Ｊａｖａプログラミング授業の薀蓄

－プログラミング教育における商業教育レガシーの継承－

平成29年度第65回全国商業教育研究大会
第3分科会発表資料

愛媛県立北宇和高等学校　教諭　松浦　哲仁

Ｊａｖａプログラミング授業の蘊蓄

－プログラミング教育における商業教育レガシーの継承－

目　　次

Ｊａｖａプログラミング授業の蘊蓄
－プログラミング教育における商業教育レガシーの継承－

©seastar@orion.nifty.jp

Ⅰ　序

現行の指導要領の高校商業科目「プログラミング」で新たにＪａｖａ言語が取り上げられ、私も平成２５年度の授業で実習を交えて指導した。配列やクラス内関数など旧のＣＯＢＯＬ言語のプログラミングの考え方が生かせることもあるが、インスタンスやフレームワークなどの扱い方など使ったことがないと実習指導も問題解説もできないこともある。

我々商業教員は、それぞれ自分なりの商業科目の指導法のレガシーを積み重ねて指導に役立てている。一般的なものならば、「借方はカリカタのリの左はねで、貸方はカシカタのシの右はねでイメージ」とか「通貨や単位の計算は、違う名数なら掛けて同じ名数なら割る」とか教わり真似たりしている。拙者のメソッドでは、前払前受・未収未払で「ミシン（未収収益）を前歯（前払費用）でカリッ(借方)とかじる。見栄え（未払費用）のええマイク（前受収益）を貸す(貸方)。」とか「ネットワーク機器は性能の低い順に、フランス語の本の意味のリーブル（リピータ・ブリッジ・ルータ）だ。」とか、自分なりの工夫で授業し、同僚たちにも広めている。

今回の研究報告では昭和の後期からプログラミング教育に携わってきた上でのレガシーを、Ｊａｖａ言語によるプログラミングの指導に生かせそうな蘊蓄として編集し個別に取り上げた。仮説検証結論の論文の流れは採っていないが、昭和時代から商業教育に携わってきた先生方が読んだだけで概念が想起できるような説明を心掛けた。各章が独立した内容なので、興味がある部分だけ読んでも役立つと考える。

拙稿が２１世紀のプログラミング教育に橋渡しする生きたレガシーとなり、僅かなりとも皆様の授業で生かせたり、プログラム開発のヒントになったりすれば幸いである。

Ⅱ　本　論

１ ファイル操作とオブジェクトのインスタンス化

まずはＣＯＢＯＬやＢＡＳＩＣのプログラミングを指導してきた先生方を想定して、ファイル操作を例にＪａｖａ言語のオブジェクト操作プログラミングの考え方を説明しよう。なおこれよりオブジェクト指向プログラミング（Object Oriented Programming）をＯＯＰと略して表記する。

ＣＯＢＯＬやＢＡＳＩＣは手続き型言語であり、ＪａｖａはＯＯＰ言語であると分類されるが、これらの中にＯＯＰの考え方と同じしくみが活用されているのである。

ここでは、ＣＯＢＯＬプログラムでのＯＯＰ的動作を２点取り上げて、クラスやインスタンスの考え方をイメージできるように解説してみる。

（１）ＣＯＢＯＬ言語でのファイル処理のＯＯＰ的な動作

第１点はテキストファイル操作のＯＯＰ的動作である。ＣＯＢＯＬでの順次編成ファイルの１レコードを読み込む命令文は*"READ ファイル名"*で、追加書き込みの命令は"WRITE

レコード名"である。その前提として、"ENVIRONMENT DIVISION"でファイルの物理パスを明記しておくことが必要であり、もし複数のファイルを操作したい場合は、その明記する記述もそのファイル分だけ必要になる。そして自然に考えれば、それぞれのファイルを読み込む命令文は、*"READ　ファイルA"*,*"READ　ファイルB"*などとなる。さて、それがＪａｖａ言語において同様にそれぞれのファイルを操る場合は、*"ファイル A.READ()"*, *"ファイル B.READ()"*といった記述になる。単純に逆さまにするととらえてよいが、英語と日本語の文法の違いを考えてもらえば、*「読め、ファイルを.」*という英語の命令文の語順と、*「ファイルを読め。」*という日本語の命令文の語順との対比を授業で示せば､印象が深まる。さらに補足すれば、日本語は動詞の前の目的格や形容詞などをスタックした後、後入先出法（ＬＩＬＯ法）で文法解釈する言語なのである。

さらに別の面から説明してみよう。ＣＯＢＯＬプログラムでは操作対象のファイルＡやファイルＢは、事前に環境部でＯＳ側に使用申し込みの記述（アサイン）しておき、この設定が正確ならばファイル処理用の数々の命令（OPEN,CLOSE,READ,WRITE...etc）が使えるようになる。

このしくみを機械語レベルのしくみで考えると、申し込みの記述をすることで、ファイルの先頭場所や大きさや現在の読込位置や読み書き区分などの変数（プロパティ）をそれぞれのファイルごとに用意することになる。

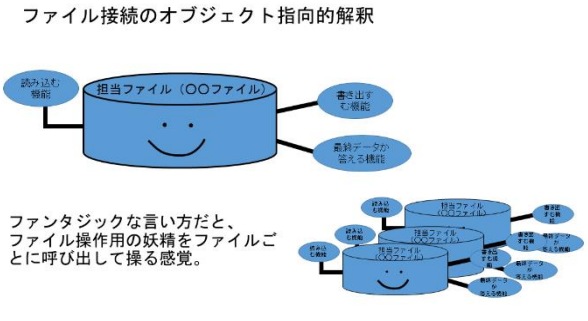


図１　オブジェクト指向的なファイル操作

つまり、ファイルＡの設定やファイルＢの設定が個別にメモリー上に確保されるのである。その個別の設定を読み書きし操作していくのが、ファイル処理用の前述のいくつかの命令である。これはファイルを対象にしたものでなければ使えない命令であり、ソート処理用においてもソート用の設定を準備し、専用の RELEASE 命令や RETUREN 命令を働かせる。

