香川大学経済論叢
第72巻 第2号 1999年9月 283-307

情報処理教育の新しい発展

—JAVAは情報処理教育をどのように変えるか—

今井慈郎* 宮武明義**
井面仁志* 石川浩*

1．これまでの情報処理教育とその問題点

最近，計算機，特にパソコンは多くのユーザにとって文房具であり，ワープロ・表計算・データベースなどのアプリケーション（以下，アプリと略記）が動作すればことが足れりとされる傾向にある。パソコンを主体とする情報処理教育においても，これらのアプリを使いこなすことに主眼をおいた「情報リテラシー教育」が一般的になりつつある。しかし，情報リテラシー教育は情報処理教育の1つの柱ではあるが，プログラミング教育の重要性が低減したわけではなく，情報処理教育の大切な支柱の1つであることに変わりはない。例えば，Excelを用いた情報処理教育でも操作を中心とした入門教育が一段落すれば，「機能拡張・定型操作実現のためのマクロ記述」などの標題で発展教育を行うケースをよく見かける。前者は，言わばExcelを用いた情報リテラシー教育であり，後者は全員に必要とは言えないまでも，プログラミング教育の1つの実現例であることは明らかで，これなどExcelを中心に据えた情報処理教育とみなすことができる。

何故，情報リテラシー教育という表現を冠した情報処理教育が出現したのだろうか。それ以前の従然とした情報処理教育が，ともすればプログラミング教

＊ 香川大学工学部 信頼性情報システム工学科
＊＊ 詫間電波工業高等専門学校 情報工学科

育偏重であったことは否めない。しかし，計算機人口が増加し，それに伴った情報処理教育へのニーズがプログラミング教育を常に重視したわけではない。少なくともある時期のプログラミング教育には教育効率の悪い，プログラミングを好きにするより嫌いにする反面教育的側面が存在したことが指摘されている。これに対し「まず計算機を利用するには代表的なアプリを使いこなすことだ」との命題に即して識字教育（「読み書き算盤」式の教育）の計算機版：すなわち情報リテラシー教育が新たに提唱された。情報リテラシー教育では，計算機を有効活用できる能力の養成を第一義的に認めた教育であり，誰でも使える計算機利用技術を目標に設定した「習うより慣れろ型」教育と言えるだろう。「習うより慣れろ型」教育とは言うものの，この要素は，プログラミング教育をはじめ多くの教育には大なり小なり存在する方法論であり，情報リテラシー教育を不当に軽視する意図はない。

その後，「情報リテラシー教育は重要な情報処理教育である」とする，文部省や通産省などの大合唱が大きな推進力になって，ほとんどの小学校にはパソコンルームなる教室が作られ，課外活動から正規のカリキュラムへと情報処理教育（特に入門教育としての情報リテラシー教育）が浸透してきた。これに押し上げられる形で以前なら，「習うより盗め」とでも言うか，徒弟制度的教育が中心だった大学での情報処理教育も一変せざる得ない状況となった。しかし，情報処理教育，特に一方の柱であるプログラミング教育が今後どうなっていくのだろうか。担当者においても不安材料を無視できない。

プログラミング自体は好き嫌いがはっきりした性質の技術である。Knuth の著した有名な “The Art of Programming” など，タイトルからして，「プログラミングは技術であると共に芸術だ」と看破しており，プログラミングの面白さと難解さとを同時に知らしめる好著であるとの評判が高い。プログラミングが好きになるのはどのような人であり，どのような人が嫌いになるのか，という究極の疑問に答えられる人など少ないものの，どこかに基準があり，それを如何にクリアしプログラミングをものにするかしないかが分水嶺となっているようだ。事実，筆者もプログラミングは苦手であった。今でも，定型的あるい

は規模の小さなプログラミングはなんとかなるものの，センスを要求される場合などはお手上げとなる。万人向けのプログラミング教育を簡単に実現するのは相当に難事業と言える。「習うより慣れろ」，「読書百遍，義おのずから通ず」などという少し古いタイプのプログラミング教育では効率に問題があるかもしれないし，教育効果も疑問視されるだろう。遠い過去においては，それなりに成果をあげてきたことも事実であるが，もはや時代遅れかもしれない。

どうすれば楽にプログラミング技術を習得できるか，がプログラミング教育を実施する上でも重要なテーマになっている。計算機と同様，プログラミング（言語）もユーザの好みで選択し，プログラミング技術など必要としないユーザには情報リテラシー教育だけで十分という時代がやってきている。しかし，だからこそ差別化を図る意味でもプログラミング能力を養成することがこれまで以上に重要だとも言える。プログラミング技術をどのレベルまで身に付けることができるかは，習得者自身の意欲と相性に大きく左右されるものの，重要な要素である。

習得における一般的原則として，「煽ててその気にさせる」という技法も無視できない。例えば，気の合った仲間何人かでプログラミング技術を競い，相互に成果を示し合うというのは確かに励みになる。周囲を「あっ」と言わせるようなプログラムを作ることができれば，「自分を誉めて」やりたくなる。これは何かを達成する上では，重要なモーティベーションであり，「自分を誉めて」やりたくなるようなインセンティブがなければプログラミングなどという孤独な作業を長期間にわたって続けることは難しいであろう。プログラミング教育では，習得者一人ひとりに「もしかすると，自分にはプログラミングの才能があるのでは...」と思わせれば大成功である。逆に，「プログラミングには向いていない」と諦めさせたのでは失敗と言うしかない。最初から完成されたプログラムを作成できる人間など少ないから，まずは「もう一度やってみよう」と思わせる状況を作り出すことが重要である。プログラミング教育の正否の鍵となるだろう。その意味では，凝り性の性格よりも己惚れの強い性格の方がプログラミングに適しているかもしれない。どのレベルまで到達すればプログラミン

グに対する及第点が取れるか，という深淵なる疑問には答えられないものの，「プログラミングを嫌いにならなければ及第点!!」と考えても大きな間違いではないだろう。嫌いにさえならなければ，成功への偉大な一歩が踏み出せるわけだから。

プログラミング教育談義を精神論だけで済ませることができるかと言えばやはり叱責を免れない。計算機という「効率至上主義の権化」のような存在に対して，精神論だけで立ち向かうことは不可能だ。効率を重視する以上，プログラミング言語の選択という要素は避けて通れない。相性を無視することはできないが，なじめない（理解に苦しむ，好きになれない）記述を要求されればそれだけで「プログラミングそれ自体が嫌い」になる。初恋と同じと言えば不謹慎かもしれないが，最初の印象が良く，馴染んだプログラミング言語あるいはプログラミングスタイルはいつまも忘れ難いものだ。プログラミング言語は何が最適か，という議論が幾度も繰り返されるのも故無しとしない。

