2004－CE－76　(2)
2004／10／2

プログラミングの入門に適した，支援システムとコースデザイン

長　　慎也[†]　日野孝昭[†]　前島真一[†]
小田嶋祐介[†]　佐々木康太朗[†]　筧　　捷彦[†]

プログラミング学習における支援システムを開発し，Java を用いたプログラミングの入門の授業において実践した事例を報告する．授業を行うにあたり，次のような点を重点的に実践した．まず，アニメーションなどの，興味の持てるプログラムを作らせること，次に，最初はプログラムとして書くべきことを少なくし，学習の発展に従い段階的に複雑なプログラムも書けるようにすること，また，ネットワーク教材を活用し，学習状況を把握したりたりすること，などである．この試みは 2003 年から行っているが，その年は，学習内容が進むにつれてそれまでの学習内容との違いに戸惑いを感じる学生が多かったなどの問題点があった．そこで，2003 年までの学習環境，コースデザインに改良を加え，2004 年度のプログラミング授業に再び適用した．

Environments and Course Designs for Introduction of Programming

Shinya CHO,[†] Takaaki HINO,[†] Shin'ichi MAESHIMA,[†]
Yusuke ODASHIMA,[†] Kotaro SASAKI[†]
and Katsuhiko KAKEHI [†]

We designed environments and courses for programming lessons. We mainly focused on three points: First, making attractive program such as animations. Second, starting with simple programs and progressing gradually to more sophisticated programs. And last, using network-based materials to grasp how students have studied. This practice is started from 2003. In that year, however, some students had several troubles as the lesson progressed. We have improved the learning environments and the course design of the lesson. In 2004, we applied them in the programming lesson again.

1. これまでの取り組み

我々はこれまで，早稲田大学コンピュータ・ネットワーク学科（以下，CS 学科）の，プログラミング入門授業（以下，単に授業）にとって適切な支援環境とそれらを使ったコースデザインを開発実践してきた．

2003 年度に行われた授業において，開発，利用した環境と，コースデザインの概要を示す．

1.1 Nigari System

Nigari System[1)] は，プログラミング言語 Nigari と，その環境 Nigari System から構成される．図 1 に Nigari System の動作画面を示す．言語 Nigari と環境 Nigari System は，次のような特徴をもつ．

- **簡素化した言語仕様** 言語 Nigari は，その文法が Java に似ているが，クラス宣言，メソッド宣言は書く必要がない．簡素化した言語によって，初学者にとってとっつきやすく，それでいて実用的な言語への前準備ができるような学習環境になっている．
- **可視化機能** Nigari System は，オブジェクトを画面上に配置し，そのオブジェクトの動作を記述するという手順で開発を行う．オブジェクトは，そのプログラムに従って実行され，オブジェクトのもつ値に応じて画像が表示され，値の変化が自動的にアニメーションとして表示される．この機能により，魅力的な出力を生成でき，学習者の興味を惹くことができる．また，画面上にオブジェクトを画像として表示させることで，オブジェクトの存在を認識させ，オブジェクト指向の考え方の基盤を作り上げることができる．

1.2 ネットワーク教材

授業においては，各学生が貸与されたノート PC を持ち込み，演習を行う形式をとる．このため教材を

† 早稲田大学 理工学研究科
Graduate School of Science and Engineering, Waseda Uniersity

図 1 Nigari System の動作画面

すべて PC 上からアクセスできるようにした．詳細を次に示す．

- **Web 教科書** すべての教材を Web から閲覧可能にして，この教材に従って毎回の授業をすすめた．
- **Web 演習問題解答システム** Web 教科書の合間に，現在学習している内容についての小問（穴埋め問題など）を入れることで，授業中には，つねに手を動かしてもらい，学習者を退屈させないようにした．
- **ネットワーク経由でのプログラム提出** 上で述べた穴埋め形式の小問の他に，プログラムを書かせる形の演習問題も出題した．この演習問題を解く際に，Nigari System で実行したプログラムを，Nigari System から直接提出できるようにした．

