FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Jednoduchý překladač programů v jazyce Java do jazyka C

KIV/FJP – Semestrální práce

Jindřich Pouba, Marek Šimůnek, Jakub Zíka, Ondřej Hovjacký
Studijní čísla: A15N0072P, A15N0082P, A15N0087P, A15N0062P
E-maily: pouba@students.zcu.cz, simunek@students.zcu.cz, zikaj@students.zcu.cz, ohovjack@students.zcu.cz
Datum: 15. 1. 2016

1 Zadání

Tvorba překladače jazyka Java do jazyka C. Program by si měl poradit s výběrem určitých podporovaných konstrukcí.

2 Programátorská dokumentace

Program nejprve ze zdrojového kódu odstraní všechny komentáře a řetězce nahradí jedinečným symbolem, který bude převeden zpět na daný řetězec až při výpisu. Následně odstraní importy a prázdná místa – k tomuto se využívá třída *Preprocessor*.

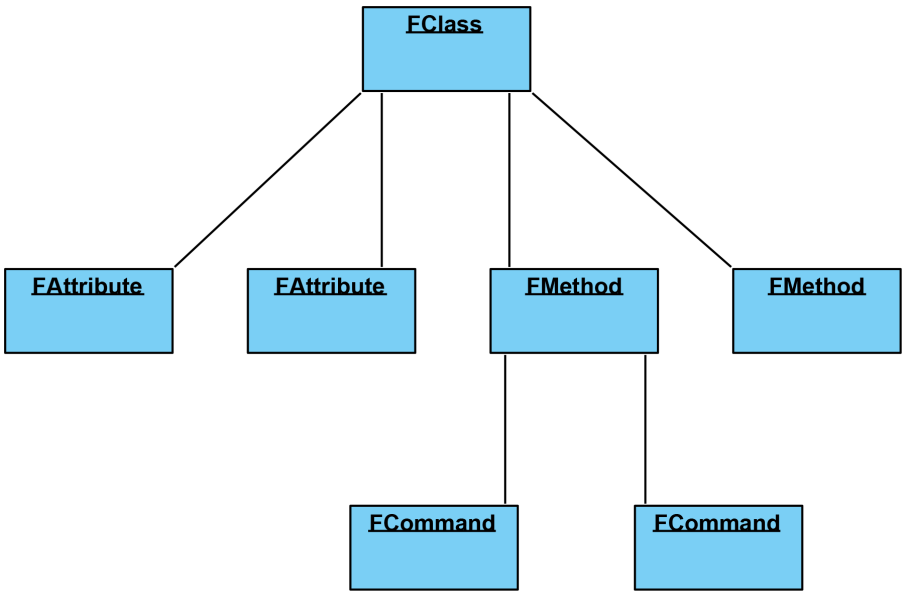
Poté načte jednotlivé symboly (klíčová slova, znaky závorek, středníky, atd. – *Preprocessor.getWords*), které postupně parsuje metodou rekurzivního sestupu.

K započetí algoritmu analýzy slouží třída *FCompilationUnit*, které se předají jednotlivá slova a zavolá se metoda *parse()*. Tado metoda pochází z abstraktní třídy *ParsableObject*, kterou dědí všechny třídy, které tvoří objekty v derivačním stromu (až na koncové, které už nemohou mít žádné potomky).

Základem jsou tedy třídy (*FClass*), které mohou mít konstanty, atributy (*FAttribute*) a metody (*FMethod*). U metody se pak uloží její název, argumenty, typ návratové hodnoty a nakonec se parsuje její tělo.

Objektu se vždy předají všechna slova, která k němu patří (metoda *setWords* z *ParsableObject*) a pak se zavolá metoda *parse*, která si je musí zpracovat. Pokud bude vytvořen další objekt (např. v metodě bude cyklus *for*), předají se mu opět ta slova, která se ho týkají (od slova *for* po uzavírací závorku }).