最初に馴染むべきプログラミング言語は，可能ならばその後も長期間使用できるものが望ましい。基礎教育ではBASIC言語を学び，その後の応用教育ではC言語を学ぶ，というカリキュラム構成では不満が生じるかもしれない。また，それ自体には特段の問題があるとは思わないが，一貫性のなさを指摘する向きもあることは事実だ。シームレスな（縫い目のない，朝令暮改にならない）プログラミング教育であることを理想とするあまりの指摘であろう。筆者自身の経験は朝令暮改型の典型で，プログラミングの経歴は多岐亡羊と言える。大学で受けた情報処理教育自体がちょうど黎明期の終わりであったという理由から，学生時代の授業ではAlgol言語，実験ではアセンブリ言語とPL/I言語，卒業研究ではマイクロプログラム（これは特殊過ぎるが）と機械語，卒業後しばらくはPASCAL言語，仕事ではC言語…。シームレスなどは決して言えない状況にあった。

その昔，情報処理教育と言えばFORTRANが当然だ，と誰もが信じて疑わなかったし，仕事に使えなくては役に立たないとする効率重視主義は常に支配的だったと言える。計算機と言えば汎用機という時代であれば無理もない。最近，

逃げられない理由からFORTRANを使用したが，よくできたプログラミング言語だったことを再認識した。文字列操作やファイル操作などはあまり得意ではないものの，数値計算に特化した記述能力は特筆すべき機能であり，豊富なライブラリ群を有するその風格はさながら世界文化遺産とも言える。科学技術計算に対しては今後も現役であり続ける存在だろう。しかし，環境の変化は確実に進み，FORTRANからプログラミング言語の主役の座を奪いつつある。今後，Objective-FORTRANとかFORTRAN＋＋などの新種が登場すれば話は別だが。ワークステーションやパソコンの登場，ダウンサイジング，そしてWindowsの流行とまさに日替わり定食のメニューよろしく変わる状況下では，ステディな存在自体が珍しく，まさに「転石苔むさず」と言えるだろう。

一方，情報屋（情報科学関係者）の間では，最近までプログラミング環境はUNIXで，プログラミング言語はC言語で，という考え方が定番であった。「本物のプログラマはPascalを使わない」などという論文が掲載されていた時代の風潮のなごりだったのかもしれない。一般に，プログラミング環境を考える時，OSの存在は無視できない。特に，UNIXの登場と流行は多くのユーザに「プログラミング環境はかくあるべき」と認識を新たにさせ，OSの提供する機能がプログラミング環境を大きく左右することを明確に示した。情報屋の支持もあり，多くの教育機関ではUNIXこそプログラミング教育の最適な環境と看做し，高価なUNIX環境をこぞって導入する動きが見られた。UNIX上で多くのプログラミング言語が利用できるのも，特に本国アメリカ合衆国におけるこのような背景があったからだと言える。UNIXマシンはミニコンピュータから比較的低価格のワークステーションへとその価格レンジを広げてきたが，UNIXマシンを導入するのはやはり経費の面で無理もあり，UNIXマシンを導入して「理想的なプログラミング環境」を活用できたのは限られた教育機関だけであった。その意味でもUNIXは(少し皮肉な言い方だが)「プロのためのプログラミング環境」であり続けた。

UNIXの登場とは非同期ながら，プログラミング教育に大きな影響があったものに，8ビットCPUを搭載したマイクロコンピュータがある。マイクロコン

ピュータは価格性能比を売り物に多くのユーザ，とりわけ個人ユーザに好感をもって受け入れられ，時間をかけながら徐々にプログラミング環境の主役の座に駆け上った。UNIX より規模を大幅に縮小し，限られた機能のみの提供に専念することでコンパクトな OS となった CP/M や MS-DOS。これらは OS と呼ぶには少し気の引ける存在だが，個人利用の軽微なコンピュータ（パーソナルコンピュータというネーミング自体がヒット商品）との相性は，まさに水を得た魚の如しであった。後日，マイクロソフト帝国皇帝となるビル・ゲイツが最初のヒット商品である BASIC インタープリタを世に問うたのもこのような OS 環境下で動作するソフトウェアとしてであった。BASIC というプログラミング言語およびその処理系は確かに良くできていたが，多くのユーザの心を捉えたというより，ほんど唯一の選択肢であったというのが実状であろう。当時のパソコン上に UNIX が走るなど，夢想だにしなかった。しかし，一度ユーザの心を捉えた BASIC はその後，N88-BASIC や F-BASIC など多くの亜流を発生させ，パソコンの ROM に搭載され，パソコンすなわち BASIC マシンといった印象を与え続けた。その意味では多くのアマチュア・プログラマにとって，BASIC が最初のプログラミング言語（環境）となった可能性は高い。

16 ビット CPU 搭載のパソコンが登場し，パソコン OS すなわち MS-DOS という構図が定着すると，堰を切ったように様々なプログラミング言語が登場し，市場にあふれた。もちろん，汎用機時代から利用され古典派に属す FORTRAN や COBOL を始め，LISP や Pascal など伝統的なプログラミング言語がパソコンで利用可能になった。例えば，UCSD-Pascal などプログラミング環境と OS とが一体となったプログラミングのための OS 環境という新しいコンセプトが登場した。国産の Prolog-KABA など，秀逸なプログラミング言語（環境）がパソコンで利用できたが，これらはどれも一世を風靡し，教育機関でも盛んに利用された言語や環境であった。大学をはじめとする高等教育機関でのプログラミング教育が汎用機からワークステーションを飛び越えてパソコンに大きくシフトして行ったのもこの時代だった。当時は高等学校や中学校で情報処理教育を行うなど遠い未来のことと思われていた。

プログラミング教育にとって，エポックメーキングな出来事としてパソコン版C言語の登場が挙げられる。「C言語と言えばUNIX，UNIXと言えばC言語」という構図から一歩踏み出したパソコン版C言語は鮮烈な印象を与えた。Lattice-CやMS-C，Turbo-Cなどが相次いで市販され，C言語にあらざればプログラミング言語にあらず，といった一大ブームを巻き起した。その間，MS-DOSはプログラミング環境としても不動の地位を獲得していった。多くの教育機関がMS-DOSマシンとしてNEC-PC（PC-9801シリーズ）を競って導入したのもこの頃である。プログラミング教育はC言語とパソコン環境とで実現するというのが多くの教育機関が出した選択結果である。もちろん，Turbo-PascalやOS-9/68000などといった注目すべき非主流派も存在したが。