1.3 コースデザイン

授業においては，授業全体の前半では Nigari System を使い，そのあと Java に移行する，という順番で授業を行った．

Nigari System を用いた前半部分には，演習問題としてアニメーションを作成させるような問題も取り入れ，興味の持てる成果物を作らせるようにした．

2. 2003 年度の結果と課題

2003 年度における授業においては，次のような結果が得られた．

2.1 Nigari System

授業で使用したときの状況をまとめると，Nigari System には，次のような有用性および，問題点が明らかになった．

- **可視化機能** 学習者の興味を惹くために非常に有用であった．しかし，表示されたものを「オブジェクト」として捉えるための機能としては貧弱であり，さらに詳細な情報を表示させる仕組みが必要である．特に配列を表示する機能がなく，配列を Nigari System を使って教えることができなかった．
- **簡素な言語仕様** 難しい点を意識しないでプログラムが書け，言語 Nigari を使っている時点まではわかりやすいとの好評を得ることができた．しかし，やがて Java に移行するに従い，学生のとって Nigari の魅力がうすれてきた．これは，Java に移行した瞬間に書くことが多くなり，突然難しくなって戸惑いを覚えたという学生が多かったためであると考えられる．

2.2 コースデザイン

Nigari System と Java では，まったく環境が異っていたため，Java に移行したときに，代入や制御構造の仕組みなどをもう一回教える必要があった．そのため，授業時間にゆとりがなくなり，後半は授業が速くなったため，あまり理解ができなかった学生が多かった．

2.3 ネットワーク教材

教材自体はおおむね好評であったが，プログラムを提出する機能においては，そのプログラムに関する説明文と，プログラムの実行結果を付加する機能がなく，レポートの体裁を整える訓練ができなかった．

2.4 学習項目別にみた課題

変数の概念，if 文，while 文などの制御構造を Nigari System で学ぶことについては，学生から比較的よい評価が得られたが，次の項目については，Nigari System を使うことがかえって弊害になったという意見が目立った．

- **配列** 配列は Nigari System では可視化することができず，Java に移行してからしか教えなかったため難しいと感じる学生がいた．
- **メソッド** Nigari と Java では，メソッドの宣言の仕方がかなり異なるため，戸惑った学生が多かった．

3. 2004 年度の取り組み

2004 年度は，前年度の反省を踏まえ，環境およびコースデザインの改良を行い，再び CS 学科の授業で適用する実験を行った．

3.1 プログラミング学習システム N-Java

ここでは，Nigari System を元に改良された学習システム N-Java を紹介する．N-Java の動作画面を図 2 に示す．

Nigari System から追加された主な機能は次の 2 つ

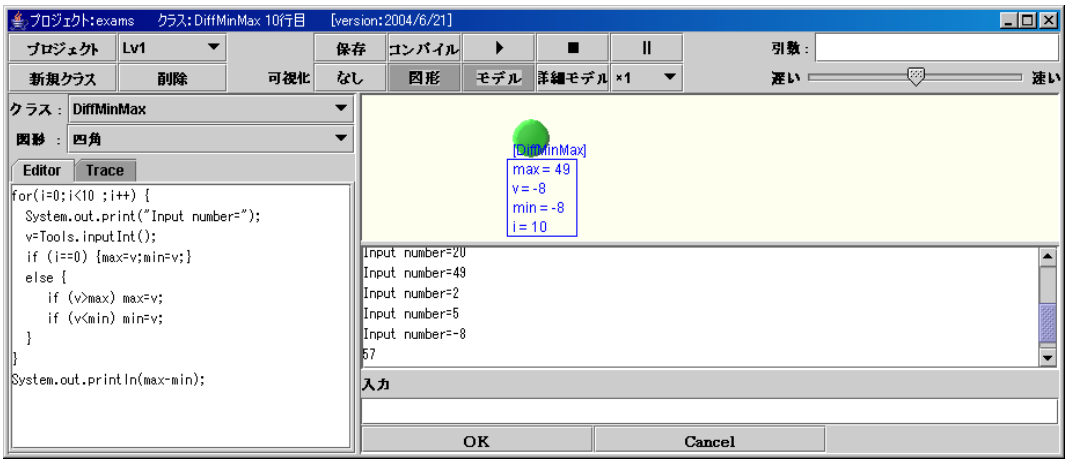

図 2　N-Java の動作画面

である．これらの機能を順次紹介する．

- 4 レベルエディタ
- モデル表示

3.1.1　4 レベルエディタ

2003 年度の授業では， Nigari と Java の 2 つの言語を使用したが，その 2 つの言語の間に大きな差があり，Java に移行したときに戸惑いを覚えた学生が多かった．そこで，N-Java は，まず，Nigari と Java の両方を実行できるようにし，さらに学習が進展するに従い，プログラムの書き方を段階的に変化させられるよう，つぎの 4 つの段階（レベル）を設けた．

