オブジェクト指向言語Koola

4G－2

－新しい概念と機能－

渡守武 和記[1] 岡崎 薫[2] 濱口 敏英[1] 西山 保[1]
1松下電器産業(株) 半導体研究センター 2映像音響研究センター

1. はじめに

Koola(Kernel Object Oriented Language)[1]は、C言語によるソフトウェア開発において、優れたオブジェクト指向プログラミング環境を提供する高速･高機能なオブジェクト指向言語である。今回は、新たに導入した概念と主な新しい機能とについて報告する。

2. Koolaの概要

KoolaはC言語の上位言語であり、プリプロセッサ方式で実現されている。構文はC言語そのままであり、クラス定義はSmalltalk-80[2]に準拠しているので、記述性･可読性が優れている。また、メッセージ･メソッド間のバインディングを全てコンパイル時に解決してしまうため、C言語の高速性と相まって、非常に高速である。Koolaでは1つのファイルに1つのクラスを定義する。

3. 新たに導入した概念－タイプクラス－

C言語をベースにしたオブジェクト指向言語で問題になるのは、C言語の基本データタイプであるintやcharなどのデータをオブジェクトとした記述ができないことである。このため、メッセージセンディングと関数コールとが入り交じり、プログラムが非常に読みにくくなる。

Koolaではこれを解決するために、クラスに類似したタイプクラスという概念を新たに導入した。普通のクラスとタイプクラスの本質的な違いは、メッセージとメソッドのバインディングが前者では動的であり、後者では静的であることにある。故にタイプクラスは、高速ではあるが、抽象クラスを定義できない。しかし、インスタンス変数やメソッドの継承は行われる。

タイプクラスを使用すると、次のようなことが可能になる。

①プリミティブな構造データの効率的な実現
→ Complex, Point, Rectangle等の効率的実現
②基本データタイプのオブジェクト化
→ プログラムの完全なオブジェクト指向記述
③メソッドのマクロ定義
→ 高速
→ ライブラリ関数コールのメッセージ化

①の機能を提供するタイプクラスを一般タイプクラス、②の機能を提供するタイプクラスを基本タイプクラスと呼んでいる。以下、順に説明する。

(1) 一般タイプクラス

定義例を図1に示す。定義は普通のクラスとほとんど同じである。例えば、図1のキーワードTypeClassNameをClassNameに、スーパークラスのVoidをObjectに変更すれば、普通のクラス定義になる。Voidはタイプクラスのルートである。

一般タイプクラスのインスタンスは、大きさがインスタンス変数全体そのものである点で、普通のクラスよりもメモリ効率が良い。また簡単な構造の場合、普通の変数のように実体を宣言で直接作ってしまうことも可能である。例えば、Point corner; とすれば、cornerはインスタンスの実体となる。自動変数的なインスタンスの生成は、newメッセージで行うよりも、この方法の方が高速である。静的に行う場合は、初期化を普通の構造体の場合と同じように行うことができる。

このように一般タイプクラスは、簡単な構造のデータをオブジェクト化するのに、メモリ･スピードの両面で効率的かつ便利である。

```
TypeClassName
    Point
Superclass
    Void
ReferenceClasses
    Instance
    Int
GlobalVariables
ClassVariables
InstanceVariables
    Int             xCoord;
    Int             yCoord;
ClassMethods
    Instance        [ newX: x y: y ]
    Int             x, y;
    {
    Point           *instance;
        instance = [ super new ];
        return [ instance x: x y: y ];
    }
InstanceMethods
    Instance        [ x: x y: y ]
    Int             x, y;
    {
        xCoord = x;
        yCoord = y;
        return self;
    }
PrivateMethods
PrivateFunctions
EndOfClass
```

図1 一般タイプクラスの定義例

Koola: An Object Oriented Language and its New Concept and Features
Kazunori TOMOTAKE, Kaoru OKAZAKI, Toshihide HAMAGUCHI, Tamotsu NISHIYAMA
Matsushita Electric Industrial Co., Ltd.

