初学者におけるオブジェクト指向プログラミングの難しさの分析と教授の工夫

土肥 紳一† 宮川 治† 今野 紀子†

概要

本学部のプログラミング教育は，手続き型の考え方を学習する「コンピュータプログラミングA」とオブジェクト指向の入門を学習する「コンピュータプログラミングB」を開講している．初学者が「コンピュータプログラミングB」でオブジェクト指向を学ぶ中で，最初につまずくところがある．授業毎に実施している理解度調査の結果を活用し，初学者が最初につまずく授業内容を明確にし，その原因を探る．この問題を回避するために，私たちが取り入れている教授の工夫について述べる．

Teaching Method and Analysis of Difficult in Object Oriented Programming for Novice Programmer

Shinichi Dohi† Osamu Miyakawa† Noriko Konno†

Abstract

Programming Educations in the School of Information Environment are Computer Programming A which learns the procedural programming and Computer Programming B which learns the object oriented programming. When novice programmers learn the object oriented programming, there is a first point that they stumble. Using results of questionnaire of students' understanding level at each class, we clarify the class that novice programmer stumble, and investigate the cause. We describe some ideas of teaching method that we are introducing.

1. はじめに

プログラミングの初学者は，大学入学以前のプログラミング経験者に対して，劣等感を抱く傾向がある．プログラミングの経験が無いため，このように思うことは自然であろう．大学入学以前のプログラミング経験の有無が，コンピュータプログラミングの授業にどのような影響を与えているのかを探るために，「入学前にプログラミングを経験してきた人」と「経験していない人」に分け，成績との関係を調べた．調査した科目は，入学後最初にプログラミングを学ぶ，「コンピュータプログラミングA」である．調査した結果，「入学前にプログラミングを経験してきた人」は，成績が良いことが示された[1]．一方，オブジェクト指向は，大学入学以前に教わることはほとんど無く，「コンピュータプログラミングA」を教わった後に至っては，受講者は同じ土俵に居ると考えられる．このような状況の中で，オブジェクト指向の入門を教わる初学者は，授業のどこで最初につまずくのか，その本質が何なのかを探る．それを乗り越えるために取り組んでいる教授の工夫について述べる．

†東京電機大学 情報環境学部
Tokyo Denki University, The School of Information Environment

2. 受講者の状況

2.1 大学入学以前のプログラミング経験

入学直後に集中講義として開講しているカリキュラム計画の授業の中で，プログラミング等の経験を調査しており，調査結果を**図 1** に示す．2012年度は，大学入学前のパソコン所有率は約 80%，プロバイダ契約は約 50%，プログラミング経験は約 20%であった．2001年度のプログラミングの経験者は約 30%を超えていたが，その後，約 20%に止まっている．

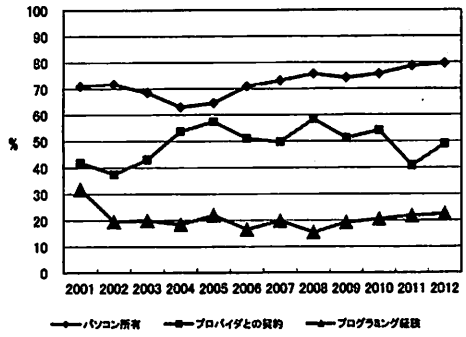

図 1 新入生のプログラミング経験

2.2 普通教科「情報」受講状況

2003 年度から普通教科「情報」が開講され，「情報 A」「情報 B」「情報 C」は，1 科目が必履修となっている．普通科の高等学校では，情報活用の実践力の育成を目的とした「情報 A」が約 80%採用されており，情報の科学的な理解を目的とした「情報 B」は約 10%に止まっている[2]．本学部が調査を開始した 2006 年度から 2012 年度までの履修状況を図 2 に示す．2012 年度は「情報 A」を履修している学生が 71.2%，「情報 B」が 12.7%，「情報 C」が 16.1%の結果になった．