- **レベル 1**　このレベルでは，図 3 のような Nigari のプログラムを記述する．Nigari は，変数宣言，クラス宣言，メソッド宣言はすべて不要で，手続きのみを記述すればよい．
- **レベル 2**　これ以降のレベルでは，Java のプログラムを記述する．レベル 1 と異なり，変数宣言が必要となるが，クラスやメソッドの宣言は自動生成される．レベル 2 では，エディタの形状が変化し，「変数宣言」と，「メインルーチン」の 2 つの編集部分が現れる．エディタに図 4 の左側で示したプログラムを入力すると，図の右側に示した Java のプログラムが自動生成される．
- **レベル 3**　レベル 3 では，再び編集部分は 1 つにまとまるが，レベル 2 に加え，メソッドの宣言も書く必要がある．図 5 の左側のプログラムを記述すると，右側のプログラムが自動生成される．
- **レベル 4**　図 6 のように，クラス宣言を含めた，Java プログラム全体を入力する必要がある．

N-Java は，レベル 2 以降で Java のプログラムを実行する際には，現在エディタ上に表示されているクラスのインスタンスを作成し，そのインスタンスの run メソッドを呼び出して実行を行う．JDK と異なり，main メソッドから実行が始まるわけではない．

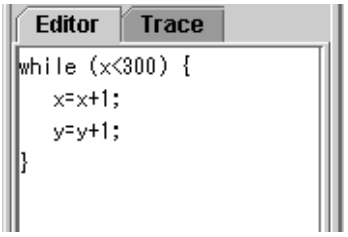

図 3　レベル 1

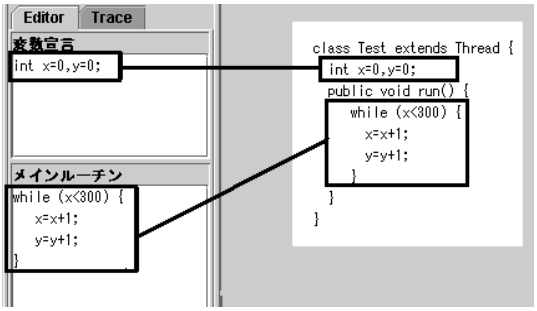

図 4　レベル 2

3.1.2　モデル表示

Nigari System の可視化機能は，オブジェクトの存在を意識させ，オブジェクト指向の基礎を学ばせるという目的をもっていたが，単に画像が表示されただけでは，オブジェクトのもつ変数がどのような状態で，他のオブジェクトとどのような関係をもっているか，あるいは配列の要素にどのような値をもっているか，などということを詳しく把握できなかった．そこで，

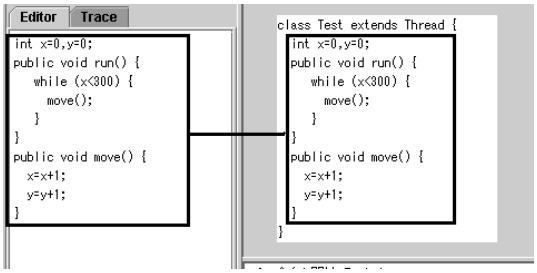

図 5 レベル 3

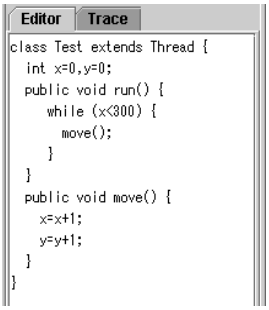

図 6 レベル 4

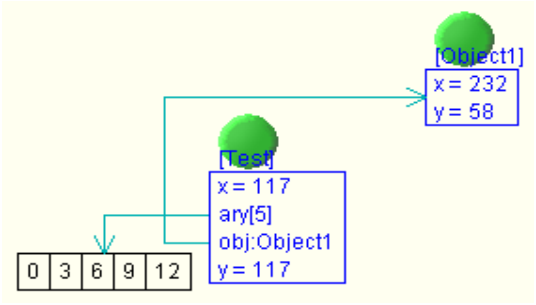

図 7 モデル表示機能

N-Java では，オブジェクトや配列のデータ構造（これらをモデルと呼ぶ）を表示するモデル表示機能を用意した．モデル表示の例を図 7 に示す．

- **変数表示** オブジェクトのもつすべての変数の内容を，文字情報として表示する．
- **配列表示** 配列の中身をすべて表示する．
- **参照関係表示** あるオブジェクト（配列を含む）が他のオブジェクトを参照している場合は，それらを矢印で表示する．

