A-006

オブジェクト指向設計のための日本語プログラミング言語の提案
A Japanese Programming Language for Object-oriented Design

馬場祐人†
YUTO BAMBA

1. はじめに

プログラムに日本語表記を用いるプログラミング手法は，40 年前から議論され続けている．また，これを実現する日本語プログラミング言語がいくつか提案されている．現在日本語プログラミングを実践するためには，一般的であるオブジェクト指向概念を採り入れることが必要である．その上で日本語プログラミング言語を評価し議論することが有意義と考える．本論文では，オブジェクト指向設計のための日本語プログラミング言語の言語仕様を提案し，その実装や構築したライブラリを用いた日本語表記のプログラムを挙げて議論する．

2. 日本語プログラミング

2.1 日本語プログラミング言語とは

日本語プログラミング言語とは，予約語や識別子に日本語表記を用いたプログラミング言語である．ソースコードのプログラム文で文字列定数のみならず積極的に日本語字句を用いる点が特徴である．また，予約語や識別子だけでなく，動詞や助詞と言った日本語文の品詞を言語仕様として採り入れ，また語順を日本語文に近い文法で形式言語の範疇で記述する点も特徴である．

神田らの議論[1]では「自然言語処理による日本語プログラミング」，「識別子に日本語表記を用いたプログラミング」とで分けて議論している．本論文で議論する日本語プログラミングは後者である．つまり自然言語処理によらず，形式言語として定義できる言語仕様で日本語プログラミング言語を実装し，その言語上で日本語表記を用いたプログラミングを実現することを目的とする．

古くから日本語プログラミングと言う言葉は存在するが，試行や効果の議論は限られる[2]．その理由には，神田らや中川らが議論した当時のコンピュータ環境では，OS や入出力環境，言語処理系，開発環境で日本語が扱うことそのものが困難であるという問題点が挙げられる．この点については現在においては Unicode が策定され OS 上で日本語が自由に扱えるようになり多くの問題点が解消した．これにより言語処理系で日本語が扱うことが過去に比べて容易となり，日本語表記でプログラミングする開発環境を整備することが可能となった．そこで現在の環境下において，日本語表記を用いた日本語プログラミング環境を整備し，日本語プログラミングを実践することで，日本語プログラミング言語の利点について議論することは有意義と考える．

2.2 日本語プログラミング言語の利点

日本語プログラミング言語の利点として考えられることは，プログラムに日本人が日々読み書きする日本語の言葉を使える点が挙げられる．日本語話者だけを対象としないソフトウェア開発や数式を多用する科学計算といった専門的な利用の目的を除けば，日本語話者を対象とするソフトウェアを日本語話者が作る場合においては，日本語で表記される事柄をプログラムコード上においても日本語表記を用いたいと考えることは自然である．

しかし一般にソフトウェアを作る場合において，プログラムコードを英数字と記号で表記している．慣習的な理由や入力の利便性などの点から，変数などの識別子も英単語を組み合わせた表記を用いることが一般的である．日本語話者を対象にしたソフトウェア開発においても，プログラム設計時には日本語で表記される事柄をプログラム上においては英単語表現で表記している．本来は，プログラム設計時に英単語表現を用いる必要は無いと考えることができる．むしろ日本語表現を英単語表現へ置き換える時に，必ずしも適切な英単語に置き換えられない場合や，開発者によるスペルミスや共同開発者間の認識の違いから誤解が生じる懸念がある．はじめからプログラムコードに日本語表現を用いれば，このような懸念を解消できる可能性がある．

このように日本語話者を対象としたソフトウェア開発では日本語プログラミング言語を用いると得られる利点があると考えられる．現在のプログラミング言語処理系では，Unicode でソースコードを扱えるようになり，文字列定数に限らず，識別子にも日本語文字が使え，部分的な日本語プログラミングは可能である．部分的となる理由は，これらの処理系が英単語でソースコードを記述することが主体であるために，予約語は英単語で定義され，標準ライブラリも英単語表記を用いて提供されているためである．識別子に日本語文字が扱えて日本語表記でプログラムが書き下せても，英単語表記と日本語表記が混在したソースコードとなることを回避できない．このような現状で，わざわざ日本語表記を部分的に用いてソースコードを書いた時に日本語表記を用いた効果を議論することは難しい．

そのためには現在のプログラミング言語に依存しない日本語プログラミングのための処理系を用意する必要がある．日本語プログラミング言語を用いれば，これまでの英単語主体の言語処理系に依存せずに日本語表記でプログラムを書くことの効果を議論することができる．