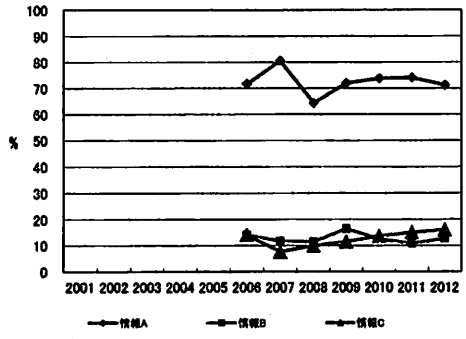

図 2 普通教科「情報」の履修状況

2.3 オブジェクト指向の初学者の感想

オブジェクト指向を学習する「コンピュータプログラミング B」の授業を長年に渡って担当しているが，最初に寄せられる受講者の感想は，「もっと簡単に作れそう」「用語が難しいです」等が挙げられる．

「もっと簡単に作れそう」の指摘は，もっともである．例えば，ディスプレイ上に学籍番号と氏名を表示するプログラムを考える場合，手続き型の考え方で Java を使ってプログラムを記述すると，System.out.println( ) 等の表示を行う文を記述するだけで，目的のプログラムを完成できる．ところが，オブジェクト指向の考え方でプログラムを記述しようとすると，例えば，学生(Student)クラスを定義し，main メソッドの中で Student オブジェクトを生成し，そのオブジェクトに何らかのメッセージを送り，返却値を表示するプログラムを記述することになる．オブジェクト指向の授業が始まったばかりの頃は，受講者にとって面倒な事だけが強調され，オブジェクト指向の何が凄いのか，さっぱり理解されない事態が生じる．

「用語が難しいです」も，素直な感想である．カタカナの専門用語が溢れ，その意図することが理解できるまでの敷居は高い．

3. カリキュラム体系

3.1 プログラミングを支える 3 つの科目

本学部で開講している科目は，全て選択科目である．このような状況の中で，コンピュータプログラミングを支える代表的な 3 つの科目がある．それは，「コンピュータプログラミング A」，「コンピュータプログラミング B」そして「オブジェクト指向設計」である．「コンピュータプログラミング A」は，手続き型の基本的な考え方を，「コンピュータプログラミング B」はオブジェクト指向の入門を，「オブジェクト指向設計」はソフトウェアの分析・設計の基礎を学ぶ．これらの 3 科目を履修することによって，自らソフトウェアを作成できるスキルが身に付くように体系化されている．プログラム言語は Java を学ぶ．

3.2 3 科目の履修状況

「オブジェクト指向設計」を履修する時期を想定し，本学部が開設してから 2009 年度までの入学者を対象に，2012 年 3 月末における履修状況を調査した．「コンピュータプログラミング A」の履修者は 2046 名，「コンピュータプログラミング B」の履修者は 1780 名，「オブジェクト指向設計」の履修者は 874 名であった．入学者総数 2168 名のうち 94.4%が「コンピュータプログラミング A」を履修し，82.1%が「コンピュータプログラミング B」を，40.3%が「オブジェクト指向設計」を履修していることが分かった．また，「コンピュータプログラミング A」の履修者のうち 86.2%が「コンピュータプログラミング B」を，「コンピュータプログラミング A」と「コンピュータプログラミング B」の履修者のうち 48.5%が「オブジェクト指向設計」を履修していることも分かった．一般的に，先に履修した科目でプログラミングに対する興味や関心が損なわれた場合，後続のプログラミン

グの科目は履修しない．学費単位従量制の制度によって，この傾向はさらに強まる．このような状況の中，先に示した履修割合は非常に高い数値であると考えている．

3.3 教育目標と授業内容

「コンピュータプログラミング B」の教育目標は，クラス図とAPI仕様が与えられると，その仕様を満足するソースプログラムをスクラッチプログラミングで完成できることである．クラス図やAPI仕様は，教授する側で準備し，受講者に与える．クラスの設計やAPI仕様の設計は，「オブジェクト指向設計」の授業で教わる．「コンピュータプログラミング B」の授業内容は，表 1に示す[3]．授業は50分を 2 コマ連続して週 2 回実施しており，1セメスターで27回の開講となる．

