オブジェクト指向プログラミング教育法序説

－ The Primary Study of Educational Method for Object-Oriented Programming －

seastar@orion.nifty.jp

目　　次

序
1. 新学習指導要領でのオブジェクト指向プログラミングの取扱
2. 現在までの開発教材の紹介
3. オブジェクト指向の要点
4. 補足点
5. ＯＯＰの汎用化（オブジェクト指向学のすすめ）
結　び

序

携帯電話や表計算ソフトの隆盛により、プログラミング教育が衰退している傾向にある。論理的思考法の訓練にとどめればよいのか、それとも、実務教育への道を開く先端技術を伝え発達方向を知らせるのか、逡巡することが多い。

しかし、デカルトの言うとおり、踏み出さなければ出口はどこであれ森からは出られない。今回は**オブジェクト指向プログラミング**について、実践的に研究し、教材化に取り組んだ。その過程で図式化することが、複雑な事象をとらえ分析するのに大変有効であることに気づいた。この手法の様々な一般化を試み、オブジェクト指向学ともいうべきものの見方を論じてみる。

なお以後は Object-Oriented Programming の頭文字を採り、オブジェクト指向プログラミングを**ＯＯＰ**（おーぷ）と略記することにする。

1.　新学習指導要領でのＯＯＰの取扱

Ｃ＋＋やＪａｖａなどのＯＯＰ言語が普及し、ソフト開発の効率化と堅牢性が追求されるようになり、ＯＯＰ教育は無視できなくなった。新しい高等学校学習指導要領においては､専門教科 情報の「情報システム実習」科目でオブジェクト指向設計について取り上げるように指示されている。

同じく商業科の科目「プログラミング」では、「データ構造と制御構造」の単元で、間接的にＯＯＰのクラスの取り扱いについて指示しているように解釈できる。また、本年度の学会総会の資料によれば、商業科では「プログラミング」科目でのオブジェクト指向型言語や手続き型言語の選択幅を拡大するように改善する方針で、新課程導入までにどのように教材化するかが急務であると判断した。（第２０回全国大会要項１３ページ「新学習指導要領　商業科の概要」より）

2.　現在までの開発教材の紹介

では、早速ＯＯＰ実習を紹介しよう。

（１）教材のダウンロード先の紹介

まず、実習用のファイル及び手引きを用意したダウンロード先が次のＵＲＬである。

http://homepage1.nifty.com/tetsuhito/OOP-Labo/oop-menu.html

マイクロソフト社製ウインドウズで動作するファイルであるが、ぜひともこのサイトから教材集ファイルをダウンロードして、実習してみながらこの研究成果を読んでいただきたい。

（２）ＷＳＨを活用したプログラミング実習

次に実習のために選択したプログラミング環境を説明する。具体的に次のコードをエディタで打ち込み､p1.vbs のファイル名でマイドキュメントフォルダ等に保存してみていただきたい。

```
msgbox "1+1=" & 1+1
```

その保存先フォルダ内の"p1.vbs"ファイルの水色の巻物のようなアイコンを確認の上、ダブルクリックする。すると､「 1+1=２ 」と表示されるはずである。これがウインドウズに元々、内蔵されているＷＳＨ機能であり、原稿末の書籍等で学べばより高度な処理が可能である。また前掲のサイ

トに実習教材を並べているので、多いに活用していただきたい。

私の勤務校では、１年「情報処理」の３学期の授業で、３時間実習するようにして、効率よくデバッグやトレースやアルゴリズムを習得させている。なお、セキュリティ設定でＷＳＨが働かない場合は、設定を変更する必要がある。

（３）扇風機オブジェクトのモデル

さて、この vbs ファイルでオブジェクトを作成し、名前を付けて操る。長年の勉強の末に考え出したのが扇風機オブジェクトである。実際のプログラムは、ダウンロードし実行やソースコードの表示をしていただければよいが、まずはＯＯＰの概念を明確にするために、扇風機という機械をＯＯＰの考え方でとらえてみよう。