Macintoshを文房具とばかり思っていた頑迷な情報屋（筆者もその一人）にとって，昨今のWindowsプログラミングの流行には不明を恥じるばかりだ。どうしてMacが登場した時，プログラミング環境と位置づけなかったか今となっては説明不能と言える。Xerox PARCが生んだ伝説のALTOワークステーションを真似ただけだと思われていが，その先進性に恐れをなしていたのかもしれない。要するに，プログラミング教育ではUNIX環境に憧れつつもMS-DOS環境で我慢し，C言語でその雰囲気を疑似体験する，というのがWindows 3.1登場までの状況であった。膨大な努力をしてグラフィックスをプログラミング教材に採用するなど，個別の対応はいろいろあったものの，スクリーンエディタとコマンドラインベースでのプログラミングスタイルがしばらく続き，プログラミング教育は安定期を迎えていた。それが大きく揺れ動き始めたのはWindows 95の登場を契機にしてからだ。何が大きく変わったかと言えば，情報リテラシー教育が情報処理教育の大きな柱になっていったことである。既にWindows 3.1時代に火種が生じたワープロ，表計算あるいはデータベースといったいわゆるオフィスソフトの利用方法を総合的（!?）に教育する情報リテラシー教育があちこちでスタートし，プログラミング教育主体だった情報処理教育は一変した。そして，プログラミング教育の百家争鳴時代が始まった。

2．Windows の登場と情報処理教育における功罪

Mac が提供したマルチウィンドウ環境に対しては比較的冷淡であり，プログラミング環境と捉えるよりも，多くのユーザが電子文房具的な印象を強く持った傾向にある。しかし，マイクロソフトがほとんど類似のコンセプトで投入した Windows PC は世界中で全く異なった反響を生んだ。オープンな IBM-PC クローンの世界市場でのシェア占有率を背景に，Windows 3.1 で既に外堀を埋められていた日本独自の PC 市場さえも完全に Windows 95 に席捲され，大学，高専はもとより全国の小中高校へ Windows PC が急速に導入されていった。情報処理教育も大きな節目を迎え，「情報リテラシー教育は新しい教養教育の基本である」という新しい見解がもて囃された。

「21 世紀はソフトウェア技術者が大幅に不足する」とのお役人発言を鵜呑みにして，情報リテラシー教育とプログラミング教育とを明確に区別しないままに，流行の Windows PC を情報処理教育の主役に抜擢したのは教育機関の功罪と言えるだろう。確かに Windows PC は情報リテラシー教育には適している。Mac もそうであったように電子文房具的要素を兼ね備え，ワープロ，表計算そしてデータベースなど「三種の神器」が比較的容易に利用できる環境であった。一時期マイクロソフトがバンドル販売を拒否しようとしたオフィスソフトが初期インストールされたケースが多く，ネットワーク接続への対応も比較的簡単になった Windows PC は，導入したその時から電子文房具として，インターネットのクライアントマシンとして，それなりに重宝な存在である。事務機器としても十分実践的であり，決してホビースト向けマシンではない機能を具備している。

しかし，Windows 環境がプログラミングに適しているか，と考えていくと必ずしも納得できる点ばかりではない。プログラミングは基本が大切だとよく言われる。初めの印象で好きにも嫌いにもなるからだ。誰だって最初から難しい話を聞かされるのはいやに決まっている。C 言語のプログラミングに関する多くの教科書は，標準出力に“Hello，World!”を出力させる有名な例題から始ま

る。一方，初期の頃の Windows 環境で同様のことを実現しようとして，呪文のような制御文があまりに多くて始めからウンザリしてしまったユーザも多いだろう。実は筆者も呪いのようなコードに恐れをなして戦う前に敵前逃亡してしまった一人である。少なくともその時点では，Windows 環境は決してプログラミングに適した環境ではないと信じ込んでしまった。冷静になった今でも，Windows 環境はコンパイラ形式のプログラミング言語（C言語もその1つ）には向いていないと思われる。これはオブジェクト指向プログラミング言語全般に当てはまる事情だが，ダイナミックリンクという手法で回避しているとはいえ，環境（すなわち OS）が複雑になればなるほど，プログラミングの基本操作である入出力すら簡単に操作できない仕組みとなり，OS 呼出しが異様に肥大して化け物のような実行コードになる傾向にある。速度よりも環境依存でスマートに実行させようとすれば，インタープリタ形式で実現するのが効率などの点で早道と言えよう。

Windows 環境でのプログラミングなら，Visual BASIC は最適的なプログラミング言語（環境）の1つであると思われる。恐らくマイクロソフトは Visual BASIC を Windows 環境でのシェル（正確にはシェルスクリプト記述言語あるいはその処理系）として位置付けようとしているのではないか。しかし，プログラミング教育にすんなりと受け入れられるにはいくつか問題もあった。コンパイラ言語のように実行バイナリ（exe ファイル）を生成しながらも，専用のダイナミックライブラリがないと動作しない点など，環境依存のインタープリタ型言語の亜種であることが伺われるし，マイクロソフトの自作自演的印象から教育上何かフェアでないとする考え方も多かった。Visual BASIC と同様のコンセプトに Tcl/Tk というプログラミング言語がある。Windows 環境でも動作するが，基本的に UNIX 環境でもそのソースコードをそのまま実行できる点など環境非依存性があり，言語および処理系の魅力となっている。後述する JAVA にとっても，ある面では先駆者的存在と言えるだろう。もちろん，今だに現役のプログラミング言語である。

Windows 環境においてプログラミング教育を始める限り，C言語（C＋＋言

語を含む）を教育用として最適なプログラミング言語であると見做すことは難しい，というのが筆者の考えの根底にある。C言語はこれまでコンパイラ型言語の典型と思われていたため，Windows環境への対応は決してスムーズではなかった。FORTRANなどからプログラミング言語の主役の座を引き継いだ感のあるC言語にとって，Windows環境への対応のまずさは，プログラミング教育の混迷を招いたといっても過言ではない。対応のまずさが，多くの教育機関において様々な模索を試みさせたとも言える。最適な実行を保証しようとすればダイナミックライブラリを充実させる必要があり，その分だけ環境との親和性が強く要求され，言語仕様にも影響が出ざるを得ない。入門でつまづくとプログラミングは順調には進まない。プログラミング教育はC言語で行うものという雰囲気に慣れていた教育機関は問題に直面してしまった。