図１で示したように、旧言語でファイル処理の道具としてファイル名を扱うことは、ＯＯＰのインスタンス操作と同様の動作なのである。

（２）ＯＯＰ操作の手順とその際のＪａｖａ言語での記述

ではＯＯＰの考え方をＪａｖａ言語流にイメージ化できるように説明してみよう。ＣＯＢＯＬ言語でのファイル処理は、ＯＯＰ的に説明するとＡＳＳＩＧＮ句でファイル命令が使えるように実体化（インスタンス化）し、手続き部で*"ファイル操作命令　ファイル名"*との形で命令文を指示しつつ、それぞれのファイルオブジェクトを操る。

これをＯＯＰ言語であるＪａｖａで操作するときは、

"ファイルオブジェクト型　ファイル名 = new ファイルオブジェクト（ファイルパス）"

と、*" = new "*を用いて設定を合図する。この設定の後で、*"ファイル名.命令（）"*の形の命令文を指示しつつ、それぞれのファイルオブジェクトを操る。この点について重ねてい

うが、ファイルオブジェクトが準備している命令（メソッド）でなければ使えないのである。プログラムが用意したオブジェクトを使うということは、この考え方に慣れる必要がある。そして、Ｊａｖａプログラム用に用意された様々なオブジェクト（データベース、ネット接続、時間計算、ウインドウ描画など）の操作例を学びつつ、メソッドを使いこなせるようになることが、Ｊａｖａプログラマにとっての重要な課題なのである。後の６章で具体的にウインドウオブジェクトの操作とその教材化について述べる。

（３）ＯＯＰにおける使うオブジェクトと作るオブジェクト

以前、日本商業教育学会論集第２０号で「オブジェクト指向プログラミング教育法序説」の題名で報告したことだが、ＯＯＰプログラミングにおいてオブジェクトには**使うオブジェクト**と**作るオブジェクト**がある。ファイル操作のオブジェクトのように、当然に使うオブジェクトの方が分かりやすいし、ＯＯＰの基礎を学ばせるときに有効である。

ＯＯＰの基礎的な概念を植え付けた上で、作るオブジェクトの技術を習得させれば、自由にＯＯＰを操れるプログラマを育成できる。そのレベルに到達させておけば、検定試験の指導においてもＪａｖａプログラムの穴埋め問題のクラス定義側の動作とメインメソッド側の動作を読み切って問題が解けるようにできるのである。

私は「プログラミング」の授業のＪａｖａ実習として、eclipse 環境で

・tasu(値)メソッド	・hiku(値)メソッド	・kakeru(値)メソッド
・waru(値)メソッド	・kotaeru()メソッド	

を並べた電卓オブジェクトを打ち込ませ、その後、main メソッドを用意して計算処理クラスで計算機１インスタンスと計算機２インスタンスを実体化させ、自由に計算させた。電卓オブジェクトは２０行たらずのコードなので、タッチメソッドが滑らかな生徒たちなら、１時間で実習できる。

もっと発展した実習をするとすれば、kakeru メソッドと waru メソッドは電卓２クラスで加減算しかできない電卓１クラスを継承させて計算処理するとか、計算機インスタンスを配列で用意して計算させることができる。このような実習を進めていけば、生徒のＯＯＰ感覚をより深く伸ばすことができる。

このように、作るオブジェクトの理解のためには、実習が不可欠であると考える。

（４）ＣＯＢＯＬのＲＥＤＥＦＩＮＥＳ句のＯＯＰ的な動作

次に第２点として、ＣＯＢＯＬのＲＥＤＥＦＩＮＥＳ句の用法が、オブジェクトの継承や参照渡しの考え方の指導に生かせることを説明する。

ＣＯＢＯＬ言語では、あるフィールド（変数）に対してＲＥＤＥＦＩＮＥＳ句を用いて重ねて別名で指定できる。例えば１００文字の長さのあるフィールドを、５文字の長さの２０個のテーブルのフィールドとして重ねて指定して、どちらのフィールド名でも呼び出しや書き換えができるようにする。

この特徴をＯＯＰのしくみと結びつけて考えれば、**オブジェクトの継承**のしくみと関連がある。Ｊａｖａプログラムでも、継承の機能で先に作ってあるオブジェクトを踏襲したり再設定したりして、プログラミングを効率的にする工夫が準備されている。

そして、別の変数名でも実は同じ記憶場所を指定していることがある。具体的に開発時

に困ることが多いのが、変数のデータを別の変数にコピーしたはずが、最初の変数の記憶場所に別の変数名も追加したことになった場合である。そうなったときは、別の変数を書き換えただけで、最初の変数の内容が予定外に書き換えられてしまい、デバックしてもその原因がなかなか見つからないようなことが起こるのである。なおこの点は、第４章で詳しく説明する。

以上のようにＣＯＢＯＬのＲＥＤＥＦＩＮＥＳ句の考え方を知っていると、ＯＯＰの継承や参照渡しの原理が理解しやすい。温故知新の心意気で、ＯＯＰの習得を図っていこう。

２ 配列の添字の０基底と１基底

Java 言語や C 言語では、番号をつけて操る配列の添え字や繰り返しの初期値は 0 から始めるのが標準である。これは COBOL で１番から数える配列（テーブル）になれたプログラマには違和感があるが、新教科においてはこの新方式で統一して教えなければならない。