表１　オブジェクトの操作例

扇風機の操作例	プログラムにおけるオブジェクト操作の記述例
扇風機とは、・・・である。	宣言 扇風機｛変数 風量, 首振オンオフ; 命令 風量変更(・・・);首振切替(・・・);｝
涼風という扇風機を制作する。 涼風の風量を 3 段階に上げる。 涼風の首を振るように切り替える。 涼風を廃棄する。	涼風 = new 扇風機 涼風.風量変更 3 涼風.首振切替 〝ON〟 涼風.廃棄

扇風機とは何か。それは、扇風機の機能を持つ機械の総称である。この抽象的な扇風機を、**抽象的オブジェクト**とすれば、様々なメーカーで作り出された様々な種類の扇風機は、**オブジェクト実体**に当たる。このオブジェクト実体を**インスタンス**という。そして機種が異なろうとも、それぞれの扇風機のチャンネル換え機能やボリューム調節機能などの基本的機能は共通している。この共通する機能を**メソッド**という。

このことをプログラムのオブジェクト操作風に表現すると、表１のように、扇風機という抽象的なものの風量を変えるのではなく、具体的な涼風と名付けた対象物の風量を変えたり、首振りを切り替えたりするメソッドを指示する。プログラムの中でも一般的な複数の操作をまとめて、抽象的オブジェクトを決めておき、その規格にあったインスタンス（オブジェクト実体）を作り出したあとで、そのインスタンスを操作するのである。だから、更に他の対象の操作を付け加えたい場合には、別の扇風機オブジェクト 「パワーファン」などと名付けて実体化宣言し、この「パワーファン」に対していろいろと操作する手順をとることになる。

いきなり専門用語が並び、読み直した先生方もいらっしゃることだろうが、現在のプログラミングは、このような緻密な記述を積み上げてできるだけ使い回しの利くオブジェクトを活用するようになってきた。例えば、大流行しているネットワーク型対戦ゲームの開発では、選手オブジェクトを決め、参加人数分のインスタンス（オブジェクト実体）をゲーム世界に生み出し、それぞれの機能をネットを介して指図しながら、ゲームを進めていく。これが、古典的なプログラム言語であれば、参加人数分の別々の処理を分けて作り、同じ動作でも各処理ごとに方言がある命令を使い分けて操作していくことになるだろう。

以上のような考え方に基づき、実際に開発したのが独自の扇風機モデルである。例えば、「扇風機とは ～ である」にあたるコードが、

```
Class Senpuuki ～ End Class
```

である。「涼風という扇風機を作る」コードは、

```
Suzukaze = new Senpuuki
```

である。「涼風の風量を２にする」コードが、

```
Suzukaze.change_tsuyosa 2
```

である。

この扇風機オブジェクトの実例のように、ＯＯＰにおいては、機能を分担したり、それぞれの動作対象を特定したりしながら処理を進めていく。

ＯＯＰにおいては、**オブジェクトを使うノウハウ**と**オブジェクトを作るノウハウ**がある。当然、作る方が難しいので、まずオブジェクトを使わせようと生徒に扇風機オブジェクトを与え、任意の名前の扇風機を設定し操作させてみたのである。

３年生の「ビジネス情報」科目の２学期で実習したところ、生徒たちはこの仮想の扇風機をおおむね理解し、自分なりにメソッドを並べて操作していた。そこで高校生にＯＯＰを指導することが無理ではないとの手応えを実感したのだった。

（４）ウインドウオブジェクトのモデル

モデル化できるようになると様々な対象をオブジェクト指向的にとらえてみたくなる。そのような自作のオブジェクトの一つがウインドウオブジェクトである。これもファイルをダウンロードできるようにしているので、操作しながら分析してみていただきたい。

