オブジェクト指向プログラミング教育法序説

seastar@orion.nifty.jp

目　　次

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の理解と進歩の方向性を知らせるのか、逡巡することが多い。

しかし、デカルトの言葉のごとく、踏み出していかなければ、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、教材化の試案をまとめてみた。さらにオブジェクト指向プログラミングについて考え、図式化するうちに、その複雑な事象をとらえ検討するための有効性に気づいた。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持論を展開してみる。

なお、Object-Oriented Programming を詰めて、以後 ＯＯＰ （オープ）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言語などのＯＯＰ言語が普及し、ソフト開発の効率化と堅牢性が求められるようになり、ＯＯＰ教育も避けて通れなくなった。新しい高等学校学習指導要領において、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。そして、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読める。

また、本年度の学会総会で西村調査官が科目の目標と改善の内容を講話して下さり、その資料で「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したと示されていたので、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。

ア　教材のダウンロード先の紹介

まず、実習用のファイルおよ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

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。

イ　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マイクロソフト社製ウインドウズでの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する。

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

そして、指定フォルダ内に保存された"p1.vbs"ファイルが水色の巻物のようなアイコンになっていることを確認の上、このアイコンを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、巻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、活用していただければ幸いである。ちなみに我が勤務校では、１年生の「情報処理」の３学期の授業で、３時間実習するようにして、軽快にデバッグやトレースやアルゴリズムを体験的に習得させている。

なお、セキュリティ設定でＷＳＨの実行が禁止されていたり、警告枠が表示される場合もあるので、その場合は設定を変更するか、実行できるパソコンの操作画面の提示だけにするなりかで対応する必要がある。

ウ　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。１０年余りの研究の末、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。

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(オブジェクト実体)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。

いきなり専門用語を並べられて、何度も読み返かえされ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汎用的な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プログラムを作るとすれば、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とになる。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、まちまちの命令を使い分けていくことになるだろう。このようにオブジェクト化というプログラム手法は、役割を分担したり、それぞれの性能を個別化するために有効な技術である。

以上のような考え方に基づき、実際にプログラミングしてみ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。

また､「涼風という扇風機を作る」コードは、

″ Suzukaze = new Senpuuki ″である。そして、「涼風の風量を２にする」コードが、

″ Suzukaze.change_tsuyosa(2) ″である。

以上の扇風機オブジェクトの例が、ＯＯＰの考え方である。補足すれば、ＯＯＰは、使うオブジェクトと作るオブジェクトがある。作ることの方

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

が当然難しいので、まずは使うオブジェクトとして、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学期で実習してみたところ、生徒たちはこの仮想の扇風機を概ね理解し、それぞれの扇風機を操作し状態表示させていた。つまり、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だった。

エ　ウインドウオブジェクトのモデル

一度モデル化を図ると、奇妙なもので、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。例えは、テレビオブジェクトや電卓オブジェクトなど創作すると興味深い。

もう一つのオブジェクト例として、ウインドウオブジェクトを説明する。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を説明しているときに、パソコン画面にいくつも重ねて出している複数の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､自分でも閃くものがあった｡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、直裁的なオブジェクトとして認識さ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、画面上に十数個並べてみたいと考え、このコードを開発した｡いくつもの技術を組み合わせており、細かい説明は省くが、ウインドウオブジェクトの要点は次のとおりである。

（１）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。

(ア) ウインドウ枠を生み出す。
(イ) ウインドウの大きさを変える。
(ウ) ウインドウを移動させる。
(エ) ウインドウの色を変える。
(オ) ウインドウに文字を表示させる。
(カ) 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)
(キ) ウインドウ上の長方形の変形･色指定・位置指定を行う。
(ク) 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。

(ア) ウインドウオブジェクトの状態を記録する変数(プロパティ)を決める。

扇風機名(namae) 電源オンオフ目印(onoff)
風力段階(dankai) 首振目印(kubionoff)
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)

(イ) 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) 実体化したオブジェクト(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) 実体化したオブジェクト(インスタンス)を解放する。

例　Set w1 = Nothing （w1 をクリアにする）

エ　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、２つのウインドウオブジェクト操作プログラムを、ＬＡＮ教室支援システムで一斉転送し、各生徒に操作させた。

一つはウインドウオブジェクトの各設定を問い合わせながら自由にウインドウを操作できるプログラムで、生徒たちはプロパティとメソッドをマウスを使わずに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、生徒たちにクラスとインスタンスの関係を理解させるために開発した。

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オブジェクトの基本を成すＯＯＰの概念を会得することができる。理解度には差が見られたが、本質的に理解できた生徒は、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操作していた。

オ　教材化の手応え

次に抜き出したような小テストを３０問ほど出題し、定着度を測ってみた。その結果、５割弱の成績で５割弱の成績だった。興味深いのはいつも成績のよい生徒が高得点とは限らなかった点である。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訳で、専門教育の意義を実感できる授業であった。

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

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の概念が捉えられることだろう。また各表を、教材としても有効に活用していただきたい。

（1）要点表

今回のＯＯＰ指導法確立のために作成した教材をいくつか挙げる。

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。

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	プログラムで変数（プロパティ）と命令（メソッド）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設定を再利用する仕組み。例．扇風機クラスに、加湿機能メソッドを追加する。
多態性	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