少し整理してみたい。Windows PC登場以前には，C言語を中心としたプログラミング教育が一応確立され，高価なUNIXマシン導入を前提にしないでも，十分な台数のMS-DOSマシンと軽快なDOS版Cコンパイラとを用いて一人一台が基本のプログラミング入門教育を実現できた。MS-DOSマシンは現在のWindows PCと比較してかなり性能が低かったことは事実だ。しかし，マシンのライフサイクルは結構長く，要求される機能もそれなりに満足されるものであったため，結果としてコスト性能比が相対的に高くなった。このような一般の教育機関での入門教育を終えた後，プロを目指してUNIX環境に移行しても（上位互換であるため）プログラミング環境のギャップは比較的少なく，プログラミング能力を環境の広がりに対して連続的に発展させていくことが可能であった。

次に，Windows PCの登場以降のプログラミング教育事情を考察してみたい。従来かなり熟練した技術が必要だったGUIプログラミングが飛躍的に簡単となり，専門家はもちろんのこと，プログラミングの素人でもマルチウィンドウをベースにしたGUIプログラミングに挑戦したいという雰囲気が広がった。しかし，ここで問題が1つ発生する。これまでいろいろと努力してきた熟練プログラマにとって，GUIプログラミングが容易なWindows環境は願ってもな

い「プロメテウスの火」となった。しかし，これからプログラミングを始めようとする初心者にとっては，Windows 環境でのプログラミングからスタートすることは，意味不明な約束事ばかりで快適なプログラミング環境を楽しむというよりもプログラミングを苦痛に感じることが多い。将来性という希望があるとしても現状不安や自信喪失などが一杯詰まった「パンドラの箱」と言うべき存在となった。

MS-DOS マシン以来の PC ユーザと，Windows PC が販売され始めた最近の PC ユーザとは世代の違い以上に次のような違いを感じるだろう。前者は高価でかつ使い難い MS-DOS マシンを利用した経験があり，ネットワークに IP 接続しようなどと思えば，NIC のボードと専用ドライバを購入するだけでも，10 万円以上必要となる時代のユーザである。昨今の低価格な Windows PC には技術進歩と標準規格の恩恵を感じている。一方，後者はコストパフォーマンスが良いと太鼓判を押され，初めて Windows PC を買い，フリーウェアをいろいろインストールして利用している。しかし，何がそんなに嬉しいのか分からない内に結局宝の持ち腐れ状態になっている。このような気持ちの相違はプログラミングへの取り組みにも反映する。MS-DOS マシンでのプログラミングはできることが少ない代わりに，挑戦した見返りも十分あった。Windows PC になって便利な点と不都合な点を峻別できるスキルを積んだことになる。Windows PC で初めてプログラミングに挑戦する場合，何が便利なのか分からず，しかも歴史的経緯が実感できないため，約束事の多さ，無味乾燥さに，プログラミングの面白さを満喫できるところまで到達できる割合は少なくなる。確かに Windows PC を使用しなければほとんど絶望的に難しいプログラミングテーマであっても，重要なポイントまで隠蔽された環境では，結果としてプログラミングそのものの価値をなかなか把握できない。初心者のプログラマにとって Windows 環境は見方なのか敵か分からなくなってしまう。

誰であれ，努力をしてもそれが報われるものなら厭わないものである。若い頃の苦労は買ってでもしろ，と言われるが，それは後年報われることが前提だ。しかし，Windows 環境だけにしか通用しないプログラミング技術は苦労して

まで習得するべきものか，大いに疑問が残る。しかし，逆にUNIX環境のみに特化してプログラミング教育を実施するには，英断というより暴虎馮河的蛮勇が必要となり，教育機関では一様に逡巡を余儀なくされてきた。UNIX環境でもWindows環境でも利用できることを特徴とし，それなりに期待されたTcl/Tkというプログラミング言語がある。GUIベースのプログラミングが可能なインタープリタ型言語であり，処理系がフリーウェアである点も魅力的だった。しかし，残念なことに市場は冷淡であり，大手ソフトウェアベンダからの支持はほとんどなく，テキストも決して豊富とは言えなかった。これでは教育機関も対応できない。そのような混沌とした状況下に突如彗星の如く登場したのがJAVAである。

3．JAVAがもたらす新しいプログラミング教育

JAVAおよびその処理系はUNIXマシンの有力ベンダであるサンマイクロシステムズのソフトウェアプロダクトである。戦略商品でありながら処理系（JDKなど）を教育機関や個人に対しては，フリーウェアと同様な使用条件で提供し，多くのプログラマの支持を得ている。特徴として，

a）GUIベースのプログラミングが可能なオブジェクト指向言語として新たに設計された点

b）UNIX環境およびWindows環境で動作するマルチプラットフォーム対応である点

c）ネットワークコンピューティング機能を装備している点

など，混沌としていたプログラミング教育事情に新風を巻き起こすに足る，魅力に溢れた機能が提供されている。インターネット時代のプログラミング教育では，GUIベースのプログラミングと同等か，あるいはそれ以上に，ネットワークプログラミングが容易に実現できることが重要となる。確かにプログラミング初心者にとっては，両者とも，簡単には取り組めない程の深い内容のテーマである。しかし，プログラミング言語を選択する際に，重要な要因となる発展性や記述能力の高さについてみれば，JAVAの記述能力は現時点で第一級の存

在である。実際，JAVA で記述された IBM 製の Desk Top On Call などのリモートデスクトップ制御ソフトウェアは複数学生の Windows PC を教師の Linux マシン 1 台で巧みに管理できる。同様のコンセプトを持ち，フリーウェアとして利用できる ORL の VNC: Virtual Network Computing などもやはり JAVA で記述されている。JAVA を使用したソフトウェアが今後ますます市場に登場するだろう。このような状況は JAVA の記述能力の高さの一端を示している。一太郎で有名なジャストシステムも JAVA を利用したアプリの開発を発表している。要するに本物のプログラマ達が JAVA で優秀なソフトウェアプロダクトを作成しているという事実が教育機関にも大きな影響を及ぼしつつある。

話題をプログラミング教育以外に転じよう。ホームページを作成するというテーマは情報処理入門教育としては手頃である。WYSIWYG 型ではないマークアップ言語仕様の HTML を理解させながらホームページを作成させるという状況下で，1）エディタの利用，2）ftp によるファイル転送（場合によって telnet の利用），そして，3）WWW ブラウザの利用などを総合的に教育できる。情報リテラシー教育の主要な題材として注目を集めているが，同時にプログラム構造を教える布石にも利用可能である。そこで，ホームページ作成の発展形として，JAVA プログラミングを導入する。簡単な例題から入れば，スムーズに Web プログラミングへと移行することも可能となる。平易な Applet 記述などから（多くの場合，例示を真似るだけかもしれないが），ホームページ作成という入門的題材を無理なく発展させることができる。教師側からすれば，竹を木に継いだようなコジツケを行うことなく，GUI プログラミングやネットワークプログラミング（上記の内容であれば Web プログラミングというべきか）の面白さを具体的に示すことができる。プログラミングそのものに興味がない学生に「プログラミング有りき」からスタートするのは，やはり無理があろう。一方 JAVA であればプログラミング対象の幅の広さが効果的に活用できる。