また，従来の Nigari System と同様に画像のみを使ったアニメーションだけを表示させたい場合は，モデルを非表示にすることも可能である．

3.1.3 実　　装

N-Java は Java（JDK1.4 以降）で実装されている．Nigari のプログラムは独自の仮想機械を用いて実行する．また，Java のプログラムは，実行中のオブジェクトの情報を捕捉するために，Java Platform Debugger Architecture[2] を用いて実行している．

3.2 ネットワーク教材

ネットワーク教材の改善点は，レポート作成機能を追加した点である．

プログラムリストを伴う課題を提出する際に，プログラムそのものだけでなく一緒に文章を書き込めるようにした．これにより，プログラムの解説や，実行例の添付も可能になり，プログラミングに対する理解度を詳細に知ることができるようになった．

なお，レポートの作成は Web にて行うため，Nigari System にはあった，プログラミング環境から直接プログラムを提出する機能はなくなったが，学生の作成したプログラムを調査し，学習状況を把握するために，PC がネットワークに接続されている場合には，学生の作ったプログラムを，専用のサーバに自動的にアップロードする機能を N-Java に搭載した．（アップロードされた内容は非公開）

3.3 コースデザイン

新しい学習環境に合わせ，コースデザインも前年度と比べて若干変更を行った．

3.3.1 概　　要

- **授業名** プログラミング A
- **対象学科・学年** 早稲田大学理工学部，コンピュータネットワーク工学科（CS 学科）1 年
- **目標** Java 言語とプログラミングの基礎の習得
- **期間，回数** 2004 年 4 月 12〜同年 7 月 5 日，12 回
- **一回あたりの授業時間** 2 時限（約 3 時間）
- **人数** 約 240 人
- **クラス構成** 3 クラスに分かれて授業．授業用資料や課題は担当教員により異なる．クラス分けは希望調査により決めた．希望もれはなかった．
- **教科書** 「Java 言語プログラミングレッスン（上）」[3]
- **試験** クラス試験，共通試験 1 回ずつ．クラス試験はクラスによって内容が異なる．共通試験は，Web を用いて行った．
- **授業形態** 各自ノートパソコン持参，授業 1 回ごとに，説明を受け，演習を行う．毎回 2，3 題程度（合計 22 題）のレポートを出題．

第 1，2 クラスは初心者クラス（ただし，プログラミング経験者も含む），第 3 クラスは中級者クラスとした．第 1 クラスでは，N-Java を利用した授業を行った．

3.4 授業の詳細

各授業の詳細は次の通りである．参考として，2003年の授業計画と比較して図8に示す．

一部の項目について，補足説明をしておく．

- **「いろいろな値」** 値の型についての概念，文字列型，整数型，実数型の特徴，代入の互換性など．ただし，変数の宣言については触れていない．
- **「問題集」** 数値計算（モンテカルロ法，ニュートン法など）の演習問題．
- **「レポート解説」** これまで出した課題の中で特に難しかったものについて，ヒントとなるような練習問題の出題．
- **「JDK」**N-Javaではなく，JDKを直接使う演習．
- **「デバッグ」** わざと間違ったプログラムを提示し，修正をさせる演習．

2003年度においては，第8回まではNigari Systemを使用，第9回からJavaへ移行した．その際に，制御構造などをJavaに合わせた格好で再度学習させたが，2004年度においてはこの部分は省略した．代わりに問題集，解説等の項目を増やした．

3.4.1 レベル別学習項目

学習内容にあわせて，N-Javaの4レベルエディタのレベルを次のように変化させた．

- レベル1は,「問題集」まで使用．変数，制御構造の基礎を学ばせるのに使った．
- レベル2は,「変数の宣言と変数の型」から「レポート解説」まで使用．変数宣言を伴うプログラムを作成させるのに使った
- レベル3は，メソッドを学習させるのに使った．
- レベル4は，あまり使用しなかった．JDKを学習するにあたり，今までN-Javaのプログラムと，実際のJavaプログラムとの対比をさせる程度にとどめた．