(2) 基本タイプクラス

定義例を図2に示す。基本タイプクラスのインスタンスはインスタンス変数を持たず、C言語の基本データタイプそのものか、あるいはそれらから導出されるデータタイプ(ポインタとか配列など)である。例えばChar*型の変数であるnameは、クラスCharのインスタンスとなり、文字列"Koola"と等しいかどうかを、メッセージセンディング[name is: "Koola"]で判別できる。またInt型の変数であるnの絶対値は[n absolute]で求まる。

```
TypeClassName
    Char
Superclass
    Integer
ReferenceClasses
    Memory
GlobalVariables
    #include <string.h>
ClassVariables
InstanceVariables
    typedef char    Char;
ClassMethods
InstanceMethods
    #macro          [ is: string ]
    {
        ( strcmp( self, string ) == 0 )
    }

    Char            *[ save ]
    Char            *self;
    {
    Char            *s;
        s = [ Memory new: [ self length ] + 1 ];
        return [ s copy: self ];
    }
PrivateMethods
PrivateFunctions
EndOfClass
```

図2 基本タイプクラスの定義例

(3) メソッドのマクロ定義

タイプクラスではメソッドをマクロ定義することができる。例えば先の例のメソッド[is:]は、図2に示したようにマクロで定義すれば、直接strcmpを呼ぶのと同じ速度で実行される。

以上のように、基本タイプクラスを使用すれば、基本データタイプに関して関数コールで行う処理(特にC言語のライブラリ関数のコール)を、メッセージセンディングの形に効率を全く落とさずに焼き直すことができる。従ってKoolaでは、統一的なオブジェクト指向記述が可能になった。

4. メソッドのハンドリング

高度なプログラミングを行う場合、メソッド自体をデータとして扱えることが望ましい。例えば、メッセージの引数としてメソッドを渡し、渡された方でこれを実行するというようなことができると、ダイナミックな処理が可能になり、言語の応用範囲が広がる。

Koolaではこれを、メッセージセンディングに似たセレクタ式･メソッド式を導入して、簡単に記述できるようにした。例えば、インスタンスpointへのメッセージセンディング[point x: 100 y: 200]で実行されるメソッドはセレクタ式[< point x: y: >]で得られ、それをmethodとすると、メソッド式[[point method: 100, 200]]で実行できる。セレクタ式のpointはクラス名Pointでもよい。得られるメソッドは、オブジェクトが普通のクラスのインスタンスの場合は動的に決定され、その他の場合は静的に決定される。

5. インスタンスの寿命に対応したメモリ割り付け

Koolaでは、寿命の違いに応じて、dynamic･permanent･temporaryの3種類のインスタンスが生成でき、使い分けによりプログラムの実行効率を上げることができるようになっている。permanentはプログラムの最後まで解放しないインスタンスに、temporaryは自動変数のように最後に生成したものから解放するようなインスタンスに使用する。

これらのメッセージは、種類を明示したい時にnewの代わりに使用する。例えば、[Point temporary]とすればtemporaryのインスタンスが生成できる。これらに対応するメソッドは、ルートクラスのObjectとVoidで定義されており、継承されるので各クラスで定義する必要はない。

6. 開発環境

処理系としては、プリプロセッサ以外に、現在次のようなユーティリティーが利用可能であり、クラスの開発効率を向上させている。

- koolamake .... makefileの自動生成
- koolatree ..... クラスの継承関係表示
- koolaextract ... ドキュメント自動生成

また、以下の4つのライブラリに100種類以上の汎用クラスを揃え、プログラム開発のための基盤を提供している。

- 標準基本タイプクラスライブラリ
- 標準タイプクラスライブラリ
- 標準クラスライブラリ
- グラフィックスクラスライブラリ

7. おわりに

Koolaは現在実際に使用しており、プログラムの開発･保守の効率向上、特に新人の開発力アップに非常に役立っている。今後は、ライブラリの充実をさらに行うと共に、ブラウザ･デバッガ･インタプリタなどを揃えた開発環境として整備して行きたい。

参考文献

1) 渡守武他: 多彩なメッセージ表現が可能なオブジェクト指向言語Koola 情報処理学会第38回全国大会, 3P-1.
2) Adele Goldberg and David Robson: Smalltalk-80 The Language and its Implementation.