4. これまでの問題点

4.1 理解度調査について

「コンピュータプログラミング B」の授業では，授業毎の理解度調査を目的にアンケート調査を実施している．例として8回目の授業内容に対するアンケート調査項目を表 2に示す．アンケート調査項目の回答は「はい」「いいえ」の二択になっており，web ベースのアンケートシステムを開発し，回収している[4]．調査項目は，課題の提出および自由記述欄を除き，授業毎に異なる．この調査は，授業時間内で解答できる程度の課題の提出を兼ねている．

アンケートの調査結果は web で公開し，次回の授業の冒頭で講評する．調査項目に対する理解度は80%を目標と考えており，これを下回った場合は，授業内容自体に問題があったと判断し，後日の授業で補足する．アンケート項目の中には，授業に対する要望や感想を記載できる自由記述欄を設けており，匿名性を維持しながら公開している．自由記述は，教授者にとって有貴な情報が含まれている．

4.2 調査対象のクラスと年度について

過去のアンケート調査結果を振り返ることによって，授業毎の理解度を分析できる．調査対象は，5 クラスにクラス分割している授業の中で，C 先生の 2010 年度を対象とした．2010 年度は，新設された i-room に移り，教授者が新しい教室環境に慣れるのに手間取った時期である．特に，教室内に分散配置された 3 面のスクリーンの取り扱いに苦労した．2011 年度は，i-room の教室環境を使いこなす工夫に専念し，ペンタブレットを工夫する方法等を思い付いた時期でもある．

4.3 最初につまずく授業の発見

授業の理解度は，授業毎の各調査項目に対

表 1 コンピュータプログラミング B の内容

回	主な内容
1	第 1 章 ガイダンス，Java 等のセットアップ
2	第 2 章 非手続き型言語，オブジェクト，オブジェクトの生成等
3	第 2 章 オブジェクト図，状態・振る舞いの追加等
4	第 2 章 クラス図とソースプログラムの関係等 (Student クラスの完成)
5	第 2 章 クラス図とソースプログラムの関係等 (Teacher クラスの完成)
6	第 3 章 サイコロ(Dice)オブジェクトの生成等(Dice クラスの完成)
7	第 3 章 複数の Dice オブジェクトの生成と ArrayList オブジェクトによる格納等
8	第 4 章 Cup クラスの完成等
9	第 4 章 Book クラス，Bookshelf クラス等
10	第 5 章 キーボードからの入力，DiceGame クラス等
11	第 5 章 特殊なサイコロ，インタフェース等
12	総合復習
13	中間試験
14	第 6 章 Coin クラス，Check クラス，インタフェースの活用等
15	第 6 章 CoinBox クラス等
16	第 4，6 章 Bookshelf，Wallet クラス等
17	第 7 章 Educatee クラス等
18	第 7 章 参照，ArrayList 等
19	第 8 章 ビンゴゲーム(Ball インタフェース等)
20	第 8 章 ビンゴゲーム(Box インタフェース等)
21	第 8 章 ビンゴゲーム(Bingo インタフェース等)
22	第 8 章 ビンゴゲーム(全体のクラス図の関係等)
23	第 9 章 継承等
24	第 10 章 応用
25	総合復習
26	期末試験
27	第 11 章 ストリームについて

表 2 アンケート調査項目 (8 回目の授業)

質問 1 Cup クラスのクラス図からソースプログラムを機械的に導出する方法は，理解できましたか．
質問 2 Cup クラスの add メソッドのソースプログラムは，理解できましたか．
質問 3 Cup クラスの get メソッドのソースプログラムは，理解できましたか．
質問 4 Cup クラスの size メソッドのソースプログラムは，理解できましたか．
質問 5 Cup クラスの cast メソッドのソースプログラムは，理解できましたか．
質問 6 Cup クラスの getSum メソッドのソースプログラムは，理解できましたか．
質問 7 Cup クラスの getValue メソッドのソースプログラムは，理解できましたか．
質問 8 引数は，理解できましたか．
質問 9 返却値は，理解できましたか．
質問 10 ここに記述する内容は，授業中に指示します．
質問 11 授業に対する要望，感想等があれば記入してください．