あるとき同僚教師にＯＯＰを説明していて、パソコン画面にいくつも重ねて出しては操作しているウインドウもオブジェクトの例なのだと説明したときに自分でもひらめくものがあった。つまり、生徒たちにウインドウを操作させるＯＯＰプログラミングを体験させることができたら、それは扇風機のようなバーチャルなオブジェクトではなく、じかにオブジェクトを扱わせることができるのである。ウインドウオブジェクトを定義し、画面上に沢山並べてみようと取り組んでみた。ＯＳが用意している特別なオブジェクトを活用したりしつつ、いくつもの技術を組み合わせて作ったのが、次のような仕様のオブジェクトである。

ア　ウインドウオブジェクトの仕様

（ア）オブジェクトの定義

まず取り扱いたい働きをまとめる。

ａ　ウインドウ枠を生み出す。
ｂ　ウインドウの大きさを変える。
ｃ　ウインドウを移動させる。
ｄ　ウインドウの色を変える。
ｅ　ウインドウに文字を表示させる。
ｆ　ウインドウに長方形を表示させる。（この機能は継承の実習で取り入れる）
ｇ ウインドウ上の長方形の変形･色指定・位置指定を行う。(継承ののち実装)
ｈ　ウインドウの現在の状況を確認する。

（イ）　オブジェクトの部品の決定

上記の定義を実現するための要素、つまり部品を並べる。

ａ ウインドウオブジェクトの状態を記録する

ウインドウクラス(window_class)
ブラウザ表示枠オブジェクト(objIE)
ウインドウ名(namae)
入力表示文(hyoujibun)　**入力背景色**(iro)
ウインドウ上隅座標(objIE.Top)
ウインドウ左隅座標(objIE.Left)
ウインドウ幅(objIE.Width)
ウインドウ高さ(objIE.Height)
ウインドウ内の文字表示枠１(objIE.Document.getElementById("div-id-1").InnerText)
ウインドウ背景色(objIE.Document.bgColor)

ｂ ウインドウオブジェクトの操作方法(メソッド)をプログラミングする。

(a) **ウインドウクラスの名称指定→**set_namae()
(b)**ウインドウクラスの名称獲得→**get_namae()
(c)**ウインドウ内文字埋め込み→**moji_hyouji()
(d) **ウインドウ位置指定**　→　idou()
(e) **ウインドウのサイズ変更**　→　henkei()
(f) **ウインドウの背景色指定**　→ ironuri()
(g)**ウインドウ情報表示**　→ jyouhou_hyouji()
(h) **ウインドウを閉じる**　→　tojiru()

ウ　できあがったオブジェクトを使う。

(ア) オブジェクトを実体化（インスタンス化）する。　―ウインドウを開き名前を与える ―

例 Set w1 = new window_class
(ウインドウ w1 を実体化)

(イ) 実体化したインスタンスを操作する。

例 w1.henkei 400,150
(w1 の大きさを横 400 ピクセル×縦 150 ピクセルに変形する)

(ウ) 実体化したインスタンスを解放する。

例　Set w1 = Nothing (w1 をクリアにする)

（４）授業での実習状況

サイトに揚げている教材ファイルのようにウインドウオブジェクト処理を、ＬＡＮ教室支援システムで一斉転送し、各生徒に操作させた。

一つは、表示される問い合わせ画面を介して、自由にウインドウを操作できるプログラムで、生徒たちは、マウス抜きでプロパティとメソッドを操作している感覚を得ることができる。

もう一つは、ウインドウを自動的に 10 回実体化（インスタンス化）したあと、いくつかのウインドウを閉じるプログラムで、生徒たちはインスタンスの生成と廃棄を実感できる。

二つのプログラムを実際に操作し、エディタでコードを分析することで、生徒たちはウインドウの形で出現したＯＯＰの概念を会得することができる。理解に差が見られたが、よく分かった生徒

