日本語プログラミング言語「言霊」におけるメソッドの記述方法

岡田　健[†]　大岩　元[††]
Ken Okada[†] and Hajime Ohiwa[††]

日本語プログラミング言語は数多く提案されてきたが、手続き呼び出しと手続き宣言を自然な日本語で記述できる言語は存在しない。本論文では自然な日本語で手続きを記述するための手法として、（1）助詞と助数詞や文脈を用いた引数渡しを日本語で表現する方法、（2）手続きを動詞、名詞、形容詞として表現する方法、（3）動詞や形容詞は相互に品詞を変換する方法、以上の３点を提案する。

1. はじめに

プログラミング教育の必要性が叫ばれている。高校教育では情報科目がスタートし、そこでプログラミング教育が行われている。プログラミング教育は当初、プログラミング能力を身に付けて実用的なソフトウェアを作成する能力を育成することを目的としていた。ソフトウェアの普及によりプロフェッショナルを除いて実用目的でプログラミングを学ぶ必要性は薄れた。だがパソコンの仕組みを理解して論理的な思考能力を育成するという教養教育の観点からは、プログラミング教育は依然として重要である。

難解で読みにくいソースコードはプログラミング教育における障壁となる。一見して意味が理解できない記号群は、初心者をプログラミングの世界から及び腰にさせる。多くの初心者は記号だらけのソースコードを見て「こんなもの私に理解できるわけがない」と早合点してしまい、プログラミングを学習する意欲が削がれてしまうのである。

プログラミング言語が難解で分かりにくい表現になっているのは、プログラマにとっての書きやすさが優先されているからである。プロフェッショナルにとっては、余計な表現が省かれた言語のほうが簡潔にプログラムを記述できる。しかし初心者にとっては表現が簡潔すぎて、何が記述されているのか分からない。

初心者教育のためのプログラミング言語として、日本語プログラミング言語が適している。プログラミング学習はソースコードの読解から始まるので、ソースコードを一読してその意味が理解できることは教育上効果的である。普段使っている日本語であればソースコードに対する苦手意識も生まれず、学習意欲が削がれることもない。

プログラミング言語を日本語化する試みは過去にいくつか行われているが、いずれも初心者にとって理解しやすい自然な日本語のプログラミング言語にはなっていない。その問題の原因は２つある。

原因の１つ目は、既存のプログラミング言語の日本語化を目標としていたことである。その例として日本語ＣＯＢＯＬ[1)]や、Ｍｉｎｄ[2)]などがある。予約語や識別子を日本語にしても、元のプログラミング言語が簡潔すぎて分かりにくい構造をしているので、初心者にとっての分かりにくさは変わらない。

原因の２つ目は、言語設計の際にプログラマにとって書きやすい簡潔な表現を取り入れてしまっていることである。日本語プログラミング言語として設計しながら、簡潔な表現を取り入れることで自ら日本語の形を崩してしまうのである。その例としてフリーウェアとして公開されているひまわり[3)4)]やＴＴＳｎｅｏ[5)]がある。プログラムを理解している人にとっての読

† 慶應義塾大学 政策・メディア研究科
Graduate School of Media and Governance, Keio University
†† 慶應義塾大学 環境情報学部
Faculty of Environmental Information, Keio University

みやすさと書きやすさを優先させるあまり、初心者にとって理解できない記法になってしまっている。これでは普通のプログラミング言語と同じ問題を抱えることになる。

言霊は初心者がソースコードを読んだときに理解を助けるようなプログラミング言語を目指す。そのためには自然な日本語の構造でプログラムを記述できるべきである。文章は名詞、動詞、助詞、形容詞などの品詞から構成され、動詞は活用して記述できるなど、普段使っている言葉☆でプログラミングできる言語が望ましい。

我々は、日本語プログラミング言語「言霊」の開発を通して、自然な日本語を使ってプログラムを記述するための手法を提案している。本論文では「言霊」で実現されている、メソッドの宣言と呼び出しを自然な日本語で記述する手法について論じる。2節では、動詞活用や文脈を利用して自然な日本語でプログラムが記述できることを示す。また、メソッドの引数や戻り値を表現する方法を述べる。3節では、様々なプログラムを日本語で記述するために、動詞だけではなく名詞や形容詞もメソッドとして記述することを提案する。4節では、メソッドの品詞変換について述べる。