する「はい」と「いいえ」の総数を求め，その割合で表現した．2010 年度の推移を図 3 に示す．この結果から授業が潜在的に持っている難しさが，何回目の授業に存在するのかを読み取ることができる．

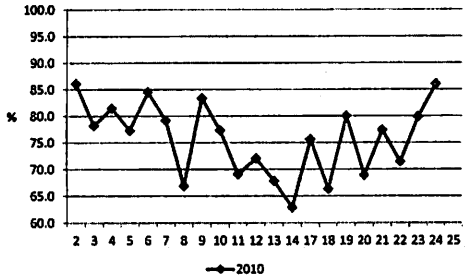

図 3 授業毎の理解度(2010 年度)

4.4 8 回目の授業内容

初学者が最初につまずく授業は 8 回目であることが分かった．8 回目の授業の中に，初学者が理解に苦しむ本質が隠れている．表 1 に示した通り，8 回目の授業内容は，「第 4 章 Cup クラスの完成等」である．具体的な内容は，ArrayList オブジェクトを使って，カップ(Cup)クラスを作成する授業である．Cup クラスは，6 回目の授業で作成したサイコロ(Dice)オブジェクトを任意の個数入れられる仕様となっている．Dice クラスの関係は図 4 に示す．Dice クラスのクラス図と API 仕様を図 5 と図 6 に，Cup クラスのクラス図と API 仕様を図 7 と図 8 に示す．

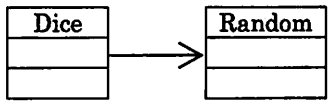

図 4 Dice クラスの関係

Dice
random : Random value : int
Dice( ) cast( ):void getValue( ):int

図 5 Dice のクラス図

5. つまずきの原因

5.1 Dice オブジェクトの作成

Dice オブジェクトは，6 回目の授業で作成する．Dice オブジェクトの仕組みは，Random オブジェクトを状態として持っており，Dice オブジェクトを振る(Cast)毎に，Random オブジェクトを使ってサイコロの目を変更する．すなわち Dice オブジェクトの状態 value を変更する． 図 3 から分かるように，6 回目の授業は 80%を超える理解が得られており，それほど難しくない事が伺える．

Dice クラスの API 仕様	
Dice	コンストラクタです．
Cast	サイコロを振ります
getValue	サイコロの目の値を取得します．

図 6 Dice クラスの API 仕様

Cup
arrayList: ArrayList<Dice> = new ArrayList<Dice>()
Cup( ) add(dice:Dice) : void get(number:int) : Dice size( ) : int cast( ) : void getSum( ) : int getValue(number:int) : int

図 7 Cup のクラス図

Cup クラスの API 仕様	
Cup	コンストラクタです．
add	カップ(Cup オブジェクト)に引数で指定された Dice オブジェクトを入れます．
get	カップに入っている Dice オブジェクトを返却します．ただし，引数は入れた順番です．順番は 0 から始まります．
size	カップの中に入っている Dice オブジェクトの個数を返却します．
cast	カップの中に入っている Dice オブジェクト全てを振ります．
getSum	カップに入っている Dice オブジェクトの目の合計を返却します．
getValue	カップに入っている Dice オブジェクトの目を返却します．引数は入れた順番です．

図 8 Cup クラスの API 仕様

5.2 ArrayListの活用

Diceオブジェクトが少ない場合は，オブジェクトを参照するための変数を個々に用意し，これを利用しながらプログラムを作れる．ところが，Diceオブジェクトの数が10個や100個に増えると，この考え方の延長線上でプログラムを作ることはできない．

7回目の授業は，複数のDiceオブジェクトを扱うために，コレクションクラスの一つであるArrayListオブジェクトを活用する．図3から分かるように，約80%の理解度が得られている．ArrayListクラス，Diceクラス，Randomクラスの関係を図9に示す．複数のクラスが関係すると，複雑さが急に増す．