は自分なりにメソッドを組み合わせて、独自のウインドウ操作プログラムを実行していた。

（５）教材化の手応え

ＯＯＰ実習の定着度を測るため例のような問題を３０問ほど出題し小テストしてみた。平均点は５割弱の得点だった。興味深いことにいつも成績の良い生徒が高得点とは限らず、実習により本質的な原理を体得できた者が高得点だった。改善の余地はあるが､ＯＯＰ実習の有効性は実感できた。

小テスト例

次の説明にふさわしい用語を答えなさい。 （１）オブジェクトの設定を記述したもの。抽象的オブジェクトともいう。 （２）オブジェクトに内蔵した変数。 （３）オブジェクトに設定した自作の命令。

答（１）クラス（２）プロパティ（３）メソッド

今後も様々なオブジェクトを教材化してみたい。具体的に、**ファイル処理クラス、日時クラス、ＸＢＲＬクラス、携帯電話クラス、生徒クラス、計算機クラス、テレビクラス、分数クラス、サーバ応答クラス**などのオブジェクトモデルを設計しており、その概念クラス図をプロパティとメソッドを含めて私のサイトで公開する予定である。

興味を持たれた方がいらっしゃったら、得意な言語でクラスを作り、生徒に操らせる実習に取り組まれることをお薦めする。

３　オブジェクト指向の要点

ここまで説明してきたＯＯＰの要点を表にしてまとめる。用語を理解した上で改めて実習教材を分析してみると、より深くＯＯＰの概念がとらえられることだろう。よろしければ、どしどし教材として有効に活用していただきたい。

（１）要点表

まず、表２が生徒のための重要用語の要約表である。厳格な用語定義ではなく、実習に根ざして端的に理解できるように工夫した説明にしている。

（２）VBScript での命令

次に、表３がVBScript言語によるＯＯＰのための命令一覧である。実習を元に理解できるように、端的に機能を説明している。

（３）言語比較表

また表４が様々なプログラム言語におけるＯＯＰのための命令一覧である。いかに多くの言語がＯＯＰの長所を認め、言語仕様に取り入れているかがよく分かる。

以上のような概念を生徒に指導し、実習とともに指導すれば、ＯＯＰの基礎理解を図ることができる。目論見通りに高校生に理解しやすい要約になっているであろうか御高察いただきたい。

表２　オブジェクト指向プログラミングの重要用語

用　語	説　明
オブジェクト	プログラムでプロパティ（変数）とメソッド（命令）をまとめた処理単位。
クラス	オブジェクトの設定を記述したもの。抽象的オブジェクトともいう。
プロパティ	オブジェクトに内蔵した変数。
メソッド	オブジェクトに設定した自作の命令。
インスタンス	名前をつけてクラスを実体化したもの。複数作ることができる。
オブジェクトの廃棄	インスタンスを作れば作るほどメモリーを多く占有してしまうので、使い終えたインスタンスのメモリーは他で使えるようにクリアしなければならない。
カプセル化	インスタンス内のプロパティ（変数）は、自作の代入メソッドや取り出しメソッドを作らなければ使えないということ。これは欠点ではなく、変数の重複によるエラーを防ぐための、開発に便利な利点である。
継承	クラスを改造するときに、先にできているクラスの機能を再利用する仕組み。 例．扇風機クラスを元に乾燥風メソッド付きの新型扇風機クラスを宣言する。
多態性（多相性）	メソッドの使い方を何通りか設定できるようにすること。 例．スイッチメソッドで、扇風機1.スイッチ(″ON″) と命令すれば、スイッチが入り、扇風機1.スイッチ()と()内を空欄で命令すれば、元のスイッチの状態を切り替えるようにする。