JAVA を利用すれば，Windows 環境上で，簡単に GUI プログラミングやネットワークプログラミングが実現できる。JAVA の処理系である JDK がマ

ルチプラットフォーム対応であるため，Windows 環境でも UNIX 環境と全く同様な手順でプログラミングができる。すなわち，JAVA を用いたプログラミングに関しては全く異なった2大 OS 環境を区別することなく統一的に扱うことが可能になる。この点でも，Windows 環境を採用すべきか，UNIX 環境で教育すべきかで，混迷していたプログラミング教育に救世主が現れたことが理解できよう。

Windows 環境下でのプログラミングだけに話題を限定すれば，Visual BASIC の利便性は衆目の一致するところである。Excel を始め様々なマイクロソフト・アプリにおいて利用可能なマクロ言語である VBA への平行移動的応用においても違和感を全く感じさせない。しかし，コマンドベースの(CUI ベースの)プログラミングから GUI プログラミングまで，Web プログラミングから真のネットワークプログラミングまで，幅広い対象を持ち，フリーウェアとして利用可能な処理系 JDK を擁する JAVA の魅力は Windows 環境でも Visual BASIC とほとんど互角の評価を得つつある。サンマイクロの OS 環境はもちろんのこと，Linux や FreeBSD など PC-UNIX の隆盛が追い風になっていることも事実だ。JDK の動作環境としては，好みのシェルが利用でき，真のマルチタスク環境を堪能できるなど UNIX マシンが有利であろう。一方，Windows PC 上でも JDK はきびきびと動き，バージョンアップのレスポンスが早く，他のプログラミング環境と比較しても引けをとらない。なにより，雑誌などの折込み CD-ROM でも容易に入手でき，個人で Windows PC を所有するユーザはほとんど無料で処理系をインストールできる。情報リテラシー教育にも使用できるため，Windows PC 上での JAVA プログラミングはこれからのプログラミング教育の推進役として熱い期待が注がれている。本稿では言及しないが，Linux などのフリー PC-UNIX を活用し，UNIX マシンをベースとしたプログラミング教育を再構築しようとする動きもあり，JAVA の動向そのものが注目を集めている。

しかし，JAVA をプログラミング教育の主役に迎えて何一つ問題がないかと言えば，それは言い過ぎかもしれない。残念ながら，オブジェクト指向プログ

ラミングをプログラミング教育のスタートポイントするには，閾値の高さが初心者にとってけっして楽ではない状況だ。特に，メソッド，プロパティ，クラス...などと概念的には一貫性があるものの，ユーザに膨大な記憶力を要求するプログラミングスタイルは，類推に慣れていない初心者にとって，プログラミングを必要以上にとっつき難くしているだろう。実際，少し慣れたとしてもリファレンスマニュアルを手元から離す訳にはいかない。プログラミング経験を積むことで能力が向上するが，その指標となるのが，デバッグの速さやマニュアルに依存したプログラミングからの脱却である。実はユーザ自身が以前作ったプログラムをマニュアル代わりに参照することが増え，参考書への依存が減ることになった結果と言える場合が少なくない。初めての表現を試すときにはやはりデバッグ用のチェックが必要になる。しかし，「この表現はどこかで一度書いたことがあるぞ」となると安心してプロパティやメソッドを同様な形式で使用することができる。結果としてプログラミングの所要時間が減り，デバッグに要する時間も低減できる。その意味では，JAVA を含めてオブジェクト指向プログラミングは，便利さよりも入門時の閾値の高さが留意点だ。オブジェクト指向プログラミングと言えば，SmallTalk という返事が返ってくる向きもあるが，その完成度もさることながら，統合環境という優秀なデバッグ環境を提供していたことが要因かもしれない。この点は留意すべきだろう。

当然ながら，プログラミング教育の効果をあげるためには，最低限のデバッグ環境を用意しておく必要がある。UNIX 環境にC言語という組み合わせで，プログラミング教育を実施したいという多く教育機関にとってGNU プロジェクトから提供される優秀なソフトウェアツール，特にデバッガなどの存在が大きいと思われる。コンパイラ型のプログラミング言語の場合，ソースコードデバッガなどの存在が重要視されるのも故無しとしない。一方，インタープリタ型のプログラミング言語では統合環境が用意される傾向にある。JAVA も JDK だけでプログラミングに挑戦するより，比較的低価格で機能豊富な統合開発環境を使用する傾向が強くなってきた。実績のある統合開発環境はいくつか市販されており，JDK に採用された高速 JIT コンパイラを開発するなど JAVA の

評価向上に貢献のあったSymantecのVisual Cafeなどが著名であろう。香川大学情報処理センターにも同製品が導入されている。これは統合環境であるRAD (Rapid Application Development) がかなり完成された形で提供されている。個人的にはRAD環境が，JAVAプログラミング教育にとって不可欠だ，とは考えていない。しかし，Windows環境ではRADの老舗であるVisual BASICなどと比較すると，JAVAのプログラミング機能自体に不安を感じるほどなら，RAD環境の導入を適宜考慮する必要もあろう。

ともあれ，混迷していたプログラミング教育がJAVAの登場とプログラミング環境の充実により，再び収束し始めたように思われる。UNIX環境（より簡易なMS-DOS環境も含めて）とC言語を中心に据えたプログラミング教育が盛んに実施されていた頃には，それを支える存在として，UNIX関連の豊富なテキストがあった。関連するテキストが充実することは目に見えない形でプログラミング言語の選択に大きな影響力を持つ。JAVAがプログラミング教育の中心的存在となる条件の１つとして，どのような形でテキストが充実してくるのかという点に注目している。

マイクロソフトがソースコード非公開路線にあるのに対して，優秀なソースコードを積極的に公開していこうとするオープンソース陣営では，Visual C/C++あるいはVisual BASICというWindows環境でのプログラミング言語の使用を避け，Windows環境およびUNIX環境での使用を前提に，JAVAをプログラミング言語の中心に据えていく傾向が伺える。ここで，オープンソースという新しいソフトウェアの開発手法（共有手法でもある）に注目したい。オープンソースという旗印のもと，多くの優秀なプログラマがボランタリで同期したり，あるいは非同期でプログラミング・プロジェクトに参加し，プログラミング能力を競いながら自慢の腕を振るう状況は，ワールドワイドなプログラミング教育環境の登場と看做すことも可能であろう。オープンソース時代が到来したことで，インターネットなどのメディアを介して優秀なプログラミングに接することができる。UNIX (特にPC-UNIX) 環境とWindows環境という互いに独立した環境をリンクするJAVAの役割はますます増加すると思わ