日本語プログラミングを実践するための環境として日本語プログラミング言語を設計し提案することは有意義であると考える．

2.3 関連研究

日本語プログラミング言語は，長年議論，提案されている．日本語プログラミング言語の実装としては，Forth 言語の逆ポーランド記法の語順が日本語文法の語順と一致することに着目した「Mind[3]」がある．ブロックプログラミング環境である Squeak を拡張し助詞を含む日本語の語

†所属なし

順での表記を実現した「ことだま on Squeak[4]」がある．近年では「なでしこ [5] 」の実装が公開されている．いずれも識別子を日本語の漢字かなで表記し，“てにをは”を用いて，普段読み書きする日本語の語順でプログラム文を表記するという点が共通している．日本語プログラミング言語の処理系では次の事項が共通している．

- 言語仕様の予約語に日本語表記を用いる
- 識別子に日本語表記を用いることを基本とする
- 助詞や動詞を用いたプログラム文を基本とする

本論文ではこれらの共通事項に加えて，実用規模のソフトウェア開発を想定しプログラムをオブジェクト指向により体系的に整理できる日本語プログラミング言語を提案する．

3. 日本語プログラミング言語「プロデル」

オブジェクト指向設計のための日本語プログラミング言語を提案するにあたり，提案する言語仕様を実装した日本語プログラミング言語「プロデル」を開発し，一般公開している[6]．本節ではプロデルのプログラムで用いる主要な構文要素について概説する．なお本論では言語仕様上の用語として日本語文法の用語を用いることがある．その場合，プログラミング言語としての予約語や字句の区分を示すことを目的としているため，日本語文法上の用語としての本来の意味や解釈と異なることに留意されたい．

3.1 助詞と動詞

前章で挙げた日本語プログラミング言語の大半に共通する特徴として助詞と動詞を用いて日本語の語順でプログラム文を記述する文法を基本とする点がある．この点はプロデルでも同様である．本言語における文の文法を図 1 に示す．

補語は，動詞に係る句であり本言語では引数を示す役割を持つ．実補語は，式と助詞の対となる補語である．実補語は実引数を指定するために用いる句である．補語は必ず文末尾の直後の動詞に係る．補語間の記述順には制約がない．

助詞は引数の意味を付加する字句である．言語仕様上は名前付き引数(キーワード引数とも呼ばれる)に相当する役割を持つ．名前付き引数を持つ言語には Ada や VBA と言った処理系がある．本言語でも役割はこれらと同じである．助詞を使った文法の性質は，名前付き引数に似ている．例えば Ada や C# 4.0 以降ではキーワード引数を使うことで，仮引数の名前と実引数とを対応付ける機能がある．この機能により，引数の順番を意識することなくコードを書くことができ，またコードが読みやすくなり保守にも貢献している．

本言語で用いる主要な助詞を図 2 に示す．なお本言語での助詞は実補語を構成する字句であり，日本語文法上の助詞と必ずしも一致していない．また後述する手順定義にて新たに助詞を定義して利用できる．

```
文::= { 補語 } 動詞
補語::= 実補語 | 形式補語
実補語::= 式 助詞
```

図 1　文の文法

```
を，へ(に)，で，が，の，から，まで，は，という，と，として
```

図 2 プロデルで用いる主要な助詞

3.2 動詞と形式補語

動詞は，メソッド名や制御文の予約語に相当する役割を持つ字句である．動詞は，プロデルのクラスライブラリに定義されている語の他に，後述する手順定義で新たな動詞語を定義することもできる．プロデルで用いる主要な動詞を図 3 に示す．

形式補語は，動詞を補いメソッド名の一部を成す字句である．動詞をメソッド名として用いた場合，メソッドの動作を動詞の一語で書き表せないことがある．そこで動詞に形式補語を付加し，動詞と形式補語を組み合わせてメソッド名を宣言できるようにし書き表せるようにした．例えば『四角形を描く』というメソッドを名付ける場合には，形式補語『四角形を』と動詞『描く』に別々の識別子に分けて宣言する．

```
表示する，報告する，追加する，削除する，加える，消す，選択する，描く，保存する，作る，繰り返す，返す
```