教え方としては、地上０メートル、１メートル、２メートルと数えるか、ビルの１階、２階と数えるかとの意識の違いであると教えると分かりやすい。蘊蓄らしいことを述べれば、欧州の言語においてビルの何階かを数えるとき、日本語での１階を地上階、２階からを１階・２階と数え、この Java 言語の添え字の数え方と共通している。

現実的には機械語の番地を指定する際に、スタート地点の番地を選んだ後で、その番地は０、次の番地は＋１、その次は＋２と考えるので、０基底の添え字処理は機械語の考え方に追随したやり方なのである。

また、カウントダウンの操作について、カウントアップしながら、カウンタを個数から引いていく形で実現することがよくある。これも０基底と１基底に応じてアルゴリズムを考えてみると、

ループがカウンタｉを定義して０基底で増加させる前提（初期値：０、終了条件：配列件数未満の間、ループごとの増分：＋１の設定）で、

０基底まで戻る場合　**カウントダウン用の変数　ｒ ＝ 配列件数－ｉ－1 で置き換えループする。**

１基底まで戻る場合　**カウントダウン用の変数　ｒ ＝ 配列件数－ｉ で置き換えループする。**

という処理の流れになる。

必ずデバッグのときのテストデータに境界値のデータを用意してエラーにならないか確かめなければならないアルゴリズムである。

表計算ソフトのセルなら、始めの連続する２つのセルで好きなように増分を設定してオートフィルすれば、早見表のような連続した表が完成するが、プログラミング的にデータを扱う際には、この添字のトレースが自由にイメージできる訓練が必要になる。プログラミング実習の際には心掛けさせたいポイントである。

３ ＳＱＬオブジェクトの操作のコツ

Java で SQL を操るためには、ウインドウズＯＳの場合は、ＯＤＢＣ設定をしなければならない。

または、ACCESS ファイルを操作する際には UcanAccess というライブラリを取り込み接続することもできる。私の活用する環境ではウインドウズ８でのＯＤＢＣ接続がうまくいかなかったので、UcanAccess ライブラリを Eclipse 環境で設定し、アクセスの accdb ファイルを操作した。

まずは全商協会が平成２５年２月に公開した情報処理検定試験サンプル問題の１級プログラミング部門Ｊａｖａの問７の駐車料金の曜日別集計プログラムの出題例で、SQL オブジェクトに executeQuery(SQL 文)メソッドを働きかける記述が出ている。実際にＯＤＢＣ設定で動作させてみたが、この専門的な設定および実習は複雑に感じられた。現実の平成２６年度のプログラミングの授業では、検定指導のみで SQL 操作の実習はしそびれたが、オブジェクトの宣言文と主なメソッドを指導すれば実習できるノウハウは確立した。

せっかく情報処理検定の３級からリレーショナル型データベースの原理と操作について指導しているのだから、ぜひ eclipse 環境で Java プログラミングにおける SQL オブジェクト操作を自由に実習できるようにしてあげたいものである。

４ インスタンスの配列の参照渡しの留意点

全商情報処理検定プログラミング１級の指導で苦労したのが、配列の形でのオブジェクトの実体化（インスタンス化）である。例えば第１章で例示した電卓オブジェクトを電卓Ａや電卓Ｂなどと名付けてインスタンス（オブジェクト実体）として操作するレベルを高めて、電卓[1]や電卓[2]などと添字を介した兄弟オブジェクトを多数操作する。

この配列の操作において留意しなければならないのが、オブジェクト間の単純な代入が思ったような動作にならないことがあることである。

予備知識のない方は混乱するかもしれないが、変数やインスタンス（実体）の内容の受け渡しには、値渡しと参照渡しの２通りのやり方がある。**値渡し**は、素直に元の変数等の内容を別の変数等に複写することである。**参照渡し**は、元の変数等の記憶場所を別の変数の設定にも真似させて、あたかも同じ記憶場所に別名を付けたような状態にすることである。それぞれの長所と短所を説明すれば、値渡しはそれぞれの変数を独立して操作できるが、大量に繰り返した場合に時間が掛かる。一方で参照渡しは、繰り返し処理で時間が節約できるが別の変数名で内容を書き換えると元の変数名で使うときに間違えた値を使ってしまう危険がある。

関数の引数の呼び出し側と受け取り側は値渡しであり、干渉の心配はない。ところが配列においては、例えば*"kazu[1] = kazu[2]"*のように"="での代入は値渡しではなく参照渡しになる。値を転送するのではなく、その配列要素が設定されている記憶場所の情報を上書きしてしまう。従ってこの命令文の後では、kazu[1]を操作しても kazu[2]と同様の場所を操作することになる。逆に元々の kazu[1]の記憶場所は消し去られ、もはや何が保存されていても呼び出せない状態が発生する。

意図的に値渡しをしたい場合は、swap()メソッドを使う。この場合では*"swap(kazu[1], kazu[2])"*と命令文を組み込めば、内容を交換してくれるのである。

実際の授業で反応がよかった例を示す。複数の生徒が携帯電話を持っていて、その携帯電話を交換して持ち主が変わったとする。するとある生徒に電話したら、当然別の生徒に

掛かる。それに対して実物の電話は移動させずにお互いの電話番号だけ書き換え手続きをしたとすれば、物理的な交換は起こらずに電話が別の生徒に掛かることになる訳である。このときもし、手違いで一方の生徒の番号が元の携帯電話にも残り、相手の携帯電話にも登録されてしまったとしたら、どのようなことが起こるであろうか。元の生徒の番号に電話したら２つの携帯電話が鳴り、相手の生徒の番号に電話したらどこの携帯電話も鳴らない。