れる。インターネット社会の新しい共通プログラミング言語がJAVAを中心に発展していくことで，プログラミング初心者にとってもJAVAは必須アイテムとなる。教育機関でもプログラミング教育にどのようなプログラミング言語を選択しようかと悩む必要がなくなるだろう。

4．JAVAプログラミングの例示

これまで述べてきたJAVAの特徴を活かしたプログラミング教育を実施する上で，題材の準備は最初の問題であり，避けて通ることはできない。そこで，本節では，具体的な例題を示すことでJAVAプログラミングを用いた情報処理教育の考え方・捉え方を議論したい。教育効果の充実を図るためにもどのような題材を準備するかが重要なポイントとなろう。例えば，簡単な電卓プログラムや高々2次方程式解法プログラムなどはサイズの点でも内容の点でも全体を容易に俯瞰でき，JAVAプログラミングの概要を簡単に理解することできる絶好の題材となる。図4－1に簡単な電卓プログラムの，図4－2に高々2次方程式解法プログラムの，それぞれ実行イメージを示す。また，付録にプロトタイプのソースリスト（リスト1：簡単な電卓プログラムおよびリスト2：高々2次方程式解法プログラム）を掲載する。これは前者がWebプログラミングの後者がGUIプログラミングの例題となっている。一方，少しプログラミング技術の習得が進んだ後の題材として次の2題を示す。

図4－1　簡単な電卓プログラムの実行イメージ

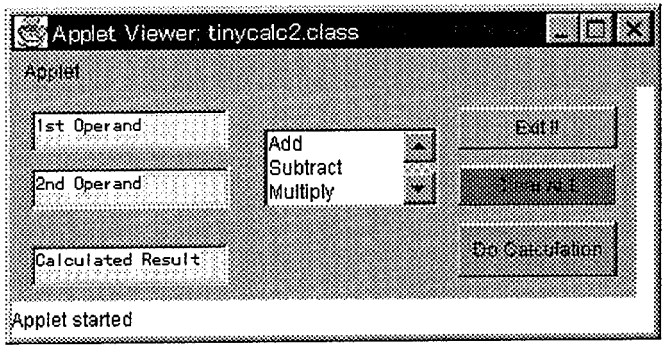
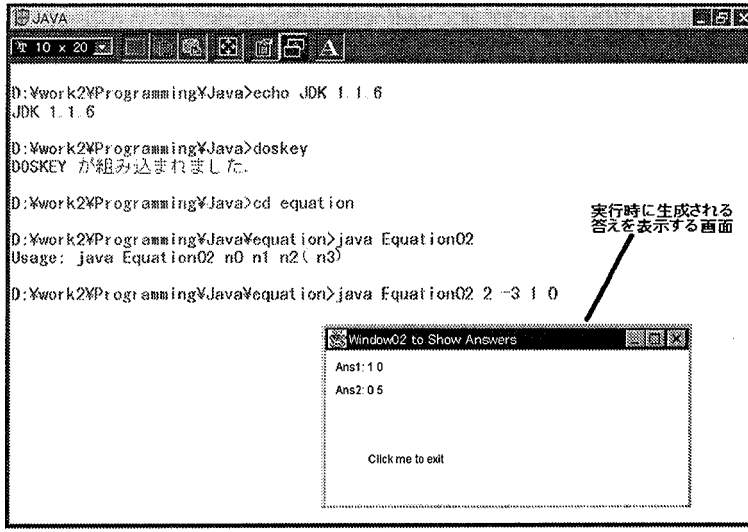


図４－２　高々２次方程式解法プログラムの実行イメージ



システムプログラミングの一例として，簡単なシェルを作成する課題を用意したい。シェルはコマンドインタープリタであり，OS の一部と看做すことができる。通常，入力行をコマンドとして解析し，いくつかのオプション処理を行って，コマンドで要求された処理を実行させる。全体をループで回すことで，終了コマンドを受理するまで，以上の処理を繰り返す。オプション処理の対象としては，コマンド履歴機能やコマンド入力補完機能などが考えられる。プログラミング教育の題材として採用する上でのアイデアとして，

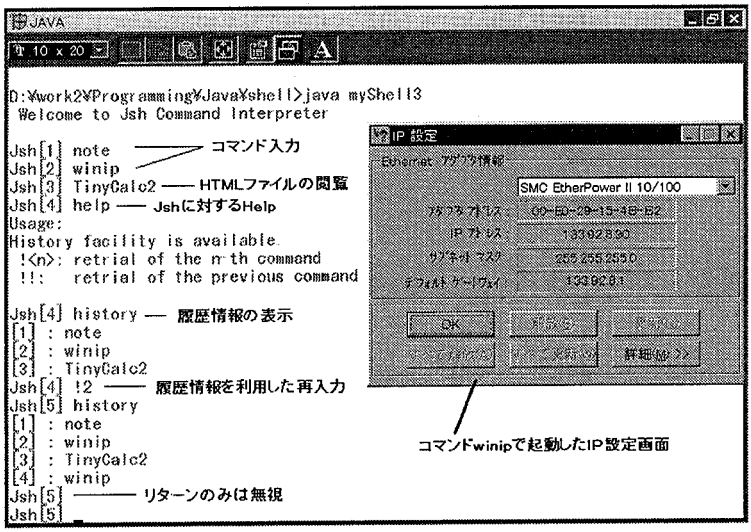
a）簡単なループ構造のプログラムで，コマンドインタープリタが実現できる点

b）比較的サイズの小さなプログラムでも例題のための例題（Toy Programming）ではない点

を具体的に示すことにある。シェルの構造を理解させ，各自で発展させることにより，ユーザ自身の使い方にフィットした（ユーザ親和性の高い）ユーザインターフェースを構成させることに力点を置いている。図４－３に簡単なシェ

ルの実行イメージを示す。参考のため，プロトタイプのソースリストを付録に掲載する(リスト３：簡単なシェルのプログラム，を参照)。プログラム自体の構造は単純であるが，プログラム演習としての発展性があり，システムプログラミングを通じて計算機システム自体を理解する上でも有効な題材と言える。