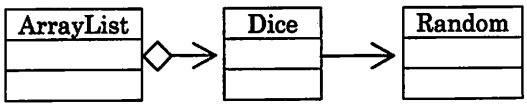

図 9 クラス図の関係

5.3 Cupオブジェクトの作成

Dice オブジェクトをカップ(Cup)の中に入れ，カップを振るとサイコロを振ることができる．その仕組みは，Cup オブジェクトの中に，7回目の授業で学んだArrayList オブジェクトを状態として持ち，これを活用するものである．図3から分かるように，理解度は約65%に急落し，8回目の授業に初学者のつまずきが存在する．Cupクラスの関係は図10に示す．

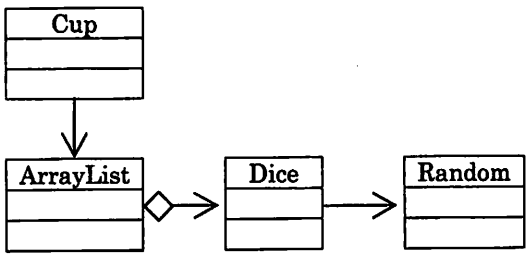

図 10 クラス図の関係

5.4 初学者のオブジェクト指向の難しさ

Cup オブジェクトは，複数のオブジェクトを活用しながら機能するが，手続き型の考え方に慣れている受講者たちは，オブジェクトの中に存在するオブジェクトを，外部から直接操作しようと考えてしまう．Cup オブジェクトを利用する立場とCupオブジェクトの仕組みを作る立場の区別が重要であり，どちらの立場でプログラムを組まなければいけないのか，その区別が曖昧な事も分かった．理解度調査結果から，初学者がオブジェクト指向を学習することの難しさの本質がここにあると考えられる．

6. 難しさを克服するための工夫

2011年度は，理解度を向上するために以下の工夫を強化した．代表的なものを紹介する．

6.1 実物の活用

Diceオブジェクトの実物の例を，図11に示す．さらにカップの実物の例を図12に示す．このように実物を見せながら．オブジェクトの関係を示す工夫を取り入れている．実物を見せることによって，この中に状態や振る舞いが存在していることを強調している．

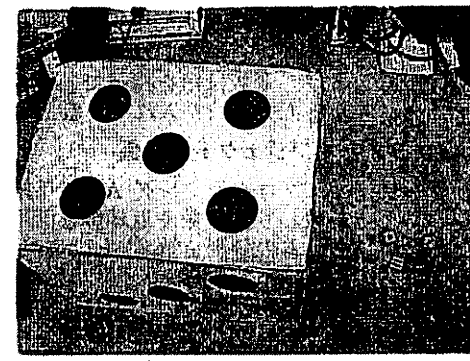

図 11 Diceオブジェクトの例

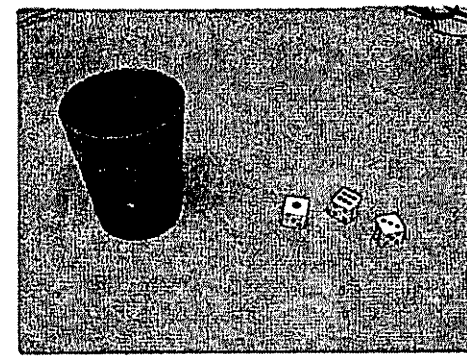

図 12 Cup と Diceオブジェクトの例

6.2 専門用語との格闘

初学者にとってオブジェクト指向の難しさは，カタカナの専門用語に遭遇することが挙げられる．一例であるが，クラス，オブジェクト，インスタンス，メソッド，フィールド，コレクションクラス，インタフェース，オー

バーライド等々，言葉の意味を理解することは，大変である．また，オブジェクトやインスタンスは同じ事を意味しており，別な言い方が存在するために，混乱が増す．授業の初期の頃は，授業の進行速度を極力抑え，専門用語の意味が理解されるように，専門用語を反復して取り入れるように工夫している．