参照渡しでは、この例えのように実体そのものは交換されず、ある実体に別の番号を振っただけの結果になる。つまり複数のインスタンス名が同一の記憶場所を扱う現象が発生する。したがって特にインスタンスの並べ替え処理のプログラムを分析するときには、参照渡しの効果について心得ておくべきである。

５　フローチャートとオブジェクト関連図の説明法の案

第５章ではＯＯＰ用の流れ図を考案する。まず現状の手続き型言語用の流れ図は以下の点で限界がある。

①メインの流れの中で、個々のインスタンスにどんな操作をしたか分からない。

②各インスタンスの操作で戻り値なし（void）か、戻り値があったときに何に渡しているかが分からない。

③呼び出される側のオブジェクト（抽象クラス）でどんな変数（プロパティ）が用意され、何をするメソッドが使えるのかが分からない。

④オブジェクト（抽象クラス）の継承とインターフェイス継承の関係が分からない。

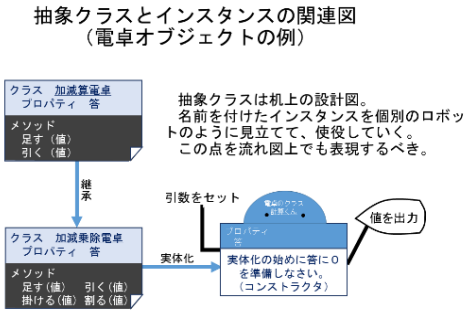


図２　抽象クラスとインスタンスの関連図

私がＪａｖａプログラムの流れ図化で困ったのは、この４点である。

③については既存の UML のクラス図を活用すれば対処できるが、抽象クラスの内部の流れと、メインメソッドでインスタンス名を決めて実体化してからの動作とを整理して指導するのが大変まどろっこしかった。

図３　インスタンス操作の戻り値の取り扱い

そこで私流の表記を考案してみた。

図２のように抽象クラスが保持するプロパティとメソッドを列記し、継承やインターフェイスの関係もＵＭＬ図のように線と矢印で表現する。

さらにメソッドが戻り値を持つ場合は、図３のような形で描き分ける。

これを流れ図の例を示せば、図４のようになる。加減乗除のメソッドを持つ加減乗除電卓クラスを、***赤電卓***と***黒電卓***と名付けて２つ実体化（インスタンス化）し、各々に計算させている。さらに２つの電卓の答を変数Ａに足し込み、Ａの値が１０以上ならば、***赤電卓***

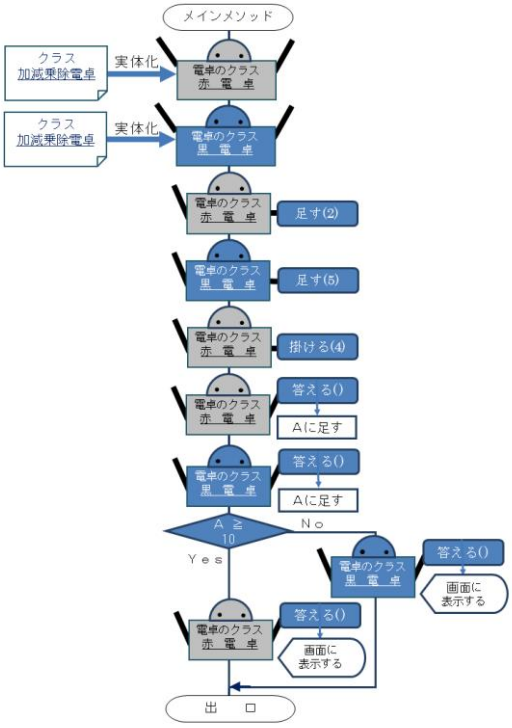


図４　松浦式ＯＯＰ記号を用いた流れ図の例

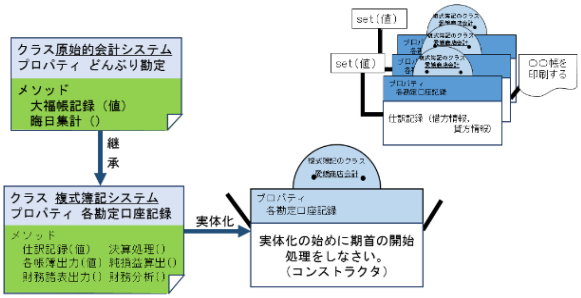


図５　簿記の概念の抽象オブジェクト化

の答を表示させ、そうでなければ***黒電卓***の答を表示させる流れである。

もう一つＯＯＰの表現例として、簿記のオブジェクトの図式化を示す。

原始的などんぶり勘定の収支計算を、より高度で緻密な簿記システムに編成したのか複式簿記であり、それを個々の企業の簿記会計に適用した帳簿や財務諸表を**インスタンス**として実体化するのである。図５のように簿記の記帳において開始記入が**コンストラクタ**の動作であり、決算処理が**デストラクタ**の動きにあたる。

簿記は現行の複式簿記だけではなく、様々な流儀の記帳方法をとることも可能であり、その際は異なったやり方を設定するように継承し、当事者に使い勝手のよい会計処理を実現できる。

以上、Ｊａｖａ言語やＣ＋＋言語のＯＯＰプログラムを説明する図式化として参考にしていただきたい。

６ フレームオブジェクト自由自在

さて第６章では、第２章で予告したように画面上で操るオブジェクトの解説をしよう。Ｊａｖａ言語では、画面上のウインドウやウインドウ内部のフレームなどを操作するツールのクラスライブラリとしてＡＷＴクラスライブラリやＳＷＩＮＧクラスライブラリが用意されている。

