10:t9=9*4;
a[t9]=53;
case 11:t10=10*4;
a[t10]=383;
case 12:n=10;
case 13:k=1;
case 14:if ( !(k < n) ) { gotoL = 2;
break;
}case 15:i=1;
case 16:t11=i+k;
if ( !(t11 <= n) ) { gotoL = 17;
break;
}case 18:t12=k*2;
u=i+t12;
case 19:if ( !(u > n) ) { gotoL = 20;
break;
}case 21:u=n+1;
case 20:l=i;
case 22:r=i+k;
case 23:im=l;
case 24:jm=r;
case 25:km=l;
case 26:if ( !(im < r) ) { gotoL = 27;
break;
}if ( !(jm < u) ) { gotoL = 27;
break;
}case 28:t13=im*4;
t14=a[t13];
t15=jm*4;
t16=a[t15];
if ( !(t14 <= t16) ) { gotoL = 31;
break;
}case 30:t17=km*4;
t18=im*4;
t19=a[t18];
b[t17]=t19;
case 32:im=im+1;
gotoL = 29;
break;
case 31:t20=km*4;
t21=jm*4;
t22=a[t21];
b[t20]=t22;
case 33:jm=jm+1;
case 29:km=km+1;
gotoL = 26;
break;
case 27:if ( !(im < r) ) { gotoL = 34;
break;
}case 35:t23=km*4;
t24=im*4;
t25=a[t24];
b[t23]=t25;
case 36:im=im+1;
case 37:km=km+1;
gotoL = 27;
break;
case 34:if ( !(jm < u) ) { gotoL = 38;
break;
}case 39:t26=km*4;
t27=jm*4;
t28=a[t27];
t29=jm*4;
t28=a[t27];
b[t26]=t28;
case 40:jm=jm+1;
case 41:km=km+1;
gotoL = 34;
break;
case 38:km=l;
case 42:if ( !(km < u) ) { gotoL = 43;
break;
}case 44:t29=km*4;
t30=km*4;
t31=b[t30];
a[t29]=t31;
case 45:km=km+1;
gotoL = 42;
break;
case 43:t32=k*2;
i=i+t32;
gotoL = 16;
break;
case 17:k=k*2;
gotoL = 14;
break;
case 2:stop = true;

for(int i=1;
i<=10;
i++){System.out.println(a[i*4]);
}
}}public static int gotoL=1;
public static int t20;
public static int t3;
public static int t21;
public static int t2;
public static int t1;
public static int a[] = new int[10000];
public static int t24;
public static int t25;
public static int t22;
public static int t23;
public static int t28;
public static int t29;
public static int t26;
public static int t27;
public static int jm;
public static int b[] = new int[10000];
public static int t10;
public static int t11;
public static int t30;
public static int t12;
public static int t31;
public static int t13;
public static int t32;
public static int t14;
public static int t15;
public static int n;
public static int t16;
public static int t17;
public static int l;
public static int t18;
public static int t19;
public static int im;
public static int k;
public static int km;
public static int i;
public static int u;
public static int r;
public static int t4;
public static int t5;
public static int t6;
public static int t7;
public static int t8;
public static int t9;
}

```

Figura9: merge.java

O arquivo Merge.class não foi colocado aqui por ter ficado muito extenso. E também por ser semelhante ao Exemplo.class, a seguir. Ambos podem ser acessados em <http://goo.gl/C6OGNy>.