4. 結　　果

4.1 アンケート結果

この授業に対するアンケート調査を行った．ここでは,「良かった点」「悪かった点」について報告する．

4.1.1 授業の良かった点

学生が答えた，授業の良かった点を図9に挙げる．

まず，圧倒的に，授業のわかりやすさが支持されたが，N-Javaを用いたことによる良さを評価した学生も16人いた．そのうち，4レベルエディタを使ったことが良かったという意見が7件，アニメーションが表示されて面白いという意見が2件であったしかしながら，モデル表示機能に関する評価はまったく見られなかった．

項目	人数
授業がわかりやすい	26
N-Javaが良かった	16
（4レベルエディタ）	(7)
（アニメーション表示）	(2)
（その他）	(7)
TAによるサポートが良かった	7
Web教材が良かった	5
その他	8

図9 アンケート:この授業の良かった点

項目	人数
レポートが多い	10
レポートの解説が足りない	6
レポートが難しい	6
授業のスピードが速い	5
授業が難しい	4
N-Javaより JDKを使いたかった	3
Web教材に不満	3
その他	7

図10 アンケート:この授業の悪かった点

4.1.2 授業の悪かった点

学生が答えた，授業の悪かった点を図10に挙げる．

ほとんどの学生が，授業や課題の難易度，速さに関する不満を述べた．

2003年では，Nigariから Javaへの移行が遅い，Nigariから Javaになったときに戸惑ったなど，環境の移行に関する不満が10件ほどあったのに対し，2004年度では，N-Javaにより，JDKを使う機会が減ったという意見がわずか3件あっただけで，環境の移行に関する不満はあまり見当たらなかった．

4.1.3 4レベルエディタにおけるレベル間移行

アンケート項目には，4レベルエディタにおいて「レベル1から2に移行したときに戸惑いがあったか」「2から3に移行したときはどうか」という設問を設けた．これらの設問の結果および，2003年度のアンケートにあった「Nigariから Javaに移行したときに戸惑いがあったか」という設問の結果を図11に示す．

この結果によれば，レベル1から2，つまり変数宣言を行うようになった時点ではあまり戸惑いを感じることなく移行できたが，レベル2から3，つまりメソッドを定義するようになった時点では，約半数の学生が戸惑いを感じた，この割合は，2003年度において，Nigariからいきなり Javaに移行したときと同じ

回	2004 年度	2003 年度
1	講義概要	講義概要
2	N-Java 基本操作	PC 基本操作
3	変数/ いろいろな値	Nigari 基本操作
4	while 文/ if 文 (1)	変数 /while 文 (1)
5	if 文 (2)/ 問題集	while 文 (2)/if 文 (1)
6	問題集 / 変数宣言と変数の型	if 文 (2)
7	配列	複数のオブジェクト/マルチスレッド
8	オブジェクト	メソッド / オブジェクトの実行時生成
9	レポート解説 (※)	Java の概要 / 変数の宣言 / if 文 (3) / while 文 (3)
10	メソッドとローカル変数	制御構造/メソッド/文字列
11	メソッドの復習	配列 / コマンドライン引数 / 並べ換え
12	JDK/ デバッグ	並べ換え（つづき） / 文字入力

図 8 授業内容

	戸惑いなし	戸惑いあり
レベル 1 →2	76%	24%
レベル 2 →3	52%	48%
Nigari→Java	48%	52%

図 11 アンケート:移行したときに戸惑いがあったか

	件数	ポイント
エディタ	23	9.5
コンパイル／実行ボタン	43	21.4
アニメーション表示	33	16.0
オブジェクト表示	40	15.7
配列表示	44	22.3

図 12 アンケート:便利な機能

程度の戸惑いであると推測される.

4.1.4 便利な機能はどれか

N-Java について，便利な機能はどれか，という質問（複数回答可）に対する結果を図 12 に示す．図中の「件数」は，その項目が選ばれた個数,「ポイント」は，学生 1 人あたりの持ち点を 1 点とし，その学生が選んだすべての項目に対して均等に加点を行った合計である.（例えば，ある学生が 2 つの項目を選んだ場合，それら 2 項目に 0.5 点ずつが加点される）

N-Java は，コンパイル，実行がすぐにできることと，配列が表示されることが，特に便利な機能であるという結果になった.