6.3 オブジェクトの変数への代入

代入は，右辺の値を左辺の変数に入れることであるが，オブジェクトを代入することのイメージが，初学者にとって難しい．先に学ぶ「コンピュータプログラミングA」の授業で代入を学ぶが，この時は基本データ型が中心となり，変数は箱やコップ等で表現し，その中に値が直接格納されることをイメージさせている．ところが，オブジェクトの代入はこれでは上手く行かない．

一例であるが，風船をふくらませた後，紐を付けて持っていないと，風船はどこかへ飛んで行ってしまう．変数は紐を持っている事をイメージさせる工夫を取り入れている．

6.4 クラスとオブジェクトの関係

「コンピュータプログラミング A」で手続き型の考え方を教わり，引数や返却値を活用しながらプログラムの分割を学ぶ．ところが，この事が頭の中に残っているため，オブジェクトの理解が難しい．受講者の大半は，クラス自身がそのままプログラムとして実行される錯覚に陥る．new によって生成されたオブジェクトの中に，状態や振る舞いが存在しており，そのオブジェクトの中に存在している振舞いを活用しながら，プログラムが動作している．図 11 に示したサイコロ等の実物を見せながら，その動作を説明する工夫を取り入れることによりイメージされ易くなった．

6.5 ArrayList の説明

ArraListオブジェクトは，配列のように，あらかじめ要素の数を定義する必要が無く，汎用性が高い．add，get，size，remove などが代表的な振舞いである．一例であるが，ArraListオブジェクトに見立てたふた付きの箱は，ふたを閉めると中が見えなくなり，開けると中の様子が見える．さらに，この箱が入る程度の空き缶を活用し，オブジェクトの中にオブジェクトが存在していることを見せながらCupオブジェクトの説明を工夫している．受講者からは分かりやすいとの感想が寄せられるようになった．この様子を図 13 に示す．

6.6 作る立場と利用する立場

既存のオブジェクトを活用してさらにオブジェクトを作る場合，すなわちクラス拡張を行う場合は，オブジェクトの中身を作ることと，そのオブジェクトを活用する事が交錯する．受講者の中には，オブジェクトの中に存在する別なオブジェクトの状態や振る舞いを，外部から直接アクセスしようと考える者もいる．作る立場と利用する立場があることを，図 13 に示した小道具を活用しながら説明している．

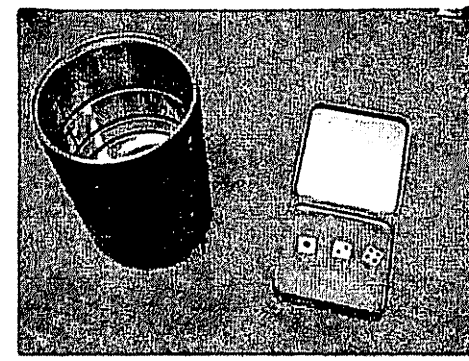

図 13 ふた付きの箱の例

6.7 インタフェース

8 回目の授業までには出てこない内容であるが，初学者にとってインタフェースの理解は難しい．オブジェクトはクラスから生成されるが，クラス名が異なると，たとえ中の仕組みが同じでも，型が異なるオブジェクトとして認識される．したがって，型をイメージさせることが重要である．

一例であるが，クラス名が異なることはケーブルのコネクタの形状が異なることと説明している．数種類の実物のケーブルを授業の中で見せ，形状が異なると接続できないことを説明する．共通の仕様である USB ケーブルを使うと，色々な周辺装置とパソコンを容易に接続できることを説明すると，理解され易い．この様子を図 14 に示す．

