Javaによるコンパイラ構築のためのクラスライブラリの設計と実現

5M－9

吉田映彦，並木美太郎
（東京農工大学工学部）

1.はじめに

現在インターネットが普及する中，Java が注目されている．その理由として，Java がプラットフォームに依存しない，オブジェクト指向言語である，といった特徴を有するからである．その特徴からプログラムを配布するのにインターネットを用いることも有用であり，配布することができるプログラムの種類も多岐にわたるが，本研究ではその中でもコンパイラに注目した．コンパイラの動作はプログラミング言語に関わらず共通化できる部分は少なく，従来ではコンパイラを開発する毎に，全ての処理を記述しなければならなかった．また，コンパイラの動作を部分的に拡張したい場合でも，コンパイラ全体の再構築・再配布が必要となっていた．さらに，インターネットを通じての配布を考えると，従来のコンパイラはプラットフォームに依存しているという点で効率が悪い．

2.目標

本研究の目標は，次の三つを達成することである．

(i)コンパイラ用クラスライブラリの提供

コンパイラの作成を補助するコンパイラ用のクラスライブラリを提供することで，新たにコンパイラの作成をする際の手間を軽減させる．本クラスライブラリは，一般のコンパイラに共通する処理を行い，プログラミング言語に依存する部分のみコンパイラ開発者が実装すればよいようにする．

(ii)拡張可能なコンパイラ構成法の提供

コンパイラを作成する上で構文解析の段階を分け，部分的な拡張を行いながらコンパイラを構築していく．すなわち，プログラミング言語の仕様に新たなBNFが追加された場合，その部分のモジュールのみを追加するだけで，他の部分のモジュールには一切の影響を与えることなく，コンパイラを拡張することができるようにする．例えば，数式のみを構文解析できるコンパイラを，条件文を含むプログラムのコンパイラに拡張するといった具合である．

(iii)プラットフォーム独立なコンパイラ

一般に，コンパイラはプラットフォームに依存する形で提供される．本研究では，プラットフォームに依存しないコンパイラの作成を支援できるようにしたい．

3.設計

3.1 設計方針

第 2 章で述べた三つの目標を達成するために，本研究ではコンパイラ開発用のクラスライブラリを Java で実現する．プログラミング言語の種類によっては，プリプロセッサやライブラリを含めてコンパイラということもあるが，ここでいうコンパイラとは，ソースファイルを読み込んで目的語へ翻訳するシステムプログラムを意味し，プリプロセッサやその他ライブラリは含まれない．対象とするプログラミング言語は，LL(k)文法で定義できるものとする．本クラスライブラリは Java を用いて作成するため，Java のプラットフォーム非依存という特

Design and Implementation of Class Libraries for Compiler Construction Using Java
Teruhiko YOSHIDA and Mitaro NAMIKI
Faculty of Technology, Tokyo University of Agriculture and Technology

徴を活かし，第2章の (iii)の目標を達成する．また，オブジェクト指向を活かすことで，(i)，(ii)の目標も達成する．本クラスライブラリは，記号テーブル，字句解析部，構文解析部，Javaクラスファイルというモジュール群からなる．

(1)記号テーブル

テーブルの管理法はコンパイラに依存しない．ただし，テーブルに登録する属性はコンパイラにより異なるので，属性の中身についてはコンパイラ開発者が定義する．

(2)字句解析部

本クラスライブラリで，字句の区切り，数値，基本的な記号の処理を行う．プログラミング言語によって異なる予約語や言語特有の記号についても，テーブルに登録するだけでよい．

(3)構文解析部

構文解析処理はプログラミング言語に依存するため，コンパイラ製作者が作成する必要がある（3.2 を参照）．

3.2 構文解析処理の拡張方式

例として，四則演算からなる式を構文解析するプログラムを段階的に作成する事を考える．そのためには，まず BNF を段階ごとに用意する．図 1 は，乗除演算のみを処理することができる途中の段階であり，図 2 が目的とするものの BNF である．現在，図 1 に示す構文を解析できる Parser が用意されているとする．そこから，図 2 に示す構文を解析するプログラムを記述するには，図 3 に示すよう，Parser をスーパークラスとし，新たに追加すべき非終端記号 expression，および element の拡張されている部分をインプリメントする新たな構文解析クラス ParserEx を作成すればよい．

```
term    ::=element {[*|/] element}*
element ::=digit
```

図 1　乗除演算の BNF

```
expression::=term {[+|-] term}*
term       ::=element {[*|/] element}*
element    ::=digit | ( expression )
```

図 2 加減乗除演算の BNF

```
class ParserEx extends Parser implements Ex{
    public boolean expression(){
        ・・・
    }
    public boolean element(){
            ・・・
        return super.element();
    }
}
```

図 3　構文解析クラスの拡張

4.C--の実現

本クラスライブラリを用いて，言語 C のサブセットである C--[2]を実現した．拡張段階で作成した構文解析クラスは，すべて合わせて 8 個である．拡張の過程であるが，始めから順に乗除演算の処理，加減乗除演算の処理，変数宣言可能なブロックの実現，if 文の導入，while 文の導入，関数の定義の導入，外部変数宣言の導入，入出力命令の追加となっている．

5.おわりに

本稿では，コンパイラ用クラスライブラリの設計と実現について述べ，また構文解析部の拡張の方法について述べた．本クラスライブラリを用いることでコンパイラの作成の手間を減少させることができ，また文法の変更などによる拡張も容易に行うことができる．

参考文献

[1] Tim Lindholm and Frank Yellin,「The Java仮想マシン仕様」，オライリー・ジャパン

[2]並木，早川，高橋，言語処理系の教育のためのプログラミング言語C--とその教材