図４－３　簡単なシェルプログラムの実行イメージ



次に，Web プログラミングの一例として，簡単な構造を有する計算機シミュレータ (特に CPU 部分を対象) を作成する例題プログラムを紹介する。具体的には，計算機内部の動作を視覚的にかつ動的に表示することを目的としたシミュレータの作成を課題としたい。プログラミング教育の題材として採用する上でのアイデアとして，

a) 少し規模の大きな Applet を作成し，JAVA による Web プログラミングの可能性を理解させる

b) 文字列処理やグラフィックスなどを盛り込み，ある程度の規模の GUI プログラミングを理解させる

などが可能となる題材を用意している点にある。しかし，プログラミング上のアイデア以外に，

c）計算機，特にCPUやメモリ，の内部構造を視覚的表示し，

d）計算機をステップ毎に動作させ，

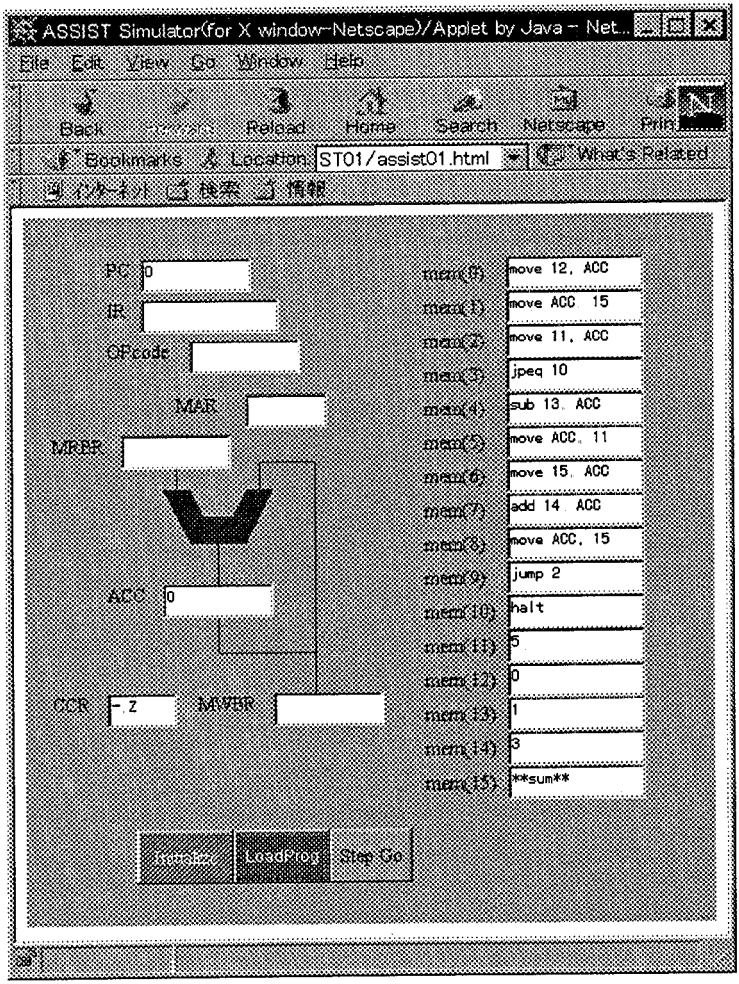
e）プログラムの実行を動的に理解させ，

f）計算機の基本構造や動作原理（フォン・ノイマン計算機の原理）をより具体的に理解させる

などの目的も同時に併せもった教材として提示したいと考えている。

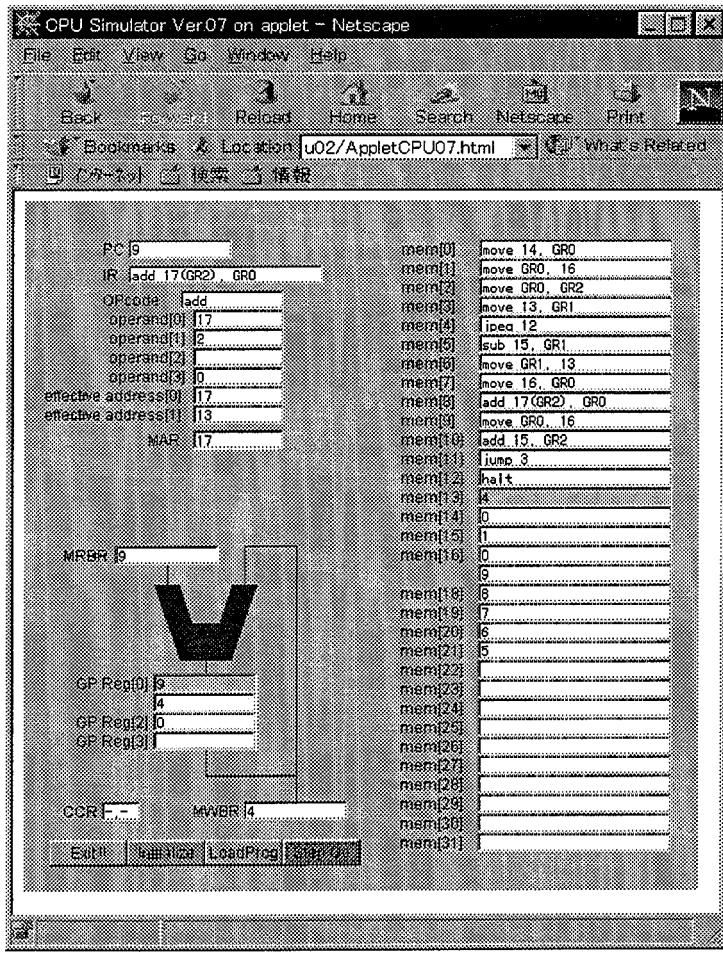
JAVAインタープリタを内蔵するWWWブラウザが広く普及しているため，JAVAの実行環境を個別に準備する必要もなく，作成されたプログラムを簡単に公開して，相互に批評し合える点も有効であろう。この点は教材を配布するという観点からも有効で，WWWサーバ上にAppletとして用意することで，個々のユーザ毎に適宜，必要に応じて配布でき，しかも実行環境はOSなどに依存しないなど，JAVAによるプログラム自体の有効性を示す題材となっている。初心者向きとしては少しサイズの大きいプログラムと思われるが，プロトタイプを示すことで，全体像を理解させ，命令のレパートリを拡充し，アドレス指定方式を増加させるなど，ループを主体とした（実際はイベント処理の固まりでもあるが）プログラムの発展と捉えることもできよう。一方，先ほど強調したように計算機の構造や動作を視覚的に理解するための教育ツールをユーザ自らが作成する題材と位置づけることもできる。いくつかのバージョンを用意しているが(図4－4（a）にCPUタイプ1（ASSIST/1）の実行イメージを，図4－4（b）により複雑なCPUタイプ2の実行イメージを示す)，ここでは入門のための題材ということで，最小モデルのASSIST/1を対象としたプログラムコードを付録に掲載する（リスト4：JAVAによるASSIST/1シミュレータのプログラムを参照)。

