オブジェクト指向プログラミング（OOP）の学び方を考える

綾　皓二郎*1
Email: aya.k2015h27@gmail.com

*1: みやぎインターカレッジコープ

◎Key Words　　手続き型と OOP，計算論的思考(CT)，OOP と英語

1. はじめに

プログラミングを学ぶ場合，オブジェクト指向プログラミング（OOP）は，手続き型プログラミングほど容易ではなく，学習のハードルがかなり高いことがよく知られている。(1) 本報告では，なぜハードルが高いかを調べ，OOP の難しさを軽減する学び方を検討した。(2)

現在の OOP 言語は多様で，OOP 諸語といってもよいくらいの文法・専門用語にかなりの違いがあるので，学習者はどの OOP 言語を選んで学ぶとよいかという問題が最初にある。

2. OOP 言語の選び方と学習の順序

現在の OOP 言語の主流は，OOP 言語を含むマルチパラダイム言語である。初学者がいきなり純粋な OOP 言語を選ぶことは，学習の準備とその後の展開，および学習環境がマルチパラダイム言語よりはるかに劣るので勧められない。OOP 言語を学ぶ前に，手続き型言語で変数やデータ構造，関数などのプログラミングに共通する概念の理解，抽象化や分解・分割，パターン認識，再帰などの計算論的思考（CT）のコア概念を理解しておくほうがよい。これにより，OOP の理解がスムーズに進むので，マルチパラダイム言語の OOP 言語を学ぶほうが優れた選択である。

OOP 言語を含むマルチパラダイム言語には, Java, JavaScript, C++，Ruby，Python などがあるが，多様な OOP 諸語の間では同じ概念が多くは異なる用語で表されている。たとえば，

- データ：インスタンス変数，メンバー変数，データメンバー，フィールド，属性，プロパティ，状態，など
- 処理： メソッド，振る舞い，操作，メンバー関数，機能，など
- 継承： 基底クラス，スーパークラス；親クラス，派生クラス；サブクラス，子クラス

学習者の頭を混乱させないためには，上記に見られる用語の洪水や，一つの言語を学ぶときに，他言語の用語をむやみに取り込んで説明することをできるだけ避けることが望ましい。たとえば，C++流の OOP での「オブジェクトに対するメソッドの呼び出し」を Smalltalk 流の「レシーバに対するメッセージ送信」などと比喩的に説明することは避けたほうがよい，と思われる。

本報告では，クラスベースの OOP 言語である Python を選んでいる。

3. 手続き型言語から OOP 言語への拡張の視点から学ぶ

手続き型プログラミングから OOP へのパラダイムの自然な拡張をみると，現在のクラスベースの OOP 言語の方法論は，まったく新しいものではなく，手続き型言語の延長線上にあると考えると，理解しやすくなる。

プログラミング言語の開発での方向軸は第一に抽象化のレベルを上げていくことにある。手続き型言語における抽象化には「データをまとめる抽象化」と「処理をまとめる抽象化」が別々にあるが，OOP 言語では，データと処理の抽象化をクラスとして一つにまとめて，抽象化のレベルを一段上の段階に引き上げ，これにより可読性や再利用性などプログラムの作成効率を高めることに成功している。このクラスの抽象化では，データ型を自由に作ることができ，さらにデータ処理のためのメソッドをもつ。クラスはオブジェクトのモデル（テンプレート，設計コード）である。オブジェクトは，手続き型の関数呼び出しと同様に，クラスを呼び出すことで生成される。手続き型プログラミングにおける関数というブラックボックスを拡張したものが，OOP 言語におけるクラスというカプセルである。

Pyhton では初めに手続き型プログラミングを学ぶ過程で，OOP を導入する準備，OOP への慣れが組み込まれている点で OOP が学びやすいと思われる。すなわち，手続き型プログラミングでの組み込みのデータ型が，OOP では組み込みのクラスに拡張され，型とクラスは区別されず同義となっている。オブジェクト，メソッドなど本来は OOP で学ぶ事項が，概念・定義はとりあえず置いておいて，使い方を優先してして出会うことができる。たとえば，組み込み関数 `list()` や `str()` は，変数のデータ型を変換（キャスト）する関数というよりは，`list` クラス，`str` クラスのオブジェクトを生成するコンストラクタと考える。しかもこのオブジェクトは型をもつだけでなく，オブジェクトに固有の関数といえるメソッドもつことができるように拡張されていることを知る。このように，Python では手続き型のプログラミングで，オブジェクトやメソッドをクラスの定義を正式に学ぶ前に知って使うことができるように慣らされる。