図 3　プロデルで用いる主要な動詞

3.3 手順

手順とは関数, メソッドに相当する機能である．典型的なオブジェクト指向言語のメソッド定義では，メソッド名, 戻り値型，仮引数(名前と型) をシグネチャとして宣言する．本言語の手順では助詞を含む仮引数, 形式補語, 動詞の識別子，戻り値型をそれぞれ宣言し定義する．手順呼出し文で用いる補語の定義を記述する点が特徴である．

手順宣言の説明するためにプロデルで BMI を計算するプログラムコードを図 4 に示す．このコードでは，手順を定義し(図の 3 行目～6 行目)，手順呼出し文で呼び出している(図の 1 行目)．手順を定義するには，手順宣言文を使う．『手順』という語が記された行が手順冒頭であり，『終わり』 予約語までが宣言範囲である．この定義した手順は『BMI を計算する』という名前であり，そのうちで『BMI を』が形式補語であり，『計算する』が動詞である．

手順の仮引数や局所変数を定義する場合は，変数を【　】で囲むことで定義する．変数の型は“: ”に続けて指定できる．型を省略して型推論に任せることもできる．

なお，本言語では主プログラムを手順宣言文が始まる前に書く．また 4 行目のように式計算と変数への値代入を，数式を用いて記述できる．『返す』文は手順を返戻値と共に抜け出す文である．すでに述べたように助詞(と,から)は，実引数と仮引数を対応付けるために使われる．図 4 では 54 が【体重】, 160 が【身長】変数とそれぞれ対応付けられる．

```
54 と 160 から BMI を計算して報告する

【体重:整数】と【身長:整数】から BMI を計算する手順
  【BMI】は、体重/(身長/100)^2
  BMI を返す
終わり
```

図 4 BMI(ボディマス指数)を計算するプログラム

4. プロデルにおけるオブジェクト指向設計

プログラミング言語においては文法だけでなく，それに適したライブラリと一体的に設計する必要がある．日本語プログラミングを実践し，それを評価し議論するためには日本語プログラミング言語においても同様に言語仕様のみならず，その言語処理系に適したライブラリも合わせて用意し議論する必要がある．

一方で現在のソフトウェア開発には様々な機能を組み合わせて開発する．そのためプログラミング言語は言語処理系と共に標準で膨大なライブラリが提供されており，パッケージを加えることでさらに機能を拡張して利用できる．これらのライブラリの構成は膨大であり，体系的に整理する際にオブジェクト指向の概念で整理すると構築しやすい．日本語プログラミング言語も実用的なソフトウェア開発を実現するにはオブジェクト指向の概念を採り入れることが必要であると考える．

プロデルにおいてはクラス型オブジェクト指向を採用した．本言語でプログラムを記述する際にライブラリにおいても日本語表記として違和感がないようなオブジェクト構造を本言語に適した形で検討し，実装した．この章ではプロデルにおけるオブジェクト指向プログラミングの機能について述べる．

4.1 種類定義

種類とは本言語におけるクラスに相当する機能である．種類には手順と設定項目をメンバーとして宣言する．本言語のオブジェクト指向プログラミングの言語仕様を説明するために，本言語で種類を定義するコード例を図 5 に示す．種類定義は『とは』という予約語を使う．種類宣言は『終わり』までが宣言範囲となる．

手順定義はすでに述べた通り，動詞，引数として仮引数とそれに対応する助詞を実補語として宣言する．それ加えて種類内の手順定義では，レシーバ補語も宣言する．レシーバ補語は，手順の宣言冒頭部において仮引数名の代わりに『自分』とその直後に助詞を書くことで宣言する．これは後に説明する．

なお本言語でも継承・多態性・カプセル化のための機能を用意しているが本論文では割愛する．

定義した種類からインスタンスを作り，種類内の手順を呼び出す例を図 6 に示す．『作る』文を用いると種類定義を元にインスタンスが生成される．

```
リストボックスとは
    自分に【内容】を加える手順

    終わり
    大きさという属性：サイズ
        設定する手順

        終わり
        取得する手順

        終わり
    終わり
終わり
```

図 5　種類を定義するプログラム

```
名簿リストというリストボックスを作る
名簿リストへ「山田さん」を加える
```

図 6　『作る』文によるインスタンスオブジェクトの生成

4.2 レシーバ補語

オブジェクト指向プログラミングにおけるメソッド呼出しには対象となるオブジェクト,メソッド名,シグネチャ (引数の個数と型)を順に書く．この時メソッドを呼び出す対象となるオブジェクトはメッセージレシーバと呼ばれる．典型的なオブジェクト指向言語ではレシーバオブジェクトをメソッド呼出し文の先頭に書き“.”や“->”に続いてメンバーと引数を記述する文法がよく採られている．