簡単にいうとＡＷＴの改良版がＳＷＩＮＧだが、実教出版の「プログラミング」の教科書ではＳＷＩＮＧを活用したウインドウプログラムを多数取り上げている。指導書の指導ＣＤ－ＲＯＭから使わせていただき、画面操作の基本からとてもよく分かった。これからのタブレット端末でのアプリ開発を想定してのＳＷＩＮＧへの傾倒だと推察するが、先見性を讃えたい。

このライブラリを活用できるようにしておくと、スマートフォン画面やタブレット機画面を操作するプログラムを作る際にも応用が利く。

その技術を応用して、授業で生徒たちがウインドウを自由に操れるようにしてあげたい

と考えた。この構想はすでにVBSファイルで実装し、旧課程の授業や各種の研究発表大会で披露してきた。今回、Ｊａｖａ言語版のウインドウオブジェクトの操作を狙い、１０時間ほどを費やし実習用のクラスを開発した。

私の設計したウインドウオブジェクトクラスは以上の仕様で、７個のプロパティを準備して１９個のメソッドを用いてこのプロパティを操る。

仕様を述べると，

（１）クラス名：Window_Class3

（２）使用プロパティ

ア．ウインドウタイトル：namae
イ．ウインドウ内表示文：hyoujibun
ウ．背景色：iro
エ．ウインドウ上端位置：wtop
オ．ウインドウ左端位置：wleft
カ．ウインドウ幅：wwidth
キ．ウインドウ高さ：wheight

（３）ウインドウクラス操作用メソッド

ア.ウインドウタイトル取得:getNamae()
イ．ウインドウタイトル指定：setNamae(String namae)
ウ．ウインドウ内表示文取得：getHyoujibun()
エ．ウインドウ内表示文指定：setHyoujibun(String hyoujibun)
オ．背景色取得：getIro()
カ．背景色指定：setIro(Color iro)
キ，ウインドウ上端取得：getWtop()
ク．ウインドウ上端指定：setWtop(int wtop)
ケ．ウインドウ左端取得：getWleft()
コ．ウインドウ左端指定：setWleft(int wleft)
サ．ウインドウ幅取得：getWwidth()
シ．ウインドウ幅指定：setWwidth(int wwidth)
ス．ウインドウ高さ取得：getWheight()
セ．ウインドウ高さ指定：setWheight(int wheight)
ソ．色指定（８色の番号）：irogime(int n)
タ．色指定（ＲＧＢ値）：irogime(int r,int g,int b)
チ．ウインドウの変形：henkei(int x,int y)
ツ．ウインドウの移動：idou(int x, int y)
テ．ウインドウの情報表示：jyouhou_hyouji(String stc)

メソッドに関してひとつ補足しておくのが、ソの色指定のirogime()メソッドは８色（黒、青、赤、紫、緑、水色、黄、白）の色番号一つだけの引数を指示する形式と、ＲＧＢコードを０～２５５の十進数で指定し、三つの引数を指示する２通りのメソッド形式を用意している。これが、**多態性（ポリモーフィズム）**の実装であり、生徒たちにもこの点を示して、多態性（ポリモーフィズム）の特徴を認識させるとよい。

図６　ウインドウオブジェクトの実体化例

このウインドウオブジェクトから１０個のインスタンスを発生させ、位置や色や表示文字を指定し操るプログラムの実行例が図６である。

ウインドウオブジェクトを教材としてオブジェクトは名前を付けて呼び出してから活用すると教えたとき、ある生徒がいったい何のためにそんなまどろっこしいことをするのかと質問してきた。そこで答えたのが、オブジェクトを改良していくときに手間が省けるからということと、既に用意された有効な機能を操作するときにこの手順を使うことになるからということであった。想定されるＱ＆Ａとして披露しておく。

この教材ファイルは次の URL でダウンロードできるので、ぜひ活用して生徒たちにオブジェクトの操作を実感させてあげて欲しい。

オブジェクト指向プログラミング入門キット 2

http://seastar.la.coocan.jp/ss/download/oop-practice-kit2.zip

７ トレンド

（１）アプリ開発

様々な開発環境は次の節で述べるが、携帯端末のアプリ開発は次のような手順になる。

a　アンドロイドスタジオ等のフレームワークで標的の携帯端末の疑似動作画面（エミュレータ）を準備する。

b　操作のための画面デザインをＧＵＩで組み込んでいくか、エディタにプログラムを打ち込んでいくかして作っておく。

c　画面デザインを生かして、操作によりプログラムが動作するようにイベントごとにプログラムを埋め込んでいく。その際にエラー対策やセキュリティー対策を実装するようにする。