表３　オブジェクト指向プログラミング用の VBScript 命令

VBScript の命令文	説　　明
Class クラス名　～　End Class	オブジェクトの設定（プロパティとメソッド）を並べる範囲。
Private プロパティ名 1, プロパティ名 2, ･･･	クラス内でプロパティ名を明示する。
Public Sub メソッド名(引数 1, 引数 2, ･･･) 　～ End Sub	クラス内でメソッド（命令）の動きをプログラミングする。
Set インスタンス名 = new クラス名	あるクラスをインスタンス名と名付けて実体化（インスタンス化）する。原則的にいくつも実体化できる。この実体化のタイミングで自動的に Class_Initialize メソッドが働く。
インスタンス名.メソッド名(引数 1, 引数 2, ･･･)	インスタンスが持つメソッドを働かせる。関数のように後に値を用意することもある。　例. SUZUKAZE.change_tsuyosa　3
Set インスタンス名 = Nothing	インスタンスを廃棄する。不要なインスタンスはクリアしてしまわなければ、メモリーが足りなくなっていく（**メモリーリーク現象**）。この廃棄のタイミングで自動的に Class_Terminate メソッドが働く。
private　プロパティ名 または メソッド名	プロパティやメソッドをクラスの範囲内でしか指定できないように秘密扱いに設定する。
public　プロパティ名 または メソッド名	プロパティやメソッドをクラスの範囲外でも指定できるように（公開するように）設定する。
Sub Class_Initialize 　～ End Sub	インスタンスを実体化するときに自動的に動くメソッド（**コンストラクタ**）。インスタンスの初期設定のためなどに使う。
Sub Class_Terminate 　～ End Sub	インスタンスを廃棄するときに自動的に動くメソッド（**デストラクタ**）。オブジェクト操作終了の合図のためなどに使う。

４　補足点

（１）ＯＯＰ実習での継承の実現

実はここまでの教材化では、**オーバーロード**や**オーバーライド**という用語と同様に、**継承**と**多態性について**は詳しく取り上げてなかった。その訳は、ＯＯＰのインスタンス操作に馴染ませることを主な目標にしたからである。そして、この VBScript 言語での継承は､特別な書き方で実現させる。紙面の都合もあり、要点だけ記載する。

ア　**コンストラクタ**の Class_Initialize **メソッド**内で、継承元のクラスをインスタンス化する。

例　Set oya = new senpuuki_class

イ　継承元クラスにあるすべてのメソッドを新たなクラスで再定義する。

```
例　Sub sw_onoff(sw)
        oya.sw_onoff(sw)
    End Sub
```

ウ　継承元クラスになかったメソッドを追加して定義する。

以上のような手順で継承を実現ができる。しかし、表４の命令比較表のとおり本格的なＯＯＰ言語はイの手順なしでも継承元のメソッドを使えるので、混乱させるおそれがある。

（２）　ＯＯＰで使われる図

大勢の開発者たちで協力しながらＯＯＰのシステム設計をうまく遂行するために、共通理解を図る図式化が有効である。この標準的な図式化をＵＭＬ(Uniformed Modeling Language)という。

ＵＭＬでは**クラス図**がよく用いられるが、米国の規格化団体ＯＭＧが定めたＵＭＬ図が**シーケンス図**や**ユースケース図**や**アクティビティ図**など１３種類ある。データベース構造を表記するＥ－Ｒ図と紛らわしいが、生徒の実習のためには、模範になる図を用意しなければならないだろう。

（３）　ＣＯＢＯＬ言語とＯＯＰ

ＣＯＢＯＬでプログラミングを教えながら実感することだが、基本的なファイル操作の手順は、オブジェクトの操作の説明に有効である。

つまり、ファイルのオープン・クローズはオブジェクトの実体化・廃棄に対応し、ファイルハンドルを指定した読込・書出命令は、インスタンスごとのメソッド操作に対応している。当然、各レコードの内容もファイルごとに違うように、各オブジェクトのプロパティ（変数）もそれぞれ別であり、カプセル化にあたる。

表４　プログラム言語別オブジェクト指向操作のための命令一覧表

操作	PHP 言語	Ruby 言語	Java 言語	C++言語

クラス宣言
（オブジェクト定義）
	Class クラス名{
	（インスタンス宣言）
	：
	Function 操作名{ }
	}
		Class クラス名
		def 操作名
		：
		end
		end
			Class クラス名{
			（インスタンス宣言）
			：
			Function 操作名
			{ }
			}
				Class クラス名{
				（インスタンス宣言）
				：
				操作名{ };
				};