4.2 成　　績

全体の成績は，平常点，クラス別期末テスト，共通テストを用いて決めた．共通テストは，第 1，第 2，第 3 クラスすべての学生に出題され，プログラムや文章の一部を選択肢または自由記入によって穴埋めするテストである.

クラス	平均点
1	47.2
2	58.8
3	61.9

図 13 共通テストのクラス別平均点

共通テストの結果を図 4.2 に示す．このように，第 1 クラスの平均点は，他のクラスより低い，という結果になった.

5. 考　　察

5.1 コースデザイン

前回に比べて，授業計画にゆとりをもたせることができた．これは，最初から最後まで N-Java という 1 つのアプリケーションを利用するため，学生が途中で新たに操作方法を習得する必要がなく，4 レベルエディタのレベルの切り替えによるレベル間の違いにそれほど戸惑いがなかったためと考えられる.

しかしながら，それでも学生の多くが授業の分量や速度について，多すぎる，速すぎるといった意見をもった．これは，後半の配列やメソッドなどの概念が予想外に高度であり，またそれらを応用した問題の難易度が高かったためと考えられる.

5.2 N-Java を利用する

アンケートの自由記入欄（4.1.1）によれば，4 レベルエディタによって段階的に変える方法はよいという評価が多く得られた.

また，便利な機能についてのアンケート（4.1.4）では，配列の表示がされることがもっとも評価され，ア

ニメーション表示や，オブジェクトの値の表示は，配列の表示ほどではなかったが，これは，去年ほどアニメーションの作成を積極的に授業に盛り込んでいないことや，授業のなかでオブジェクトの概念がそれほど必要でなかったことなどが原因として考えられる．

5.3 ネットワーク教材

ネットワーク教材の改善により，学生のプログラムを見るだけでなく，解説や実行例を載せることで，学生の学習態度を知ることができた．

例えば，解答の中には，解説がほとんど載っていない，掲載されているプログラムを実行しても実行例のとおりにならない，といった解答が見受けられた．このように，プログラムのできの問題以前に，レポートの書き方が不十分であるといった，学生の問題点を見つけることも可能になった．

また，N-Java から送られてきた学生のプログラムを記録し，学生が理解できていない点を解析することも可能になった．その結果は 6 にて後述する．

ただ，レポートの採点をするための仕組みには問題があった．採点を行った後，再提出の必要があった学生がレポートを修正した場合には再採点をする必要があるが，修正があったかどうかを知る手段がなく，定期的に再提出者をすべて調べなければならなかった．

6. 学生の行動からみる，授業の改善点

今回，N-Java を利用したクラス（第 1 クラス）の共通試験の成績が低かった理由を探るため，提出されたレポートや，N-Java から送信されたプログラムの履歴を解析する試みを行った．

学生が特に難しいと感じた項目に，配列とメソッドが挙げられる．これらの教え方について，問題点を抽出してみた．

6.1 配列の問題点

配列の学習において，次のような学生の行動が目立った．

- **配列を使わない**
  そもそも，配列を使わないでも解ける問題というものがあり，十分意義が伝わらない場合があった．例えば，配列の問題として典型的な「入力した値を配列に入れ，配列要素の合計・最大値を求める」といった問題は，入力した数値を逐一処理することで，配列を使わないでも，同一の目標が達成できてしまう．
  また，配列を使うべき問題でも，要素数の最大がわかっている場合，配列でない変数をその数だけ用意してしまう（例：`a[0]` ... `a[9]` の代わりに `a0` ... `a9` を 10 個用意する）学生がいた．
  この問題の対処法としては，配列の要素数を可変にさせる，例えば，要素の個数を最初に入力した数値により決定する，といった方法がある．

```
for (j=0 ; j<10 ;j++) {
  if (j==b[i]) a[j]=n;
}
```

図 14 配列の添字の適切でない使い方

- **配列の添字の使い方が不十分**
  例えば，図 14 のように，`a[b[i]]=n;` と書く代わりに，ループ使った非効率な処理を行っている学生がかなり見られた．
  この原因としては，初歩的な例においては，`a[i]` という，添字内に変数を単独に書く例があまりに多いため，`a[i+1]` や `a[b[i]]` といった，添字に任意の式が書けるということに気づいていない学生が多かったためであると考えられる．