2. 日本語によるメソッド記述

本節では、言霊においてメソッド宣言とメソッド呼び出しをどのように記述しているか、記述された名詞や動詞、助詞がどのように扱われるのかを述べる。なお、3節においてメソッドは様々な品詞で記述できること（名詞メソッド、形容詞メソッド）を述べるが、ここでは動詞によるメソッド（動詞メソッド）だけを使って説明する。本節で「メソッド」と記述したらそれは動詞メソッドを指す。

2.1 動詞活用

言霊のメソッドは、動詞活用をサポートしている。リスト1で「挨拶する」「退出する」というメソッドの呼び出し例を示す。メソッドの呼び出しには終止形と命令形を使って記述することが出来る。また、連用形で記述してメソッドを二つ並べて記述することが出来る。

リスト 1: メソッド呼び出しの例

```
1: 挨拶する。
2: 挨拶しろ。
3: 挨拶して、退出する。
4: 挨拶し、退出する。
```

メソッドを宣言する際には、メソッドの品詞と活用形を示す必要がある。リスト2に「挨拶する」というメソッドの定義例を示す。2行目において、このメソッドの品詞が動詞であることと、動詞の活用形がサ行変格活用であることを宣言している。動詞活用形には「五段活用」「上一段活用」「下一段活用」「サ行変格活用」「カ行変格活用」のいずれかを記述することが出来る。

リスト 2: メソッド定義の例

```
1: 「挨拶する」とは｛
2:   サ行変格活用の動詞である。
3:   「こんにちは。」を出力する。
4: ｝ことである。
```

動詞活用を用いてプログラム記述が出来ることは、自然な日本語記述の観点から多くの利点をもたらしている。例えばメソッド呼び出しが連続する場合、「～する。～する。～する。」などのような日本語になってしまうが、これを不自然な日本語と感じる人が多い。普通の日本語ならばこのような無味無乾燥な記述ではなく、意味の近い文をまとめようとする。例えばリスト3では、7行目において「１００歩進んで、９０度曲がる。」という形で文をつなげている。「四角形の一辺を描く」という目的で2つの文は非常に近い関係なので、自然とつなげて記述したくなる。文をつなげることで読みやすいソースコードを記述できる可能性がある。

リスト 3: 動詞活用のメリット

☆ 口語ではなく、丁寧な日本語でプログラムするという意味である。

```
1: ※タートルグラフィックにおいて四角形を描くプログラム
2: ※（不自然な日本語）
3: ｛１００歩進む。９０度曲がる。｝ことを４回繰り返す。
4:
5: ※上と同じ処理になるプログラム
6: ※（自然な日本語）
7: ｛１００歩進んで、９０度曲がる。｝ことを４回繰り返す。
```

動詞活用にはその他にもいくつかの利点がある。形容詞メソッドを活用することで not 演算を分かりやすく表現できることは 3.2 節において、動詞メソッド呼び出しを名詞化して表現できることは 4.1 節において、説明する。

2.2 助詞と助数詞による引数表現

言霊のメソッド呼び出しでは、引数渡しの記述には助詞と助数詞を用いる。

メソッドに引数を渡す１つの方法は、メソッドの前に実引数と助詞を記述することである。リスト 4 に、助詞による引数渡しの例を示す。この例では「に」という助詞を記述することで、その実引数がどの仮引数に対応するのかが明らかになる。

リスト 4: 助詞を用いた引数渡し

```
1: 「岡田」に挨拶する。
```

助詞による引数渡しを実現するには、引数宣言において助詞を指定する必要がある。リスト 4 の「挨拶する」メソッドの定義をリスト 5 に示す。３行目において「相手」という引数を取ることを宣言している。そして４行目において、「に」という助詞を付けることを宣言している。引数渡しは、助詞「に」を付けている値を実引数とみなすことで実現している。