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	インスタンス名というクラスを実体化（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()内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	インスタンスをセットしたときに、自動的に動くメソッド。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。オブジェクト操作終了の合図のためなどに使う。

（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。

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	JAVA	C++

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();

以上のような概念を生徒に指導し、プログラミングできるようにさせれば、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も理解しやすい要約になっているであろうか。ご高察いただきたい。

４　補足点

（１）ＯＯＰ実習での継承の実現

実は実習教材において、継承と多態性は、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 vbs ファイルでの継承は、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　まず、コンストラクタである 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全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の動作を実現することができる。しかし、言語比較表で分かるように本格的なＯＯＰ言語はイの手順をしなくても継承元のメソッドを使えるので、混乱する心配があるのが問題点ではある。

（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OMG団体が規定している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。データベースの構造を表記するＥ－Ｒ図と紛らわしいく独特の表記なので、生徒に使いこなさせるためにはしっかりと指導しなければならないだろう。

（３）　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。私なりの表現なので、分かりやすい言葉を並べたつもりが逆に妙な先入観を呼び起こすかもしれないが、なぜここに４つの用語を取り上げたのか、趣旨をお汲み取り願いたい。

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デザインパターンである。このＧｏＦデザインパターンは、１９９４年に書籍「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」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。

本稿のモデルでは、プロトタイプパターンとアダプタパターンを活用しているが、まだ全てのパターンを学習し切れていないので、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。巻末の書籍等を参考にしていただきたい。

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、開発者と利用者共に、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

ろのない雲のような存在という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いなどの特徴を手なずけながら開発していく必要があるそうである。

実習授業のためには、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を活用できる技能が必要になる。その私なりの研究状況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）に連携して処理させるようにシステムを開発していく考え方をエージェント指向という。

単純に表計算ソフトの関数でも、括弧内の引数の数が２つでも３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が、エージェント指向的な動作である。

模試将来実習するとしたら、ゲームプレーヤークラスを宣言し、複数実体化したプレーヤー同士でゲームし、勝負が付くまで値を比較しながら自動的にゲームを繰り返す処理などが教材化できる。

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１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、あえて近いものを挙げれば、再帰プログラミングの簡潔な記述によって帰納的に処理を果たすしくみが関数型言語に動作に似ているので、簡単な再帰プログラミングの実習例題を指導することから取り組めば、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垣間見せてくれる４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。

このような比喩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つけた方がよいが、この章で述べたいことは、ＯＯＰの概念の一般化であり、文系の発想で捉えたオブジェクト指向学のすすめである。

カプセル化したオブジェクトを取り扱うこと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。

同様の例で言えば、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ようが、裁判員が加わって決めようが、判決としては有効であり、正にカプセル化の事象である。

別の例えで言えば、三権分立の政治モデルも、司法-行政-立法を古代社会における神官－貴族－王の形態が変化したものととらえることができ、民主主義社会の構成単位が連携して機能しているのであり、ＯＯＰのオブジェクト間の相互の影響に似ていると考える。

また別の例えでは、マックス・ウェーバーの主張するプロテスタンティズムだけが、勤勉さを肯定的に考えさせ、近代資本主義のインフラを蓄積できるだけの余剰資本を積み上げることができたとの主張も、同様の構図を日本の近代化に照らしてみれば、反証することができる。すなわち、石田梅岩が町人に説いた心学のような日本独自の儒教精神がプロテスタンティズムの代わりに商人倫理

を維持させて余剰資本を蓄積していった。その資本の蓄積によって明治維新を成し遂げるだけの基盤を備えていたのである。このようにオブジェクト化された概念を別の図式で論理付けしてみる思考法が、ＯＯＰ的だと考える。

もうひとつ、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も、とてもオブジェクト指向的な考え方である。その趣旨は、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いうものである。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。この実例は、現代のアフガニスタンやイラクの戦後復興の遅れと比較しても、生々しく身にしみるとらえ方である。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、多方面で時代の要請や自主的な改革によって、高度情報化社会へ否応なしに変容してきている。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。

さらにいくつも図式化した例を考えているのだが、それぞれのプロパティとメソッドを解説しつつ特徴を述べていくと凡長になりそうなので、言葉を箇条書きにするにとどめる。

オブジェクトモデルの例

・環境対策オブジェクトモデル
・官僚制オブジェクトモデル
・業績主義オブジェクトモデル
・雇用問題オブジェクトモデル
・アントレプレナーオブジェクトモデル
・主要簿と補助簿オブジェクトモデル
・金融工学オブジェクトモデル
・サイバースペースオブジェクトモデル
・スポーツオブジェクトモデル
・生体オブジェクトモデル
・生物進化オブジェクトモデル
・源平史観オブジェクトモデル
・陰陽五行オブジェクト
・宗教オブジェクトモデル　　など

大胆に論点を広げてみたが、いかがだろうか。複雑な構図を捉えて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、オブジェクト指向プログラミング教育法序説と大上段に構えた主題を掲げ、実際的かつ学術的に論述したが、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ＯＯＰは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とも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２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保ちながら、より創造的で機動力の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。

志気高い学会員の諸先生方も身近な場面から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ゆかれることを大いに期待する。

参考文献

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
3) 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
4) 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
5) 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
6) 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
7) 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) seastar オブジェクト指向プログラミング関係ページ　http://homepage1.nifty.com/tetsuhito/OOP-Labo/oop-menu.html

2) 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

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) 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/