本言語ではメッセージレシーバとなるオブジェクトを示す補語をレシーバ補語と呼ぶ．レシーバ補語は，メッセージを受けるオブジェクトを示す特別な実補語である．手順を呼出す際に，レシーバ補語で示された助詞に対応するオブジェクトの種類で定義された手順が呼び出される．表記上はレシーバ補語と実補語との区別がなく，補語間の語順も不定である．一方で，日本語文として見た時にレシーバ補語になり得る日本文法上の語は主語や目的語であり，これは動詞によって異なる．例えば図 7 に示すように『追加する』手順ではレシーバ補語は助詞『へ』が適切であり，また『削除する』手順であればレシーバ補語は助詞『から』が適切でといった具合である．このように，レシーバ補語は特定の助詞に限らず，動詞や手順によって異なる．そこで手順定義でその動詞に応じてレシーバ補語の助詞を変えて宣言できるようにした．

そして，手順呼出し文を意味解析する際には，まず先に動詞をもとにレシーバ補語の助詞を特定し，レシーバ補語で示された式からレシーバオブジェクトを特定し，そのオブジェクトの型(種類)に適した手順を呼び出す，と言った段取りを採ることで目的の手順を呼び出す．

自分へ追加する	自分へ加える	自分を増やす
自分から削除する	自分から消す	自分を減らす
自分で開く	自分を選択する	自分を表示する
自分を保存する	自分に描く	自分に作る

図 7　レシーバ補語と動詞の対応(一例)

4.3 ライブラリ設計

これまでに述べた本言語の仕様に基づいてライブラリの構築方法を検討した．本言語では.NET Framework 上で動作することを前提に設計したが，ライブラリを実装するに当たって必然的に.NET Framework で提供されているクラスライブラリに依存することになった．本言語のライブラリを構築するに当たっては主に 3 つの方針で構築した．

1. 既存クラスを継承し英単語の識別子を日本語表記に翻訳する形でオーバライドする
2. プロデル言語に適したオブジェクト構造で種類を構築する
3. オブジェクト指向によらず広域関数を定義する

ライブラリ構築に際してはいくつかの機能を本言語向けに構築した．例えば GUI 画面を表示する機能を用意したが，ボタン部品のような GUI 部品は 1.の方法で概ね単純な翻訳で構築できた．これはボタンやテキストと言った GUI 部品は視覚的に理解できる現実に近しい構造のものであり，英

単語表記と日本語表記として概念に違いが無いためである．一方で，三角関数や算術演算関数，正規表現機能，暗号化といった機能群は，メソッドの識別子を単純に翻訳して日本語表記の手順へ定義し直すと，違和感がある場合があった．これらは 2.と 3.の方法を採った．特に引数の数が多い場合には，必ずしもすべての引数に日本語表記として意味が通る適当な助詞を当てはめることはできないケースがあった．このような場合，引数に相当するパラメータのための種類を別途定義しておくといった本言語に即した構造に再定義する方法を採った．

5. 評価

プロデルの言語仕様のもと，4.3 で述べた方針でプロデルのライブラリを設計し実装した．本言語のライブラリで用意した種類の一例を挙げて説明する．図 9 に示した種類(ファイル情報およびフォルダ情報)は，特定のファイルもしくはフォルダのパス情報を表す種類である．またファイル情報およびフォルダ情報を使ってファイルを取り扱うコード例を図 9 に示す．図 9 はフォルダ情報を操作してフォルダを作成し，ファイル情報を操作してファイルをコピーしたり削除したりしている．

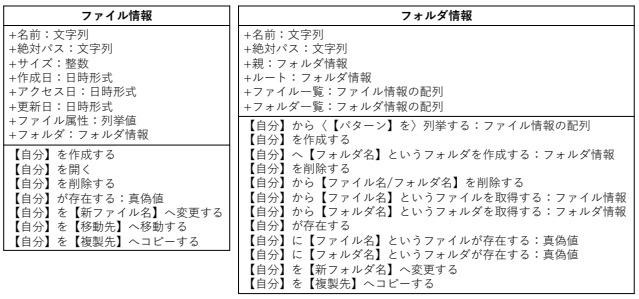

図 8 ファイル情報とフォルダ情報の手順と設定項目

```
デスクトップに「ABC」というフォルダを作成して ABC フォルダとする
デスクトップから「報告書.xlsx」というファイルを取得して報告書ファイルとする
報告書ファイルの絶対パスを報告する
報告書ファイルを ABC フォルダへコピーする
報告書ファイルを削除する
ABC フォルダをマイドキュメントへ移動する
```