オブジェクトの実体化(インスタンス化)	$インスタンス名 =new クラス名( );	$インスタンス名 =クラス名.new( )	クラス宣言 インスタンス名 = new クラス名 ( );	クラス名 インスタンス名; インスタンス名 .create( );
実体化オブジェクト （インスタンス)の 操作(メソッド)活用	$インスタンス名 ->操作名( )	インスタンス名 ->操作名( )	インスタンス名 .操作名( );	インスタンス名 .操作名( );

クラスの継承	Class クラス２
	extends クラス名１{
	：
	}
		Class クラス２
		extends クラス名１{
		：
		}
			Class クラス２
			extends クラス名１{
			：
			}
				Class クラス２
				：クラス名１{
				：
				};

実体化オブジェクト の 廃 棄	自動的に消滅。(ガベージコレクション機能) ただし、消滅時にしたいことを __destruct()メソッドに用意できる。	自動的に消滅。(ガベージコレクション機能)	インスタンス名 .destroy(); 自動的に消滅。(ガベージコレクション機能)	インスタンス名 .destroy();

同様に考えれば、変数を多重定義するREDEFINES句の使い方も、変数オブジェクトの継承にあたり、オーバーライドの説明に例示できる。さらに副プログラムを多重に呼び出す流れを組めば、クラスの連携に類似した動作になる。

このようにＯＯＰの記述ができなくても、操作対象を明確にして説明すれば指導できる。

（４）　新潮流

ここで平成２１年末時点での情報処理教育で注目しておくべき新潮流を挙げてみる。

ア　デザインパターン

ＯＯＰの特徴を理解しても有効に使いこなすためには、独特のパターンを会得しなければならない。それは、例えば旧来の手続き型言語のアルゴリズムの基本パターン（最大最小処理や順位付け処理やファイル更新処理など）を理解すれば、必要な要素と全体の構造を理解することで応用できるようになるようなものである。

そのＯＯＰの基本パターンの集大成が２３種の**ＧｏＦ（ゴッフ）デザインパターン**である。このＧｏＦデザインパターンは、１５年以上前に参考書籍３の中で取り上げられたのが最初で、その後、ＯＯＰのシステム設計の前提知識となっている。

本稿のモデルは、**プロトタイプパターン**と**アダプタパターン**に該当するが、まだすべてのパターンを学習しきれていないため、高校商業科でどれを取り上げて教材化するかは本稿では触れないので原稿末の参考書籍等で調べてみるとよい。

イ　クラウドコンピューティング

シンクライアント（ブラウザが十分に働く程度の性能と低コストの端末パソコン）とサーバとの連携で働く情報処理システムが広がりつつあり、もっと進歩した形が**クラウドコンピューティング**である。すなわち開発者も利用者も、端末のブラウザや携帯電話などでネットの奥に展開した仮想ＯＳを操作し、資源の集中管理、ハードウェア保守の簡素化、設備の拡充縮小の柔軟化を図る。

具体的に、グーグルはGoogle Appsを、マイクロソフトはWindows Azureを、ニフティはニフティクラウドをサービス供用している。グーグルのカレンダーやオープンオフィスのサービスは、個人が無料で利用できるクラウドコンピューティングであり、どのようなことができるかを体験するのによいサービスである。

クラウドとはサイバースペース内のとらえどころのない雲といった意味が込められた新しい情報処理用語で、クラウドオブジェクトと呼んでもよいであろう。クラウドのＯＯＰでは、間延びした応答はエラーになったり、データベースの併合が難しくなったりするなどの特徴に対応しながら開

発していく必要があるそうである。

実習するためには、まず教師自身がサーバサイドプログラミングとクライアントサイドプログラミングの技能が必要となる。私なりの開発教材をいずれかの機会に公開する予定である。