Tělo metody tvoří blok příkazů (stejně jako např. tělo cyklů), pro získání jednotlivých příkazů (*FCommand*) se použije metoda *CommandBlockParser.parseBlock*, která oddělí příkazy podle středníku nebo podle klíčových slov, pokud jsou to např. cykly. Tyto příkazy jsou uloženy do vnitřní proměnné *FMethod* – *List<FCommand> commands*. Stejně tak i u dalších objektů jsou jejich potomci uloženi do nějaké proměnné, aby se takto vytvořil strom.



Obrázek 1: Příklad části derivačního stromu

Nad metodou *FMethod* je pak opět zavolána metoda *parse*, která prochází jednotlivé příkazy a opět nad nimi volá *parse*, dokud se nedojde k základním entitám nebo už příkaz nejde dál rozdělit.

Mezi příkazy patří (dědí od *FCommand*):

- *FAssignment*
- *FFor*
- *FIf*
- *FMethodCall*
- *FReturn*
- *FSystem*
- *FVarDeclaration*
- *FVarDeclarationWithInitialization*
- *FWhile*

Pokud je někde v kódu deklarace proměnné, její typ je uložen do tabulky proměnných příslušné danému kontextu – svou tabulku má třída, metoda a cykly. Toto se provádí za účelem znalosti typu proměnné při výpisu pomocí funkce *printf*.

Po sestavení stromu se strom začne procházet podruhé (třídy v adresáři *writer*), a jednotlivé entity jsou vypisovány v jazyce C.

2.1 Struktura projektu

V hlavním adresáři je soubor *Main.java* a několik dalších souborů. Další adresáře jsou:

- *enums* – klíčová slova a operátory
- *objects* – objekty používané při tvorbě stromu
  - *commands* – příkazy používané při tvorbě stromu
- *parser* – několik souborů, které provádí hlavní parsování
- *writer* – balík umožňující výpis stromu v jazyce C
  - *commands* – výpis příkazů

2.2 Podpora příkazů

Program si dokáže poradit s následujícími:

- proměnné s jednoduchým datovým typem,
- proměnné typu *Scanner* (pro *System.in*), String (podporované metody viz níže),
- třídy, metody, podmínky (i s *else* větví), cykly, přiřazení, a výpis pomocí *System.out.print(ln)*.

Metody volané nad String objekty jsou následující:

- *charAt*
- *length*
- *equals*
- *indexOf*

2.3 Příklad překladu

Zdrojový kód v jazyce Java:

```
public class StringMethods {
        public static void main(String[] args) {
                String text = "Hello World!";

                char firstLetter = text.charAt(0);
                int size = text.length();
                int isEqual = 0;
                if(text.equals("Hello World!")){
                        isEqual = 1;
                }

                int indexOfW = text.indexOf("W");

                System.out.println("firstLetter - " + firstLetter);
                System.out.println("size - " + size);
                System.out.println("isEqual - " + isEqual);
                System.out.println("indexOfW - " + indexOfW);
        }
}
```

Cílový kód v jazyce C:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

#define STR_LEN 256

char str[STR_LEN];

char* readLine() {
  fgets(str, STR_LEN, stdin);
  str[strlen(str) - 1] = '\0';
  return str;
}


void main(char** args);

void main(char** args) {
char text[STR_LEN];
strcpy(text, "Hello World!" );
char firstLetter = text[0] ;
;
int size = strlen(text);
;
int isEqual = 0 ;
;
if ( strcmp(text, "Hello World!") ) {
isEqual = 1 ;
}


int indexOfW = strcspn(text, "W");
;
printf("firstLetter - %c \n", firstLetter);
printf("size - %d \n", size);
printf("isEqual - %d \n", isEqual);
printf("indexOfW - %d \n", indexOfW);
}
```

3 Uživatelská dokumentace

Pro spuštění použijte přiložený .jar soubor, kterému zadáte jako parametr cestu k souboru, který chcete přeložit. Výstup i s logem se uloží do stejného adresáře, kde se nachází .jar soubor.

4 Závěr

Program je schopný přeložit poměrně velké množství konstrukcí, nicméně bylo by možné ho rozšířit a taky vhodné zkonstruovat nejprve přesnou gramatiku, kterou by měl zvládnout.