リスト 5: 助詞の引数を持つメソッドの定義

```
1: 「挨拶する」とは｛
2:   サ行変格活用の動詞である。
3:   「相手」は文字列型の引数である。
4:   相手に付ける助詞を「に」とする。
5:
6:   相手、「さん、こんにちは。」を出力する。
7: ｝ことである。
```

メソッドに引数を渡すもう１つの方法は、メソッドの前に実引数と助数詞を記述することである。リスト 6 に、助数詞による引数渡しの例を示す。この例では「秒」という助詞を記述することで、その実引数がどの仮引数に対応するのかが明らかになる。

リスト 6: 助数詞を用いた引数渡し

```
1: ５秒待つ。
```

助数詞による引数渡しを実現するには、引数宣言において助数詞を指定する必要がある。記述は助詞の場合とほとんど同じである。

リスト 7: 助数詞の引数を持つメソッドの定義

```
1: 「待つ」とは｛
2:   五段活用の動詞である。
3:   「時間」は整数型の引数である。
4:   時間に付ける助数詞を「秒」とする。
5:
6:   ※ウェイト処理の記述は省略
7: ｝ことである。
```

渡す引数の記述順序は、変化しても同じ意味になる。リスト 8 のメソッド呼び出しは、１行目も２行目も同じ意味を表している。助詞と助数詞により実引数と仮引数の対応が取れるため、実引数の記述順序は重要でなくなる。また日本語を記述していても自然とこれらの順序は入れ替わるので、日本語の書きやすさの観点から記述順序は順不同であるべきだ。

リスト 8: 実引数の記述順序は自由

```
1: ９０度右に曲がる。
2: 右に９０度曲がる。
```

2.3 文　　脈

言霊において、戻り値の受け渡しには文脈を用いる。リスト 9 にその例を示す。「足す」メソッドは戻り値を返す。１行目と２行目は同じ意味の処理である。どちらも「足す」メソッドが実行されることで文脈に戻り値が一時的に保存され、「出力する」メソッドが実行される際に文脈からその値を取り出している。

リスト 9: 文脈を介して戻り値を渡す

```
1: １に２を足す。それを出力する。
2: １に２を足して、出力する。
```

文脈から値を取り出す方法は２通りあり、ひとつは「それ」引数を利用することである。１行目を実行すると、「３」の値が出力される。

文脈から値を取り出すもう１つの方法は、コンパイラの予測に任せて省略する方法である。「出力する」メソッドを実行する際には実引数が必要であるが、２行目ではどこにも実引数が記述されていない。その場合は文脈から適合する実引数を検索する。検索条件は、引数の型と文脈にある値の型が一致するか否かである。リスト 9 ２行目の例では、「出力する」メソッドは引数を文脈から求め、その結果「３」の値が見つかるので、それが出力される。

文脈を利用することにより、自然な日本語で戻り値の受け渡しを記述することが出来る。「それ」という言葉は直前の概念を表現するのに向いているし、同じく日本語プログラミング言語である「ひまわり」でも採用している方法である。

コンパイラの予測に任せた省略も、日本語的な表現方法である。日本語は不要なことはなるべく排除して、最小限度の言葉で物事を表現する傾向がある言語である。この場合もあえて助詞の「を」などを付けなくても、暗黙のうちに直前の値を使うことを期待して記述している。

しかしこれらの表現の濫用には気をつけなくてはならない。「それ」が連発する文章は読みにくいし、省略も過ぎれば意味が分からない文章になってしまう。これは我々が日常的に使用する日本語と同じである。

言霊で戻り値を返すには、メソッド宣言の中で戻り型を宣言して、処理の中で return 文を記述する必要がある。return 文は「～を返す。」というメソッドである。リスト 10 にその例を示す。７行目において戻り型を宣言して、９行目で return 文を記述している。