ウ　エージェント指向

オブジェクトを巧妙に設計すれば、まるで自立したロボットのように数々の要求に自動的に対応してくれるものができるであろう。そのような自立したオブジェクト部品（モジュール）たちが連携して処理を片付けていくようなシステム開発をしていくことを**エージェント指向**という。

表計算ソフトの関数でも、括弧内の引数の数が二つでも三つでもエラーなく動いてくれたり、強制終了したワープロソフトのデータが復元できたりするような動作に助けられることがある。このような動作がエージェント指向である。

実習するとしたら、エラー例外処理の組み込み処理や、ゲーム選手クラスの操作処理などが教材になるだろう。

エ　関数型言語

関数型言語とは、プログラミング言語の種類の一つで、カッコや演算式を基本的な記述要素とする言語である。**手続き型言語**の動作に対して、計算順位の後の式まで答えられることが確定してから処理するので、エラーを予防しやすい。具体的に Lisp や Scheme や Haskell という言語が利用者が多い関数型言語である。

ＯＯＰとの関係は説明しづらいが、関数の入れ子の手法や開発効率のよさが、カプセル化や継承やデザインパターンの考え方に通じるものがあり、正確に早くソフトウェア開発を進めたいという同様の発想から生み出された言語体系である。

正直言って簡単に教材化できそうにないのだが、あえて類推してみれば、簡潔な記述によって帰納的に処理を果たす**再帰プログラミング**のしくみが関数型言語の動作に似ている。簡単な再帰プログラミングの教材から指導してゆけば、関数型言語の指導の糸口になるであろう。

以上、情報処理技術の新潮流をかいま見せてくれる四つの用語を取り上げてみた。多少なりとも興味を持たれた先生は、さらに書籍やサイトで調べるとよいであろう。

５　ＯＯＰの汎用化（オブジェクト指向学のすすめ）

さて次のような例えで、ＯＯＰの理解を助けることができるのではないだろうか。魔法使いが呪文を唱えて魔物たちを呼び出し、呼び出した魔物たちにあれこれ命じて働かせる。クラスの宣言と、実体化したインスタンスの操作というＯＯＰ独特の構図をうまく例えられると思う。

このようなひゆは多用しすぎると説明したいこととあまり結びつかなくなるので、気を付けるべきだが、この章で述べたいことは、ＯＯＰの概念の一般化であり、文系の発想でとらえたオブジェクト指向学のすすめである。

カプセル化を身近な料理の例で例えれば、イモを茹でてから切ろうが、切ってから煮ようが、カレーはできるようなものである。

司法制度のひゆとして本年度導入されたばかりの裁判員制度も、本職の裁判官が判断していたときも、裁判員が加わって決めるようになっても判決は有効となり、正にカプセル化のモデルである。

別のひゆとして**三権分立の統治モデル**も、司法－行政－立法は古代社会における神官－貴族－王の形態の発展形ととらえることができる。クラスにあたる政府機関が連携して機能しているのであり、ＯＯＰのクラス同士の相互作用に似ている。

また**プロテスタンティズム**だけが勤勉さを肯定し、近代資本主義のインフラを整備できるほどの**余剰資本の蓄積**が可能であったとのマックス・ウェーバーの主張も、日本の近代化を図式化してみれば反証できる。すなわち石田梅岩の**心学**のような日本流の儒教精神が商人倫理を維持させ余剰資本を蓄積していった。その資本の蓄積を基盤とし明治維新を成し遂げたであり、欧州の宗教改革の申し子ではないのである。このように図式化した概念を、オブジェクト指向的に考察し、対応を準備する思考法が、ＯＯＰ的だと考える。

続いて野口悠紀雄氏の唱える**現代日本の戦時生産体制の継承説**が、ＯＯＰの継承のモデルとしてふさわしい考え方である。アジア・太平洋戦争後の奇跡的な日本の復興は、大政翼賛会的な戦時生産体制が、意識化させずに戦後まで継承され温存されたために、戦災後の日本で有効に働いたと分析している。これは、先に述べた江戸時代の社会