d　デバッグやテストランを経て完成させる。

e　取扱説明を編集しアプリと同じフォルダに入れる。

f　必要とする利用者を探し、有効に活用してもらう。

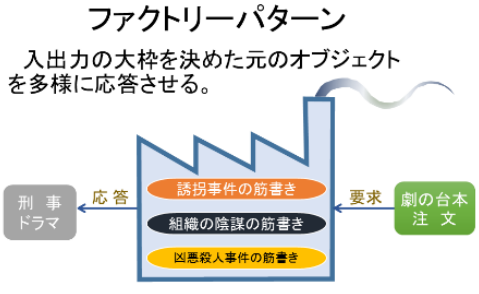
このような手順でアプリ開発と配布ができる技能を養成する必要がある時代である。

（２）デザインパターン

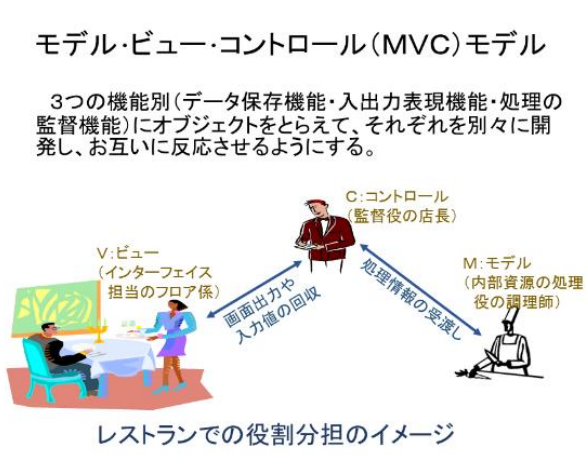
ＯＯＰプログラミングにおいて、将棋や囲碁の定跡に例えられるのがデザインパターンである。ここで簡潔に４つのデザインパターンを取り上げてみる。

例．梅の実のヘタ取り

まず**イテレータデザイン**は、データの集合を一つずつ最後まで取り出す手順のパターン化である。次に**ファクトリーデザイン**は、処理の結末のみを指定してデータ形式や途中の動作は融通が利くようにリフォームできるパターンである。また**アダプターデザ**



例．ステレオタイプの刑事ドラマ



インは既にできあがったオブジェクトを元に前後の処理を付け足して都合のよい入力並びや出力形式をとらせるようなパターンであり、例えれば海外用の電源コンセントアダプタのような働きを果たす。最後に**ＣＭＶモデル**は、処理の役割を制御担当とデータ管理担当と画面表示担当の３社の役割分担に分けてプログラミングしていくパターンである。

ここで取り上げた４つのデザインパターンのうちで私が特に活用しているのがＣＭＶモデルである。現任校でのシステム開発において、XAMP 環境での Web プログラミングを応用した部活動データベースを運用しており、そこで php プログラムによる動作制御と MySQL でのテーブル管理と Apache サーバーでの Web ページデザイン設定の際に、このＣＭＶモデルに沿って開発した。さらによりパターン化を進めるために Ruby on Rails の技術を習得中である。

（３）ラムダ式

配列やリストや連想配列などのデータの集合に対して一つももれなく同様の操作をしたいとき、Ｊａｖａ８ではラムダ式という簡潔な表記法をとる。

例えば得点配列が 40 人の得点を記憶しているとする。これを元に５段階評定を別の５段階評定配列に同じ順番に並べるプログラムを作る。その際に、指定した得点を元に評定判定し、その値評定を５段階評定配列における対応する添え字の位置に代入するメソッドを***"評定登録処理（）"***として作っているものとする。

これまでの標準的な繰り返しなら、

```
for(int カウンタ = 0; カウンタ < 得点配列.size(); カウンタ++ )
    評定登録処理( 得点配列.get(カウンタ) );
```

のような形式で書く。

これが新たに採用されたラムダ式だと、

```
得点一覧配列.forEach( 個人得点 -> 評定登録処理( 個人得点 ) );
```

のようにより簡潔に書く。つまり個数や添え字を介さなくても繰り返しが完結するのなら結果オーライという発想が込められている。

ここでランダムに３つのトレンドを述べたが、見逃した重要なテーマもあろうし、新たなトレンドもあがってくるであろう。ＩＣＴの新技術の情報収集を継続的に心掛けて、新たな教材化を推進していかなければならない。

Ⅲ　結　び

以上のようにＣＯＢＯＬプログラミング指導のレガシーを継承しつつ、Ｊａｖａ言語を

指導する際の実用的な考え方と実習案を述べてみた。ＯＯＰの会得と指導法の確立のために少しでもこの報告がお役に立てば幸いである。

さて、率直に言ってクラウド時代、フレームワーク開発の時代に商業科の情報処理教育は停滞しているのではないかと危惧している。当然、教員の自己革新も必要ではあるが、ここ十数年間のデフレによる教育界の予算配分の低さがこの状況を招いている。我々はこの構造主義的な呪縛に馴らされた思考法のままで淀むことなく、現場で知恵を絞り出して実のある教育内容の改善を図り、新たなイノベーション時代に出遅れないように心掛けておかなくてはならない。

商業科の情報処理教育の過去のレガシーから２１世紀のＩＣＴ教育への橋渡しを果たすべく、私も今後の研究と実務への導入を継続し情報処理教育を推進していきたい。ぜひとも諸先生方の各分野の貴重なレガシーも死蔵することなく継承させて、各指導現場で実践成果をあげていこうではないか。

参考書籍

1) 最新プログラミング オブジェクト指向型言語
中澤興起 監修　実教出版刊

2) Ｊａｖａ言語で学ぶデザインパターン入門
結城浩 著　ソフトバンク刊

2) Ｊａｖａスーパーリファレンス
山田祥寛 著　秀和システム刊

3) Ｊ２ＥＥパターン第２版　Deepak Alur/John Crupi/Dan Marks 著　日経ＢＰ社刊

4) 帳簿の世界史　ジェイコブ・ソール著　文藝春秋刊

5) Ruby on Rails5.0 初級①②Kindle 版
黒田努著 オイアクス刊

参考 Web サイト

1) 小生の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/OOP-Labo/oop-menu.html