リスト 10: 「足す」「出力する」の定義

```
1: 「足す」とは｛
2:　 五段活用の動詞である。
3:　 「右項」は整数型の引数である。
4:　 右項に付ける助詞を「を」とする。
5:　 「左項」は整数型の引数である。
6:　 左項に付ける助詞を「に」とする。
7:　 戻り型は整数型である。
8:
9:　 右項＋左項を返す。
10: ｝ことである。
11:
12: 「出力する」とは｛
13:　 サ行変格活用の動詞である。
14:　 「対象」は整数型の引数である。
15:　 対象に付ける助詞を「を」とする。
16:
17:　 ※出力処理は省略
18: ｝ことである。
```

2.4 名前付き引数による引数渡し

助詞や助数詞を使った引数渡しを先に解説したが、これだけでは日本語らしく表現できないメソッドも存在する。そのために文脈と名前付き引数による引数渡しの仕組みを用意している。名前付き引数の概念は、Python などで実現されているものと同じと考えてよい。リスト 11 にその例を示す。

リスト 11: 名前付き引数による引数渡し

```
1: 半径を１０として、円の面積を計算する。
```

名前付き引数とは名前と値のセットを文脈に保存し、メソッド呼び出し時に仮引数名を鍵に検索することで実現する。リスト 11 で「半径を１０として」（する（＝して）はメソッドである）と記述すると、「半径」という名前と「１０」という値のセットが文脈に保存される。そ

の後「円の面積を計算する」というメソッドが呼び出されると、スタックから実引数を検索しようとする。リスト12に示される定義によれば、仮引数名は「半径」である。そこで文脈上にそういう名前と値のセットを検索する。その結果「半径」と「１０」のセットが見つかり、仮引数「半径」に１０の値が代入されてメソッドが実行される。

リスト 12: 「円の面積を計算する」メソッドの定義

```
1: 円の面積を計算するとは｛
2:    サ行変格活用の動詞である。
3:    「半径」は整数型の引数である。
4:
5:    半径×半径× 3.14 を出力する。
6: ｝ことである。
```

「円の面積を計算する」のように、助詞や助数詞によって表現できない引数は多い。その場合は名前付き引数を用いることで、日本語の形を損なわずにメソッド呼び出しを表現することが出来る。

3. 様々な品詞によるメソッド記述

本節では、動詞以外に名詞や形容詞もメソッドとなりうることを述べる。

メソッド呼び出しを自然な日本語で記述しようとすると、メソッドが様々な目的で記述されていることが分かる。実験的に様々な Java プログラムを日本語で記述してみたところ、メソッドの目的により記述に使う品詞が変化することが分かった。

様々なメソッドを検討した結果、動詞と名詞と形容詞の３種類が主にメソッドとして記述できることが明らかになった。これはメソッドが呼び出される目的によって異なるためだと考えられる。例えば動詞で表現されるメソッドは、主に何らかの処理を行うことが目的であり、名詞で表現されるメソッドは何らかの値を戻り値として得ることが目的のメソッドなのである。形容詞で表現されるメソッドは、真偽判定が目的のものが多かった。

以上の考察により言霊では、動詞メソッド、名詞メソッド、形容詞メソッドを記述する仕組みを用意した。この３種類のメソッドで、大部分のプログラムが日本語で表現できそうである。

動詞メソッドは既に 2 節において説明したので、本節では名詞メソッドと形容詞メソッドについて解説する。

3.1 名詞メソッド

名詞メソッドは主に何らかの値を返すメソッドを表現するときに使われる。

3.1.1 引 数 渡 し

名詞文は全て「〜の〜」として表現されることが特徴である。名詞メソッドを用いたプログラム例をリスト 13 に示す。「半分」が名詞メソッドであり、「１０」がその実引数である。「１０の半分」と記述することで「半分」メソッドが呼び出される。戻り値として「5」が返される。

リスト 13: 名詞メソッドの呼び出し

```
1: １０の半分を出力する。
```

名詞メソッドを宣言する際には、自身が名詞メソッドであることと、戻り型を示す必要がある。名詞メソッドの定義をリスト 14 に示す。２行目において自身が名詞であることを示し、また整数型を戻り型として返すことを宣言している。名詞メソッドは必ず戻り値を返すので、戻り型は宣言しなければならない。