システムにまで根付いた体制と考えられる。日本という国家オブジェクトに戦争遂行という命令を与えれば着実に実行し、経済成長という命令を与えれば奇跡の成功を果たすというとらえ方である。

この実例は、現代のアフガニスタンやイラクの復興の遅れを対比すれば、生々しく切実なモデルである。日本経済がドルショックやオイルショックやバブル崩壊の危機を乗り越えてきた事実は、多態性のモデルとしてふさわしい。しかし、バブル崩壊後の構造改革により､その戦時経済体制が、時代の要請や多方面の構造改革によって、否応なしに高度情報化社会へと変貌してきている。クラスを組み替え新たなるメソッドを働かさなければならない時期である。

同様の観点で他の図式化も考えている。例を挙げれば、**官僚制オブジェクトモデル・球技オブジェクトモデル・生物進化オブジェクトモデル・共同幻想オブジェクトモデル・源平史観オブジェクトモデル・宗教組織オブジェクトモデル**などである。それぞれのクラス構成とプロパティとメソッドを的確に描けば、後の対処方法を検討することができるのである。

文系向きにＯＯＰの発想を広げてみたが、いかがだろうか。複雑な構造をモデル化し対処法を構想するのに有効なオブジェクト指向学の可能性が見えないだろうか。

結　び

以上、大仰な主題を掲げて、実際的かつ学術的に論述してみた。ＯＯＰは、新しい学習指導要領で求められる体験的な学習教材として有効なものになると確信している。

変革の２１世紀だからこそ原点に帰ろうではないか。教師の肝心の役割は、

- **生徒ができなかったことができるようになったときの達成感を共に喜んであげること。**
- **知らなかったことを知ったときの知的好奇心の満足を得る場所を用意してあげること。**

この二点であると私は考える。この役割を果たすために、我々は今後も進んだ見識を心掛けつつ、より創造的で意義ある教育活動を企画推進していかなければならないのである。

拙稿の研究を参考に志気高い先生方が身近な場で多彩な才能を発揮され、ますます有意義な商業教育を展開していかれることを大いに期待する。

参考書籍

1) Ｗｓｈクイックリファレンス　羽山　博著
平成 11 年 10 月　オライリー･ジャパン刊

2) オブジェクト指向でなぜつくるのか―知っておきたいプログラミング、UML、設計の基礎知識
平澤 章著　平成 16 年 6 月　日経ＢＰ社刊

3) オブジェクト指向における再利用のためのデザインパターン　エリック・ガンマ他著　平成 11 年 10 月　ソフトバンククリエイティブ刊

4) PHP によるデザインパターン入門
下岡 秀幸著 平成 18 年 11 月 秀和システム刊

5) Head First デザインパターン―頭とからだで覚えるデザインパターンの基本
エリック・フリーマン他著　平成 17 年 12 月
オライリー･ジャパン刊

6) プロテスタンティズムの倫理と資本主義の精神
マックス・ヴェーバー著　平成元年１月
岩波書店刊

7) 都鄙問答 経営の道と心　由井 常彦著
平成 19 年 10 月　日本経済新聞社刊

8) 1940 年体制―さらば戦時経済 野口 悠紀雄著
平成 14 年 12 月　東洋経済新報社刊

参考 Web ページ

1) 新しい高等学校学習指導要領
http://www.mext.go.jp/a_menu/shotou
/new-cs/youryou/kou/kou.pdf

2) 矢沢久雄の早わかり GoF デザインパターン
http://itpro.nikkeibp.co.jp/article
/COLUMN/20051123/225074/

3) グーグルのクラウドサービス案内
http://www.google.com/apps/intl/ja
/business/index.html

4) seastar3 オブジェクト指向プログラミング関係ページ
http://seastar.la.coocan.jp/ss/2016/09/10/オブジェクト指向プログラミングの指導について/