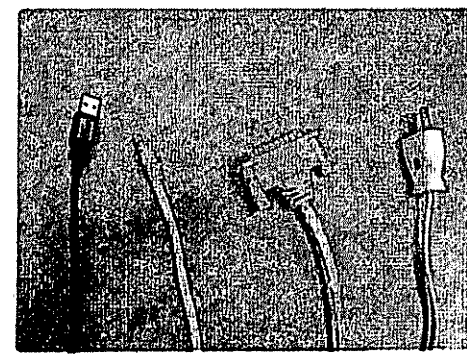

図 14 異なる形状のケーブルの例

6.8 クラス図とソースプログラムの関係

クラス図が定義されると，ソースプログラムの骨格は機械的に導出できる．初学者にとって，テキストエディタを使って説明するよりも，紙とペンでクラス図を模写し，その図を見ながら手書きでソースプログラムを記述することが，理解されやすい．クラス図のどの部分が，ソースプログラムと対応しているかが良く理解され，効果的である．慣れて来るとクラス図を見ながらテキストエディタを使ってプログラムを入力できるようになる．ペンタブレットを使うようになり，今入力している部分が，クラス図のどこに対応しているかを，適宜説明しながら授業を進めており効果的である[5].

6.9 ブロック構造を意識した入力

クラスの中に存在するメソッドの数が増えて来ると，ブロックを意識したインデントが重要な意味を持つようになる．うっかり，中括弧を閉じ忘れたりすると，膨大な数のコンパイルエラーを誘発する．「コンピュータプログラミング A」の授業では，プログラムの入力方法について作法を守るように指導しているが，その重要さがここで再認識される．

6.10 名前の付け方

クラス名の先頭は大文字，変数名の先頭は小文字，途中，単語が出現する場合は，最初の文字を大文字にして連結する等，重要な作法が沢山存在する．また，振舞いの最初の単語は動詞を置き，小文字で始める．このような作法の説明は重要であり，スクラッチプログラミングを実践する中で，適宜，補足説明を行っている．初学者は，コンパイルエラーが無くなれば正しいと理解しがちであるが，そこに潜む，作法の問題は，多くのプログラムを入力し，体験しないと身に付かない．

6.11 for 文の制御変数は 0 を初期値に

for 文の制御変数の初期値をいくつにするかは，プログラムを作る人の自由であるが，0 を入れることを指導している．日常生活では，数を数える時には 1 から数えることが一般的であるが，コンピュータの世界では 0 がベースになっており，0 から数えることを説明している．数を減らす場合は，繰り返す仕組みを確実に完成し，繰り返しの中で必要な数を生成することを指導を工夫している．

7. 理解度分析結果と考察

7.1 理解度の推移

先に述べた 2010 年度の理解度の推移に加えて，2011 年度の推移を図 15 に示した．2011 年度の理解度は 2010 年度と比較し，全体的に向上し，工夫を強化した効果と考えられる．

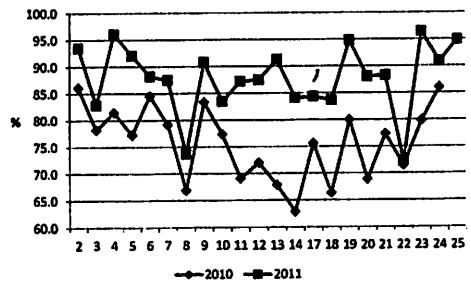

図 15 授業毎の理解度（各年度）

7.2 理解度パターンの表現

理解度をパターン化して表現するために，以下の方法を採用した．各受講者について，ある回の授業の理解度が，「はい」の総数が「いいえ」よりも多かった場合は，その回の授業は理解できたものと判断し Y を割り当てる．逆に「いいえ」が多かった場合は，その授業が十分に理解できていないと判断しNを割り当てる．同数の場合は D を，欠席は#を割り当て，これらを順に連結しパターン化した．

なお，1 回目の授業はガイダンスのためアンケート調査をしていない．したがって理解度の分析は，2 回目から 8 回目の計 7 文字となる．例えば，#YYYDNN の理解度パターンは，2 回目を欠席し，その後 3 回連続して理解できており，6 回目の授業で「はい」と「いいえ」が同数になり，その後，2 回連続して理解できなくなった事を示す．