4. OOP の基本的な用語の原義を理解する

OOP のクラス，オブジェクト，メソッドという基本用語を，学校英語でクラスを学級，オブジェクトを物体や目的語，メソッドを方法という訳語で類推しても，何か掴みどころがないと感じるのではないだろうか。こういうときには，英英辞典で単語の原義を理解すれば，これらの単語がなぜ術語として選ばれているかがよくわかる。COD によれば，

class: a set or category of things having some property or attiribute in common and differentiated from others by kind, type , or quality（クラス：ある特性や属性を共通に持つものの集合またはカテゴリーであり，種類，タイプ，または品質によって他と区別される）。この原義からクラスは抽象化されたものであることが理解できる。

object: a material thing that can be seen and touched. Philosophy a thing external to the thinking mind or subject. a person or thing to which an action or feeling is directed（オブジェクト : 見て触れることができる有形の物。哲学では，思考する心，あるいは主体の外側にある物）。この哲学での定義に留意しておきたい。

method: a particular procedure for accomplishing or approaching something（メソッド：何かを達成する，またはアプローチするための特定の手順）。メソッドはサブルーチンや関数と同義の言葉である。

5. OOP における俯瞰的視点の重要性を知る

日本語話者が OOP の学習のハードルを下げるには，英語 OOP 言語の背景に，日本語とは根本的に異なる英語の視点・認識があることを知っておくことが重要である。すなわち，英語はモノ志向，場面外視点，客観的観点，分析的把握に，日本語はコト志向，場面内視点，主観的観点，体験的把握に特徴がある,と言われている。[3][4] すなわち，英語では，モノを場面外から客観的に観て分析的に認識する。たとえば，川端康成の小説『雪国』の冒頭の文が，よく引用される。「国境の長いトンネルを抜けると雪国であった」は，サイデンステッカーの英訳では「The train came out of the long tunnel into the snow country」となっている。英文では，原文にはない汽車というモノが主語として出てきて，トンネルから抜け出る汽車を，視点を汽車の外に置いて，それも上方から見下ろして認識していることが分かる。[5]

英語特有の視点は，次の文章に明確に捉えられている。“ものごとを抽象化・理念化 =「物」化してとらえる思考（&志向）は英語（等欧米語）の本性である。「物」と「物」の（さまざまな「組み合わせ」としてものごとをとらえ，あるいは構成していく思考は，機械，機械文明に最も端的に現れている。”[6] 英語 OOP 言語がこの英語の発想を受け継いでいることを日本語話者はまず知っておかねばならない。

日本語話者がオブジェクト指向をとらないわけではない。たとえば，将棋の棋士や舞台の演出家，野球の監督は，オブジェクト指向で場面全体を場面外から俯瞰的に観て分析して, 多数のオブジェクト（駒, 俳優，選手など）をオブジェクトが持つ固有のメソッドを使って操作して，それらを協調的に振舞わせることで，その職分を果たしている。

OOP を用いるアニメやゲーム，GUI アプリのプログラミングでは，プログラム作成者は画面上に生成したオブジェク（構成要素）の状態を外から俯瞰的に見て，それぞれに固有なメソッドを使ってオブジェクトに指示を出して課題を解決していく方略をとらなければならないことがわかる。

6. 計算論的思考に基づいて OOP を学ぶ

計算論的思考（computational thinking，CT）とは何かについてはさまざまな考え方ある[7]-[12] が，ここでは「計算が関わる，複雑な課題の解決やシステムの設計に, 数理科学やコンピューター科学, 計算科学, 理工学などにおける思考の諸概念を用いる知の方法