2) UCanAccess ダウンロードページ
http://sourceforge.net/projects/ucanaccess/

3) ドットインストール　http://dotinstall.com/lessons

4) Swing を使ってみよう
http://www.javadrive.jp/tutorial/

補足資料Ａ 商業教育の蘊蓄９題

１　商の字源

商の字形は、高台に住む人たちを意味する。中国の古代に高台で優雅に暮らしていた旧勢力の殷の民が、新しい周王朝によって周辺に追いやられ、生活のために商売を仕事にした。ユダヤ人や新教徒も同様に迫害を受けて新しい国やビジネスを創造した。いつの世も一歩踏み出すためには何かを捧げなければならない。進んで踏み出すか、やむを得ず踏み出すか、ビジネスの意思決定の重要さを教えてくれる。

２　貨幣の殺菌効果

金属は鉛中毒や水銀中毒のように有害な面があるが、そのイオン効果により殺菌作用がある。１円玉を靴に入れて置いておくと消臭効果がある程であり、通貨として金や銀や銅の金属貨幣が流通しているのは図らずして伝染病の流行を抑制する効果があるのである。

３　民話　小僧の猫

昔ある大商人の奉公人に猫を可愛がっている孤児の小僧がいた。大商人が奉公人たちに貿易船に載せる商品を一つ差し出すように命令した。貧しい小僧は唯一の家族ともいえる可愛い猫を差し出した。他の奉公人たちは哄笑したが、後に交易から帰ってきた大商人は、ネズミの繁殖に困っていた遠い国の国王に大金で売れたことを知らせてくれて、お礼の代金も小僧に払った。小僧はそれを元手に商売を始め、立派な商人になった。

４　民話　わらじ長者

昔ある貧しい男がさまよっていたが、落ちていたワラジを拾い誰かに売りつけてやろうと考え、川で洗った。するとワラジの底から砂金がこぼれたのに気がついた。男はさっそくタダでワラジを交換する店をはじめて、旅人に喜ばれ、自分も砂金を集めて裕福になった。

５　明治維新と樟脳交易

幕末の頃、日本の樟脳を薩摩藩が密貿易としてイギリスに輸出していた。産業革命後、毛織物が流通していた西洋では防虫剤としての樟脳が貴重だった。木綿中心の日本では商品価値の低い樟脳をイギリスに高い値段で輸出することで、薩摩藩は国力をつけ明治維新を成し遂げた一因になった。

坂本龍馬は１８６２年（文久２年）に土佐藩脱藩後に薩摩藩に向かった。西郷隆盛たちと盟約を結んだのであろうが、そのきっかけとして樟脳交易があったと考えられる。また高知県西部の宿毛藩も樟脳の収益で外国船打払用の銃器を買い入れている。

６　マーシャントの語源

商人は英語で"merchant"だが、その語源であるラテン語で"marcatus"は商売の意味である。その類語としての"marketing"やフランス語の"marche"（市場）も商業をシンボライズした言葉である。また、ローマ神話の神マーキュリー"Mercury"も商業の神、水星のシンボルであり、ギリシャ神話のヘルメス"Hermes"と同一視される。ちなみにヘルメスをフランス語風に H の音抜きで発音するとエルメスと発音され、有名なブランド名にあたる。

また、フランス語の"mercy"(慈悲、ありがとう)も同じ語源で、相手の寛容・忍耐の姿勢に対して感謝する意味を込めてお礼の言葉として用いられている。

つまり、マーケット、マルシェ、マーシャント、マーキュリー、メルシー等は、語源が同じ言葉である。

７　桜井漆器の割賦販売

しまなみ海道の南端の愛媛県今治地方は、瀬戸内海の海運の要衝として交易や水軍や信仰の深い歴史が残る地域である。現在でも造船業やタオル産業や用船業などのビジネスが盛んでビジネス教育を重んじる気風がある。

この地で江戸時代に発達した漆器生産業は、瀬戸内海一帯で椀船と呼ばれる行商の船で漆塗りの食器を後払いで売り、次に寄ったときに代金を受け取る仕組みで交易をしていた。特に金融面において割賦販売方式を導入していた点が先進的であり、顧客本位に考えて商売をする発想が江戸時代に確立していたことが興味深い。

８　身の丈の単位

メートル法で度量衡が統一されて便利になったが、古くからある単位は人間の身の丈が基準になっている。例えば、１尺や１フィートは肘から先の腕の長さが基準で一間は人一人が寝たりくぐったりするのに丁度よい長さである。容積の１合や１カップは、人が飲み物を飲むのに丁度よい分量であり、米の１俵は人が１年間生きるために必要な量である。

情報処理における 10 進数と 2 進数の互換操作のように身の丈の単位が優先する場合もある。

９　「はたらく」と「あきない」

仕事は生活していくための収入を得る活動ではあるが、サービス経済化の進んだ現在では、自分の仕事に意義を見いだせなければ続けられない。和言葉の深い意味として、「はたらく」は他者を楽にしてあげることを、「あきない」は飽きずに働くことを大切にするようにとの思いが込められている。そして、その意識で仕事に取り組めば、向上心を保ち新たな成功や達成が果たせるのである。