リスト 14: 名詞メソッドの定義

```
1: 「半分」とは｛
2:    整数型の名詞である。
3:    「数値」は整数型の引数である。
4:
5:    数値÷２を返す。
6: ｝ことである。
```

3.1.2 助詞を用いた引数渡し

名詞メソッドは、助詞を使って引数を指定することも出来る。メソッド呼び出しの例をリスト 15 に示す。「１０と」「１５」が実引数であり、「最大公約数」が名詞メソッドである。実行されると戻り値として「5」が返される。

リスト 15: 助詞を用いた名詞メソッドの呼び出し

```
１０と１５の最大公約数を出力する。
```

助詞を用いた引数渡しを名詞メソッドで実現するには、動詞メソッドと同じように引数宣言で助詞を指定する。そのプログラム例をリスト 16 に示す。

リスト 16: 名詞メソッド「最大公約数」の定義

```
「最大公約数」とは｛
  整数型の名詞である。
  Ａは整数型の引数である。Ａの助詞を「と」とする。
  Ｂは整数型の引数である。Ｂの助詞を「を」とする。
  Ｒは整数型の変数である。

  Ｒに、ＡをＢで割った余りを書き込む。
  Ｒ＞０である限り｛
    ＡにＢを書き込み、ＢにＲを書き込む。
    ＲにＡをＢで割った余りを書き込む。
  ｝ことを繰り返す。

  Ｂを返す。
｝ことである。
```

3.2 形容詞メソッド

形容詞メソッドは、返り値として真偽値を返すメソッドを表現するときに使われる。基本的な構造は動詞メソッドと変わらない。

形容詞メソッドは、仮定文や繰り返し文で使用される。そのプログラム例をリスト 17 に示す。１行目で「Ａが０より大きい」という形容詞メソッドが呼び出されている。「Ａが」「０より」が実引数であり、「大きい」が形容詞メソッドである。「大きい」メソッドが呼び出されると演算が行われ、その結果が真偽値で返る。

形容詞メソッドは、活用形を利用することで

リスト 17: 形容詞メソッドの呼び出し

```
もしＡが０より大きいならば｛
  ※省略
｝ことをする。

もしＢが０より大きくないならば｛
  ※省略
｝ことをする。
```

リスト 18: 形容詞メソッドの定義

```
「大きい」とは｛
  形容詞である。
  「左項」は整数型の引数である。
  「右項」は整数型の引数である。
  左項に付ける助詞を「が」とする。
  右項に付ける助詞を「より」とする。

  左項＞右項を返す。
｝ことである。
```

not 演算を自然に表現することが出来る。リスト 17 の５行目では、「大きくない」という表現で形容詞メソッドが呼び出されている。これは「大きい」メソッドの否定形であり、呼び出された結果を not 演算した値が戻り値となる。

形容詞メソッドを宣言するには、自身が形容詞であることを示す必要がある。「大きい」メソッドの定義をリスト 18 に示す。形容詞メソッドを宣言する際には、自身が形容詞であることを示す必要がある。その他の引数の宣言などは動詞メソッドと変わらない。また、形容詞メソッドは戻り値が真偽値であることが確定しているので、戻り型の宣言は行われない。

4. メソッドの品詞変換

3 節では様々な品詞によるメソッド記述を紹介した。動詞メソッド・名詞メソッド・形容詞メソッドは、それぞれ処理・値・条件を日本語で表現するためのメソッドであることを示した。

しかしこの３種類のメソッドだけでは読みにくい日本語になってしまう事例があり、言霊では解決のためにメソッド呼び出し文を品詞変換する仕組みを取り入れている。本節では具体的に不自然な日本語になる例を示した上で、品詞変換を導入することで自然な日本語表現が可

能になることを示す。

4.1 動詞の体言化

動詞メソッドを呼び出す目的が処理と値の両方である場合、これまでの仕組みでは自然な日本語で表現できない。リスト 19 にその例を示す。「削除する」というメソッドは、配列からデータを削除して、削除したデータを戻り値として返す。データを削除することだけが目的ならば、「削除する。」と動詞の形で記述すればいいのだが、削除した上で戻り値のデータを利用したい場合は、どうにも上手く表現できない。１行目は無理やりこれまでの枠組みで表現した例であるが、不自然な日本語となってしまっている。