図 9　ファイルとフォルダを操作するプログラム

5.1 日本語の語順で書くことの評価

プログラム文を引数と助詞とを対応付けて日本語の語順で書くことで，操作対象や入力値などの動作の対象や関係を明確にできる．またプログラムを書く際にその関数の仮引数の順番を覚える必要がなくなり，プログラムを読む際も，直感的に引数の関係が理解できる．図 9 のコード例では，形式言語の範疇で言語仕様に則ったプログラム文でかつ，日本語の文章としても意味が通る表記を実現できた．

5.2 レシーバの解釈の曖昧さ

実装においてレシーバの解釈の曖昧さが問題となるケースがあった．2 つのオブジェクトを扱う場合，手順の引数としてオブジェクトを渡す時に，レシーバ補語の助詞を主語とすべきか目的語とすべきか一意に定まらない時がある．例えば図 10 の文では，『コピーする』手順をファイル情報とフォルダ情報のどちらの種類で定義するべきか曖昧となるケースがある．設計時に『を』とすべきか『へ』とすべきかを定めないと，設計者が意図せず意味上で曖昧に定義してしまう可能性がある．本言語では，このような曖昧さが発生した時には意味エラーとして示し，定義を改めるように促すこととした．このように種類の手順を宣言する際には定義上で曖昧とならないように考慮して宣言しなければならない．先に述べたように，動詞ごとにレシーバ補語で適切な助詞を予め定め，ライブラリ全体で定義を統一することで曖昧さを解決することができる．

```
報告書ファイルを ABC フォルダへコピーする
```

図 10 レシーバ補語の助詞には"を", "へ"のどちらが適切か

5.3 プログラム文の冗長さ

コードに日本語表記を用いることでタイプ数の観点から冗長に感じられる場合がある．この点についてはプログラムエディタの開発環境の入力補完機能の工夫により改善できると考えられる．一方でプログラム文そのものがコードの説明の役割を果たすケースもあった．これまでコードへ併記した説明のための日本語のコメント文が冗長となり，本言語のコードでは不要となると言った利点が期待できる．

6. まとめ

本論文ではオブジェクト指向設計のための日本語プログラミング言語を提案した．プロデル言語は，形式言語の範疇で日本語表記を用いた言語仕様であり，オブジェクト指向を導入した．本論文では言語仕様のみならず本言語に適したオブジェクト指向に基づく日本語表記によるライブラリを構築する方針を述べ，その実装について議論した．

本言語を用いることで日本語プログラミングの効果をより実践的な観点で評価することが可能となった．一方で主流の英単語や記号を用いたプログラミングと比べて，日本語表記を用いることを評価する上で客観的に述べることは難しく，プログラムに日本語表記を用いることで直ちに日本語プログラミングに効果があるとは言えない．

今後も日本語表記や日本語文法を活かしたプログラミングのための言語仕様や技法を検討し，さらに日本語プログラミングを実践することで，議論を深めることが望ましいと考える．本提案を通じて今後も日本語プログラミングの議論が進むことを期待したい．

参考文献

[1] 神田 泰典 , 杉本 正勝 , 沢井 進, “エンドユーザのための日本語によるプログラミング”, 情報処理, p208-213, Vol.21, No.3 (1980).
[2] 中川 正樹ほか, “日本語プログラミングの実践とその効果”, 情報処理学会論文誌, Vol.35, No.10(1994).
[3] 木村 明, 片桐 明, “日本語プログラミング言語 Mind について”, 情報処理学会第 16 回プログラミング言語研究会, pp.25-32(1988).
[4] 杉浦 学ほか, “アルゴリズム構築能力育成の導入教育：実作業による概念理解に基づくアルゴリズム構築体験とその効果”, 情報処理学会論文誌, Vol.49, No.10, pp.3409-3427 (2008).
[5] クジラ 飛行机, “日本語プログラミング言語「なでしこ」に関する解説”, 情報処理 Vol.62, No.5, pp.e26-e42 (2021)
[6] ゆうと, “プロデル”, https://rdr.utopiat.net/ (2021 アクセス).