論および思考のプロセス」と定義しておく。[2] CT のコアとなる概念は，抽象化とモデル化，分解と分割，パターン認識，再利用などであるが，これらがアルゴリズムの考案や OOP プログラミングで実際にどのように使われているかを理解する必要がある。クラスやオブジェクトの概念の理解には計算論的思考が必須である。なお，Python などの近年の OOP 言語は，CT の概念に基づいてプログラミングするように設計されているので，CT はプログラミングの過程で自ずと育まれるのであるが，学習の初期には，CT を常に意識してプログラミングすることに越したことはない。CT は，アルゴリズムの構築のみならず，プログラムの作成にも適用できる方法論であることに注意する。

7. OOP でプログラムが効率的に作成できることを学ぶ

OOP 言語の登場の背景には，爆発的に拡大する情報処理に対処できる，信頼性の高い，大型化するプログラムを手続き型言語の手法で効率的に作成することはとても難しくなってきたことがある。特にユーザーとの対話的処理やイベント駆動型の処理を必要とする GUI アプリケーションや Web アプリの作成を容易にさせる生産性の高いプログラミング手法が求められてきたことが OOP 言語の隆盛の要因となっている。

OOP では，クラスを作っておけば，これを再利用して同じ型のデータの異なるオブジェクトをいくつでも作り，オブジェクトをプログラムの構成要素（部品）として利用することができる。さらに，OOP ではあるクラスを継承する派生クラスを作ることにより，プログラム作成の再利用性と効率化を高めることもできる。また，クラスのカプセル化により，クラスへの好ましくないアクセスを未然に防ぎ，プログラムの安全性を高めることもできる。

このように，OOP の手法は，複数のクラスから生成された多数のオブジェクトを部品として利用し，これらの多数の部品を組み合わせ，協調動作させることにより，信頼できる大きなプログラムを効率的に作成することを可能にさせるものである。

OOP では新しいプログラムを一から作ることで，時間と労力の無駄となる，いわゆる「車輪の再発明」をできるだけ避けて，プログラムの大半を容易に手に入る部品の組み合わせで作成する。このとき，クラスから作成された部品の内容の詳細は知らなくてよい。これはロボットを製作するときに，回路部品の中身の詳細を知る必要がないことと事情はまったく同じである。

OOP では実行効率の高いアルゴリズムの考案よりも，再利用性や信頼性，保守性を確保して効率的にプログラムを作成する設計のほうがより重視される。さらに GUI／Web アプリの作成では，アプリのアイデアやデザイン，使い勝手の評価の優先順位が高い。そのためには，高品質のアプリを効率的に作成できるライブラリやフレームワークを整備することが必須となっている。

8. OOP プログラミングの易しい例を学ぶ

OOP の基本的な文法を学んだら，OOP が威力を発揮する，アニメやゲーム，GUI アプリなどの平易な作成例について学ぶことが欠かせない。文法を苦労して学んでも，ここまでやっておかないと，OOP の有効性が実感できないからである。

8.1 タートルグラフィックスにおける OOP

Python の turtle モジュールでは，OOP でタートルグラフィックスを行うことができる。これは，子どもたちがオブジェクトを外から操作する簡単な OOP に触れることで，コンピューターとの能動的な付き合い方を学ぶことや，日常的に接している Web アプリの働きを推し量ることができるようになることを狙っているのではないかと思われる。

タートルグラフィックスでは，たとえば，二匹の亀さんオブジェクトが円柱を周る出会いの場面を簡単にプログラムすることができる（図１）。また，数匹の亀さんオブジェクトを競争させるアニメーションプログラムでもよい。

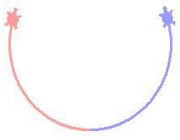

図１　二匹の亀さんの出会い

8.2 GUI アプリの作成における OOP

GUI アプリの作成は分析と設計，実行，評価までを含んだ OOP である。アプリの作成では，まずどのようなインターフェースとオブジェクトを必要とするかの GUI の分析と設計に取り組む。複数のクラスから多数のオブジェクトを生成し，画面にどのよう