リスト 19: 動詞の体言化が必要な例

```
1: 削除して、出力する。
2: 削除したものを出力する。
3: 削除した値を出力する。
```

動詞メソッドの呼び出し表現を体言化することで、読みやすい日本語表現が実現できることがある。それがリスト 19 の２行目「削除したもの」である。動詞を連体形で表現して「もの」という語をつなげることで、体言化している。

動詞の体言化では、動詞がかかる名詞の表現が問題になることがある。例えばリスト 19 では「もの」という言葉で戻り値を表現した。だが状況によっては「値」「整数」「文字列」など、異なる名詞が適することもある。言霊ではこの戻り値の表現を、手続き宣言の中で定義することが出来る。定義例をリスト 20 に示す。４行目にて「値」を戻り値表現としている。この場合は、リスト 19 の３行目のように記述することが出来る。

リスト 20: 戻り値表現

```
1: 「削除する」とは｛
2:   サ行変格活用の動詞である。
3:   戻り型は整数型である。
4:   戻り値表現は「値」である。
5:
6:   ※処理は省略
7: ｝ことである。
```

4.2 形容詞の体言化

形容詞文を呼び出す目的は、主に条件を記述することである。例えばリスト 17 の１行目と５行目のように、「もし～ならば」「～限り」などと接続して仮定文や繰り返し文を記述する場合である。この場合は形容詞文の表現で記述しても適切な日本語文になる。

だが形容詞文を値として記述する場合、日本語としておかしくなってしまう。例えば形容詞文の判定結果を出力しようとすると、「ＡよりＢが大きいを出力する。」などのように、日本語として不自然な表現になってしまう。

形容詞文を値として利用する場合は、形容詞文の末尾に「かどうか」をつけることで名詞化する。その例をリスト 21 に示す。「数値Ａが数値Ｂより大きいかどうか」と記述することで、形容詞文を名詞化している。

リスト 21: 形容詞を名詞化している例

```
1: ＡがＢより大きいかどうかを出力する。
```

5. おわりに

本論文では、自然な日本語でメソッドを記述するための手法を提案した。メソッドへの引数渡しは、値に助詞・助数詞を付けることや、文脈を用いることで日本語らしさを損なわずに表現する。現在使われているメソッドの使途は多種多様であり、動詞メソッド・名詞メソッド・形容詞メソッドといった表現を用いると日本語らしいメソッド表現が可能であることを示した。

本論文では紹介しなかったが、メソッド宣言や変数宣言、仮定文や繰り返し文などは全てこの記法に従って定義され、呼び出されている。言霊のコンパイラを言霊を使って記述する試みを行っており、それによって言霊がアルゴリズムの表現に適していることを実証するつもりである。

今後の課題として、この構文の上でどのような表現をすれば初心者にとって読みやすいソースコードになるのかを検討することである。現在実験的に大学の授業において言霊を使ってプログラミングを教えているが、代入文や宣言文の表現をどのようにすれば初心者にとって分かりやすい教育が行えるかという議論が行われている。

様々な問題点はあるが今後も検討を重ねて、プログラミング教育における活用の可能性を考えていきたい。

謝辞

有益な討論にお付き合い頂いた慶應義塾大学の松澤芳昭氏、中鉢欣秀氏に感謝致します。

参　考　文　献

1) 床分眞一、今城哲二. COBOL における日本語機能の現状と今後の動向. プログラミング言語 16-9, pp. 53–56, 1988.
2) 木村明、片桐明. 日本語プログラミング言語『Ｍｉｎｄ』について. プログラミング言語 16-4, pp. 25–32, 1988.
3) http://hima.chu.jp/
4) 山本峰章. 日本語でかんたんプログラミング！「ひまわり」で学ぶアプリケーション作成. 毎日コミュニケーションズ, 2004.
5) http://hp.vector.co.jp/authors/VA021321/