補足資料Ｂ　Swing によるウインドウ操作用クラス：Window_Class3.java

```
package swing.study.OOP;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Container;
import java.awt.Font;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import javax.swing.JFrame;
import javax.swing.JLabel;

public class Window_Class3 {
   InputStreamReader kin
                        = new InputStreamReader(System.in);
   BufferedReader rdr = new BufferedReader(kin);

   private JFrame mainFrame;
   private JLabel label;
   private String namae;
   private String hyoujibun;
   private Color iro;
   private int wtop;
   private int wleft;
   private int wwidth;
   private int wheight;

   public String getNamae() {
      return namae;
   }

   public void setNamae(String namae) {
      this.namae = namae;
   }

   public String getHyoujibun() {
      return hyoujibun;
   }

   public void setHyoujibun(String hyoujibun) {
      this.hyoujibun = hyoujibun;
   }

   public void moji_hyouji() {
      label.setText(getHyoujibun());
   }

   public Color getIro() {
      return iro;
   }

   public void setIro(Color iro) {
      this.iro = iro;
   }

   public int getWtop() {
      return wtop;
   }
   public void setWtop(int wtop) {
      this.wtop = wtop;
   }

   public int getWleft() {
      return wleft;
   }

   public void setWleft(int wleft) {
      this.wleft = wleft;
   }

   public int getWwidth() {
      return wwidth;
   }

   public void setWwidth(int wwidth) {
      this.wwidth = wwidth;
   }

   public int getWheight() {
      return wheight;
   }

   public void setWheight(int wheight) {
      this.wheight = wheight;
   }

   public Window_Class3() {
      setNamae("ウインドウズインスタンス 1 号");
      setHyoujibun("インスタンス 1 号生成完了");
      setIro(Color.GREEN);
      setWtop(20);
      setWleft(50);
      setWwidth(300);
      setWheight(200);
      mainFrame = new JFrame(getNamae());
//
mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    mainFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
      mainFrame.setSize(getWwidth(),getWheight());
      mainFrame.setLocation(wleft, wtop);
      mainFrame.setLocationRelativeTo(null);

     // JFrame より ContentPane を取得
      Container contentPane = mainFrame.getContentPane();
      contentPane.setBackground(iro);

     // ラベルのインスタンスを生成
      label = new JLabel(getHyoujibun());
      label.setHorizontalAlignment(JLabel.CENTER);
      label.setFont(new Font("ＭＳ　ゴシック", Font.BOLD, 48));

     // ラベルを ContentPane に配置
      yout.CENTER);
     // ボタンのインスタンスを生成
     // JButton button = new JButton("SwingButton");
     // ボタンを ContentPane に配置
     // contentPane.add(button, BorderLayout.SOUTH);

      mainFrame.setVisible(true);
   }
```

```
    // コンストラクタメソッド
    public Window_Class3(String na) {
        setNamae(na);
        setHyoujibun(na + "-インスタンス生成完了");
        setIro(Color.YELLOW);
        setWtop(20);
        setWleft(400);
        setWwidth(300);
        setWheight(200);
        mainFrame = new JFrame(getNamae());
//      mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mainframe
              .setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        mainFrame.setSize(getWwidth(),getWheight());
        mainFrame.setLocation(getWleft(), getWtop());
        mainFrame.setLocationRelativeTo(null);

    // JFrame より ContentPane を取得
        Container contentPane = mainFrame.getContentPane();
        contentPane.setBackground(iro);

    // ラベルのインスタンスを生成
        label = new JLabel(getHyoujibun());
        label.setHorizontalAlignment(JLabel.CENTER);
        label.setFont(new Font("ＭＳ　ゴシック", Font.BOLD, 48));

    // ラベルを ContentPane に配置
        contentPane.add(label, BorderLayout.CENTER);

        // ボタンのインスタンスを生成
        // JButton button = new JButton("SwingButton");
        // ボタンを ContentPane に配置
        // contentPane.add(button, BorderLayout.SOUTH);
        mainFrame.setVisible(true);
    }

    public void irogime(int n) {
        setIro(Color.WHITE);
        if(n==0) {
            setIro(Color.BLACK);
        } else if (n==1){
            setIro(Color.BLUE);
        } else if (n==2){
            setIro(Color.RED);
        } else if (n==3){
            setIro(Color.MAGENTA);
        } else if (n==4){
            setIro(Color.GREEN);
        } else if (n==5){
            setIro(Color.CYAN);
        } else if (n==6){
            setIro(Color.YELLOW);
        } else if (n==7){
            setIro(Color.WHITE);
        } else {
            setIro(Color.WHITE);
        }
        this.mainFrame.getContentPane().setBackground(iro);
        this.mainFrame.setVisible(true);
    }

    public void irogime(int r,int g,int b) {
        if(r > 255) r = 255;
        if(g > 255) g = 255;
        if(b > 255) b = 255;
        if(r < 0)     r = 0;
        if(g < 0)       g = 0;
        if(b < 0)       b = 0;
        this.mainFrame.getContentPane()
                              .setBackground(new Color(r, g, b));
        this.mainFrame.setVisible(true);
    }
    public void henkei(int x,int y) {
        if(x < 0 || x > 800) x = 100;
        if(y < 0 || y > 600) y = 100;
        setWwidth(x);
        setWheight(y);
        this.mainFrame.setSize(getWwidth(),getWheight());
    }

    public void idou(int x, int y) {
        if(x < 0 || x > 1000) x = 0;
        if(y < 0 || y > 800)    y = 0;
        setWleft(x);
        setWtop(y);
          mainFrame.setLocation(getWleft(), getWtop());
    }

    public void jyouhou_hyouji(String stc) {
        if(stc.length() > 20) stc = stc.substring(0,20);
        setHyoujibun(stc);
        label.setText(getHyoujibun());
    }

    public void tojiru() {
//      System.exit(0);
        this.mainFrame.setVisible(false);
    }

    public void kiin_aizu() {
        System.out.print("エンターキーを押せば次に進みます。");
        try{
            String str1 = rdr.readLine();
        }catch(Exception e){
            String str1 = "";
        }
    }

    public void yasumu(int i) {
        try{
            Thread.sleep(i * 1000);
        } catch(InterruptedException e)       {
            System.out.println("スリープ操作失敗。");
        }
    }

    public void subete_tojiru() {
        System.out.println("ウインドウオブジェクトを廃棄し、
                              このプログラムを終了します。");
        kiin_aizu();
        System.exit(0);
    }
}
```