6.2 メソッドの問題点

例えば，図 15 のような問題に対して，図 16，図 17 のような解答を行う学生が多かった．図 16 では，値を返すことなく，メソッドの内部で結果を標準出力に表示してしまっている．また，図 17 では，値を返す代わりにインスタンス変数に代入してしまっている．さらに，いずれの解答も，本来変更すべきでない `run` メソッドに変更を施してしまっている．しかも最大の問題は，プログラム全体としては，正解とまったく同じ動作をしてしまう点である．

このような解答が行われた背景には，次のような要因があると考えられる．

- **インスタンス変数をすでに知ってしまっている**
  N-Java では，インスタンス変数を利用しないと，その値が可視化されない．そのため，ローカル変数は授業であまり利用せず，ローカル変数はメソッドを習った時点で導入した．
  しかし，インスタンス変数を利用すれば，値を渡したり，値を返したりすることも可能である．上で示した例のように，誤った解答でも正しい出力が得られるのだから，という理由からか，ローカル変数や，値を返す仕組みの必要性が感じられなかったものとみられる．
- **メソッドの存在意義がわからない**
  そもそも，なぜメソッドを作成するのかがわからない，という学生が多かった．その理由は，プログラムが小規模であるため，わざわざメソッドに

```
public void run () {
  System.out.println(repeat("abc",3));
}
public String repeat(String a,int b) {
// この部分を埋め，a を b 回連結した
// 文字列を返すようにせよ.
}
```

図 15 メソッドの演習問題の例

```
public void run () {
  repeat("abc",3);
}
public void repeat(String a,int b) {
  for(int c=0;c<b;c++)
    System.out.print(a);
}
```

図 16 メソッドに関する誤解 1

```
String buf;int c;
public void run() {
  repeat("abc",3);
  System.out.println(buf);
}
public void repeat(String a,int b) {
  for(c=0;c<b;c++)
    buf+=a;
}
```

図 17 メソッドに関する誤解 2

しても恩恵が得られない，という点である．

また，メソッドを自分で作って自分で使うため，定義と使う部分が分けられている，という意識ができなくて，メソッド本来の意義が実感できなかったものと考えられる．

6.3 その他の問題点

他にも，レポートの解答結果を実際に実行してみたところ，実行が正しく行われない，という例がかなり見られた．

主な理由は，特定の入力（特に，レポートの説明時に例示された入力）しか考慮しておらず，それ以外の入力があった場合の事態を想定していない，という点である．

例えば，図 18 のプログラムは，配列 a の要素の最大値を変数 max に代入するつもりのプログラムであるが，このプログラムは配列 a の最後の要素が最大値をもつ場合のみ正しく動作し，それ以外では正しく動作しない．

```
for (i=0 ; i<a.length-1 ;i++)
  if (a[i]>a[i-1]) max=a[i];
```

図 18 特定の入力にしか使えないプログラム

このように，たまたま正常に動作する入力だけを試して，正しく動作する，と結論づけているレポートが多かった．

これを防止するには，実行例，入力例を複数提示し，提示されたすべての入力に対応するプログラムを作ることを徹底させる必要がある．

7. 今後の展望

プログラミング学習のための環境として，N-Java および Web 等のネットワーク教材を利用した事例を報告した．

授業の進度（レベル）に合わせて，プログラムに書くべきことを変化させたり，データ構造を可視化したりすることにより，Nigari という簡単な言語から，Java という複雑な言語への導入を去年と比べて，スムーズに進めることができた．

しかし，メソッドや配列など，初学者にとっては高度な概念については，教え方，環境の改善が必要である．今後は，問題例や実行例を充実させたり，学生が起しがちな誤解，間違いに対して早期発見と早期対処を行えるような環境を整備したりすることを目指す．

参 考 文 献

1) 長慎也, 甲斐宗徳, 川合晶, 日野孝昭, 前島真一, 筧捷彦: プログラミング環境 Nigari - 初学者が Java を習うまでの案内役, 情報処理学会論文誌：プログラミング, Vol. 45, No. SIG9, pp. 25–46 (2004).
2) Sun Microsystems: Java Platform Debugger Architecture. http://java.sun.com/products/jpda/.
3) 結城浩: Java 言語プログラミングレッスン (上), ソフトバンク パブリッシング (1999).