に配置し，オブジェクトをどのように連携・協調動作させるかを俯瞰的に考える必要がある。

具体的な例として，BMI 計算器の GUI アプリ（図 2）の作成を取り上げる。必要なオブジェクトには，GUI のウインドウ，ラベル，入出力ボックス，ボタンがあり，各オブジェクト（部品）が何個必要で，どこにどのように配置するかを，俯瞰的に見る外観図を作って決めなければならない。さらに，データ入力とボタンなどのクリックによるイベントの発生を受けて BMI の演算処理をどのように始めるかのイベント処理，さらに演算結果の出力までを設計しなければならない。

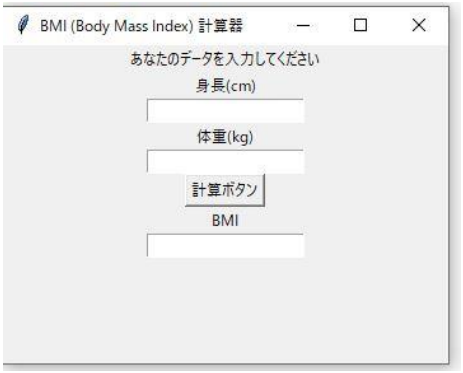

図 2　BMI 計算器（GUI アプリ）の外観図

GUI アプリの作成の場合，MVC アーキテクチャー（デザインパターン）を用いることで効率的にプログラムを作成することができる。MVC アーキテクチャーは一種の分割統合法であって，システムを M(Model)，V(View)，C(Controller) という 3 つの基本単位（サブシステム）に機能分割して作成する。

OOP の下で GUI アプリ作成するプログラムでは，標準ライブラリ Tkinter の tk モジュールが用いられる。View では，tkinter モジュールに収められている，ラベルやボタンなどのクラスを呼び出してオブジェクトを生成し，配置する。Controller は，bind()メソッドを呼び出すことで，イベントと Model にあるイベントハンドラー（演算処理関数）を結合する。この結合により，イベントがハンドラーに渡され，Model では入力されたデータを用いて演算を開始する。このように，OOP は GUI におけるイベント駆動型の処理に適している。

9. おわりに

本報告は，拙著[2] の原稿執筆での OOP の学習についての考察をまとめたものである。OOP の入門学習は，本報告で述べたことを頭に入れてやれば決して難しいものはないことが分かる。

生成系 AI（Artificial Intelligence，人工知能）技術が驚異的な進歩を見せている。たとえば，ChatGPT のサービスに，課題のアルゴリズムとプログラムを教えてくださいとリクエストすると，直ちに回答が得られて，課題解決はきわめて容易である。このようなサービスに依存していて，課題解決能力の育成ができるだろうか。プログラミングを学ぶ楽しさが期待できるだろうか，AI の時代に学校教育でプログラミングを学ぶ意義と学習法を真剣に考察しなければならないときが到来している。

参考文献

(1) 平澤 章：“オブジェクト指向でなぜつくるのか　第 3 版”，日経 BP(2021).

(2) 綾 皓二郎：“計算論的思考を育む python プログラミング入門”，近代科学社(2023).

(3) 池上嘉彦：“「する」と「なる」の言語学―言語と文化のタイポロジーへの試論”，p.256，大修館書店 (1981).

(4) 濱田英人：“認知と言語: 日本語の世界・英語の世界”，開拓社(2016).

(5) 金谷武洋：“英語にも主語はなかった”，pp.28-29，講談社(2004).

(6) 岩谷 宏：“にっぽん再鎖国論―ぼくらに英語はわからない”，p.265，ロッキング・オン社(1982).

(7) Wing J. M.: “Computational Thinking”, *Commun. ACM*, Vol.49, No. 3,　pp.33–35(2008).

(8) 中島秀之(訳)：“計算論的思考”，情報処理, Vol.56, No.6, pp.584-587 (2015).

(9) 磯辺秀司，小泉英介，静谷啓樹，早川美徳：“コンピテーショナルシンキング”，共立出版 (2016) .

(10) Dennin P. J., Tedre M. : “Copntational Thinking”, MIT press (2019).

(11) 綾 皓二郎：“計算論的思考（CT）」を育むプログラミング教育を考える”，2021 PC Conference論文集，pp.120-123 (2021) .

(12) 中島秀之，平田圭二他：“計算論的思考ってなに? ―コンピュータサイエンティストのように考える”，近代科学社 (2022) .