```

00000000 CA FE BA BE 00 00 33 00 4C 0A 00 14 00 30 09 00 31 00 32 09 00 13 00 33 0A 00 34 00 35 09 00 13 00
00000022 36 09 00 13 00 37 09 00 13 00 38 09 00 13 00 39 09 00 13 00 3A 09 00 13 00 3B 09 00 13 00 3C 09 00 13
00000044 00 3D 09 00 13 00 3E 09 00 13 00 3F 09 00 13 00 40 09 00 13 00 41 09 00 13 00 42 09 00 13 00 43 07 00
00000066 44 07 00 45 01 00 05 67 6F 74 6F 4C 01 00 01 49 01 00 02 74 33 01 00 02 74 32 01 00 01 61 01 00 02 5B
00000088 49 01 00 02 74 31 01 00 01 6A 01 00 01 69 01 00 01 76 01 00 02 74 34 01 00 02 74 35 01 00 02 74 36 01
000000aa 00 02 74 37 01 00 02 74 38 01 00 02 74 39 01 00 01 78 01 00 06 3C 69 6E 69 74 3E 01 00 03 28 29 56 01
000000cc 00 04 43 6F 64 65 01 00 0F 4C 69 6E 65 4E 75 6D 62 65 72 54 61 62 6C 65 01 00 04 6D 61 69 6E 01 00 16
000000ee 28 5B 4C 6A 61 76 61 2F 6C 61 6E 67 2F 53 74 72 69 6E 67 3B 29 56 01 00 0D 53 74 61 63 6B 4D 61 70 54
00000110 61 62 6C 65 01 00 08 3C 63 6C 69 6E 69 74 3E 01 00 0A 53 6F 75 72 63 65 46 69 6C 65 01 00 0C 65 78 65
00000132 6D 70 6C 6F 2E 6A 61 76 61 0C 00 26 00 27 07 00 46 0C 00 47 00 48 0C 00 15 00 16 07 00 49 0C 00 4A 00
00000154 4B 0C 00 1D 00 16 0C 00 1B 00 16 0C 00 19 00 1A 0C 00 18 00 16 0C 00 1E 00 16 0C 00 1C 00 16 0C 00 17
00000176 00 16 0C 00 1F 00 16 0C 00 20 00 16 0C 00 25 00 16 0C 00 21 00 16 0C 00 22 00 16 0C 00 23 00 16 0C 00
00000198 24 00 16 01 00 07 45 78 65 6D 70 6C 6F 01 00 10 6A 61 76 61 2F 6C 61 6E 67 2F 4F 62 6A 65 63 74 01 00
000001ba 10 6A 61 76 61 2F 6C 61 6E 67 2F 53 79 73 74 65 6D 01 00 03 6F 75 74 01 00 15 4C 6A 61 76 61 2F 69 6F
000001dc 2F 50 72 69 6E 74 53 74 72 65 61 6D 3B 01 00 13 6A 61 76 61 2F 69 6F 2F 50 72 69 6E 74 53 74 72 65 61
000001fe 6D 01 00 07 70 72 69 6E 74 6C 6E 01 00 04 28 49 29 56 00 20 00 13 00 14 00 00 00 0F 00 09 00 15 00 16
00000220 00 00 00 09 00 17 00 16 00 00 00 09 00 18 00 16 00 00 00 09 00 19 00 1A 00 00 00 09 00 1B 00 16 00 00
00000242 00 09 00 1C 00 16 00 00 00 09 00 1D 00 16 00 00 09 00 1E 00 16 00 00 00 09 00 1F 00 16 00 00 00 09
00000264 00 20 00 16 00 00 00 09 00 21 00 16 00 00 00 09 00 22 00 16 00 00 00 09 00 23 00 16 00 00 09 00 24
00000286 00 16 00 00 00 09 00 25 00 16 00 00 00 03 00 00 00 26 00 27 00 01 00 28 00 00 00 1D 00 01 00 01 00 00
000002a8 00 05 2A B7 00 01 B1 00 00 00 01 00 29 00 00 06 00 01 00 00 00 01 00 09 00 2A 00 2B 00 01 00 28 00
000002ca 00 01 45 00 03 00 02 00 00 01 13 03 3C 1B 9A 01 0F B2 00 02 B2 00 03 B6 00 04 B2 00 03 AA 00 00 00 00
000002ec FD 00 00 00 01 00 00 00 0B 00 00 00 3A 00 00 00 FB 00 00 00 3A 00 00 00 65 00 00 00 42 00 00 00 90 00
0000030e 00 00 6D 00 00 00 A8 00 00 00 A1 00 00 00 BB 00 00 00 E1 B2 00 05 04 60 B3 00 05 B2 00 05 10 08 68 B3
00000330 00 06 B2 00 07 B2 00 06 2E B3 00 08 B2 00 08 B2 00 09 A2 00 0A 06 B3 00 03 A7 00 09 B2 00 0A 04 64 B3
00000352 00 0A B2 00 0A 10 08 68 B3 00 0B B2 00 07 B2 00 0B 2E B3 00 0C B2 00 0C B2 00 09 A4 00 0A 07 B3 00 03
00000374 A7 00 70 B2 00 05 B2 00 0A A2 00 0B 10 08 B3 00 03 A7 00 05 B3 00 03 A7 00 58 B2 00 05 10 08 68 B3
00000396 00 0D B2 00 07 B2 00 0D 2E B3 00 0E B2 00 05 10 08 68 B3 00 0F B2 00 0A 10 08 68 B3 00 10 B2 00 07 B2
000003b8 00 10 2E B3 00 11 B2 00 07 B2 00 0F B2 00 11 4F B2 00 0A 10 08 68 B3 00 12 B2 00 07 B2 00 12 B2 00 0E
000003da 4F 04 B3 00 03 A7 00 05 04 3C A7 FE F3 B1 00 00 00 02 00 29 00 00 00 06 00 01 00 00 00 01 00 2C 00 00
000003fc 00 14 00 0D FC 00 02 01 FB 00 49 07 22 07 22 10 06 12 25 19 01 02 00 08 00 2D 00 27 00 01 00 28 00 00
0000041e 00 25 00 01 00 00 00 00 0D 04 B3 00 03 11 27 10 BC 0A B3 00 07 B1 00 00 00 01 00 29 00 00 00 06 00
00000440 01 00 00 00 01 00 01 00 2E 00 00 00 02 00 2F

```

Figura10: Exemplo.class

Referências

[1] AHO, Alfred V., LAM, Monica S., SETHI, Ravi, ULLMAN, Jeffrey D. *Compiladores: Princípios, técnicas e ferramentas*.

<http://dragonbook.stanford.edu/dragon-front-source.tar>

[2] O grafo de dependência foi gerado usando a ferramenta JDepend.

<http://www.clarkware.com/software/JDepend.html>

[3] O código class foi visualizado usando o editor hexadecimal *bless*.

<http://home.gna.org/bless/>

[4] Para instalar o Java da Oracle no Ubuntu, o seguinte script foi utilizado:

```

echo "deb http://ppa.launchpad.net/webupd8team/java/ubuntu precise main" | tee -a /etc/apt/sources.list
echo "deb-src http://ppa.launchpad.net/webupd8team/java/ubuntu precise main" | tee -a /etc/apt/sources.list
apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys EEA14886
apt-get update
apt-get install oracle-java7-installer

```