図4－4（a） CPUタイプ1の実行イメージ



情報処理教育の一部としてプログラミング教育を扱うという暗黙の前提で，計算機システムから好んで題材を得ている。計算機システム自体が便利になり，誰にとっても利用し易い存在になったことは事実だが，一方では最近急速にブラックボックス化が進み，一般のユーザにとってその内容や内部構造を具体的

図４－４（ｂ）より複雑なCPUタイプ２の実行イメージ



に理解することが非常に困難な状況にある。JAVAプログラミングの題材だけに限定しないが，視覚的に理解し，応用力を身につける工夫がこれまで以上に必要となるであろう。もちろん他にもJAVAプログラミングの題材は多い。しかし，情報処理教育には，計算機システムの理解をある程度前提としている点

があり，両者を可能な限り組み合わせることで，教育効率を上げたいという我々の意図があることは否定できない。他にもアルゴリズムの視覚化やグラフィックス処理などに注力した題材も有望と思われるが，既にJDK内に用意されフリーで配布されているものもある。その意味でも教育現場にとってJAVAプログラミングを題材とすることは有益な結論であろう。

5．ま と め

JAVAプログラミングの導入により，プログラミング教育を中心とした情報処理教育がどのように発展するか，また情報処理教育の題材をどのように設定することができるか，などについて実例を挙げて議論した。プログラミング教育において，題材をどのように選ぶか，どのようなニーズを持ったユーザや学生を対象にするか，などいつも担当者を悩まし続ける課題が多い。一方，計算機，特にパソコンは高機能化，低価格化が今後ますます進み，情報リテラシー教育の拡充が標榜される傾向にある。できるだけ，多くの習得対象者を満足させる情報レテラシー教育が必要である。それと同時に，敬遠されがちなプログラミング教育も扱うべきテーマの充実に意を用いるべきである。そこで，JAVAプログラミングの守備範囲の広さは注目に値する。動作環境や多様なアプリケーション作成能力はプログラミング教育の題材として格好の存在になるであろう。

情報処理教育において，計算機に対するあるレベル以上の理解は，応用力を身につけ，様々な環境で計算機を利用できる為には，不可欠な条件である。計算機システムの原理や動作などを視覚的に捉えることができれば，これまでよりも数多くの初心者（習得者や学生）にとって，前述したような理解も可能となるであろう。プログラミングにおけるテーマを通じて，計算機システム自体を題材とした課題に取り組むことで，理解もより，具体的で，応用力に富んだものとなろう。前節で述べたプログラミングの題材も，JAVAによるプログラミング教育のテーマを拡充させ，計算機システムを視覚的に理解し，より広い範囲に応用できる情報処理教育のための題材の一部として位置づけようとして

いる。JAVA プログラミングをひとりプログラミング教育の効率化のみに適応するのではなく，計算機システムをより効果的に理解するための題材を提供するツールとして捉えたい。これまで Windows PC においてはプログラミング教育自体の扱いも難しかったため，題材として活用できなかった計算機システム内における種々のテーマを簡易に扱うことができるようになろう。JAVA をプログラミング教育に活用することで，UNIX 環境かあるいは Windows 環境かを悩むことなく，扱うべき題材のバリエーションを拡大させることができる。その結果，プログラミング教育を情報リテラシー教育と同等な立場に置くことができ，バランスのとれた情報処理教育を実現することで，新しい情報処理教育の発展にも寄与すると考えられる。

謝　　辞

JAVA プログラミング環境の最新動向を知る上で，㈱技術評論社の JAVA Press 編集長，谷戸伸好氏には大変お世話になりました。香川大学教育学部教授，青木昌三先生には JAVA プログラミング環境についていろいろとご指導いただいております。香川大学経済学部教授，本田道夫先生には，大学卒業以来一貫してプログラミング教育の諸問題を相談し，常に具体的な示唆と御助言をいただいてきました。情報処理センターの曽根計俊氏，丸山久美子氏には本稿を作成する上でいろいろアドバイスをいただきました。ここにお礼を申し上げます。

香川大学経済学研究科修了生である，㈱四国情報通信ネットワークの小手川拓氏，経済学部卒業生である，商工ファンド㈱の森清治君には JAVA プログラミングについて熱心に議論していただきました。経済学部 4 年生，藤原伊知郎君，宮竹憲水君，仲山理子君には実際に JAVA プログラミングに関する挑戦をしていただき，具体的な教育成果について有意義な情報をいただいています。その他，個別に氏名を挙げることはいたしませんが，今井ゼミのみなさんには様々な局面で支援をいただいております。ここに謝意を表します。

参 考 文 献

JAVA に関する参考書は多数存在する。本稿で紹介した JAVA プログラミング教育にも不可欠な要素である。以下では，プログラミングでの題材作成時に参照した参考書を中心に，その一例を示す。

[1] Java プログラミング講座 Sun Microsystems, Inc.，アスキー　1996

[2] JAVAプログラミング実践講座・アプレット作成編 植田龍男 アスキー 1996
[3] JAVAイントラネット構築技法 電通国際システム 星野光一 西川忍 瀬良征志 アスキー 1997
[4] ジャストJAVA Peter Linden (中田秀基訳) アスキー 1997
[5] 入門 Visual Cafe & Java 大津真 日経BP社 1998
[6] JAVA プログラム クイック リファレンス David Flanagan (小俣裕一監訳) オライリージャパン 1998
[7] Java で学ぶはじめてのプログラミング 高橋友一 戸松豊和 サイエンス社 1998
[8] C と Java で学ぶ数値シミュレーション入門 峯村吉泰 森北出版 1999

一方，本稿で述べた筆者の考え方は以下の書籍などで既に公表しているものを書き改めたものである。

[9] 「Linux パワーガイド」(山崎康宏，はねひでや監修) PP. 133-PP. 146 (分担執筆「Linux におけるプログラミング環境」) 技術評論社 1997
[10] JAVA Press Vol. 2 PP. 44-PP. 53 (分担執筆「PC UNIX 上での Java 環境」) 技術評論社 1998
[11] JAVA Press Vol. 6 PP. 57-PP. 61 (分担執筆「Java は情報処理教育をどのように変えるか」) 技術評論社 1999

付録（ソースリスト）

紙面の都合で，本文中で紹介した5本の JAVA プログラムのソースは次の URL に掲載する。必要に応じて参照いただければ幸いです。

http://www.eng.kagawa-u.ac.jp/~imai/JAVA/Ronsou72021/source.lzh