7.3 理解度パターンの分析

理解度パターンの分析を行うために，8 回目の授業をYと回答した母集団とNと回答した母集団に分け，8 回までの各文字の出現頻度を探った．この結果を，表 3 と表 4 に示す．

表 3 8回目をYと回答(2010 年度)

2010	Y	N	D	#	合計
人数	243	12	2	6	263
割合	92.4	4.6	0.8	2.3	100.0

表 4 8回目をYと回答(2011 年度)

2011	Y	N	D	#	合計
人数	249	1	1	8	259
割合	96.1	0.4	0.4	3.1	100.0

2010 年度は，8 回目の授業で Y と回答した母集団の Y の比率は 92.4%, N の比率は 4.6% であった．2011 年度は，8 回目の授業で Y と回答した集団の Y の比率は 96.1%，N の比率は，わずかに 0.4%であった．この事は，両方の年度を通じて 8 回目の授業内容を理解できている受講者は，それ以前の授業も良く理解できていることを示している．特に，2011 年度は，2010 年度と比較し，N と回答する比率が低かった．2011 年度は，先に述べた工夫を強化した効果が現れていると考えられる．

一方，8 回目の授業を N と回答した母集団の各文字の出現頻度を探った．この結果を，表 5 と表 6 に示す．2010 年度は，Y が 58.6% あり，N が 40.2%であった．2011 年度は，Y が 56.0%あり，N が 31.0%であった．2010 年度の方が N の割合が高くなり，2011 年度と比較して理解が得られていない事がうかがえる．この事は，両方の年度を通じて 8 回目の授業内容を理解できていない受講者は，それ以前の授業内容も，良く理解できていないことを示している．特に，2010 年度は，2011 年度と比較し，N と回答する比率が高いことも分かった．2011 年度は，先に述べた工夫の効果が現れていると考えられる．

表 5 8 回目を N と回答(2010 年度)

	Y	N	D	#	合計
人数	51	35	1	0	87
割合	58.6	40.2	1.1	0	100.0

表 6 8 回目を N と回答(2011 年度)

	Y	N	D	#	合計
人数	47	26	9	2	84
割合	56.0	31.0	10.7	2.4	100.0

表 7 8 回目を N と回答した理解度パターン

2010 年度	頻度	2011 年度	頻度
YYYYYYN	6	YYYYYYN	3
NNNNNNN	2	#NYYNNN	1
YNNNNNN	1	#YYY#NN	1
YNNNYYN	1	#YYYDNN	1
NDYYYNN	1	#YYYY#N	1
YYYDNNN	1	NNNDDNN	1
YNYYYYN	1	YNYVYDN	1
		YNYYYDN	1
		YNYYDYN	1
		YDYNNYN	1
		YDYDDYN	1

7.4 理解度パターンの頻度を分析

8 回目の授業で N と回答した受講者の理解度パターンの頻度に着目した．その結果を表 7 に示す．2010 年度と 2011 年度共に，YYYYYYN の理解度パターンが最も多く，6 名と 3 名であった．その他のパターンは 1 名程度であるが，8 回目の授業内容を理解できない受講者は，それ以前の授業内容も良く理解できていないことが，一層明確になった．

8. まとめ

オブジェクト指向の入門を学ぶ「コンピュータプログラミング B」の授業を対象に，初学者が授業のどこでつまずき，何が本質的な原因であるかを探った．その結果，ArrayList オブジェクトを活用した新しいオブジェクトを作成することにつまずくことが分かった．2011 年度は工夫を強化したことが，受講者の理解度全体への効果に繋がったと考えられるが，個々の工夫がどの程度効果をもたらしたかについては，十分に分析できていない．

オブジェクトの説明に実物を活用することによって受講者はイメージし易くなるが，複数のオブジェクトが関係して動作するプログラムの理解は，やはり難しい．実物を見ることによって，オブジェクトの関係は分かるものの，プログラムを記述する段階で混乱している様子がうかがえる．今後はオブジェクト指向の鬼門を一人でも多くの受講者が乗り越えられるよう，授業の理解度のモニタリングを継続しながら，授業の工夫に取り組みたい．

参考文献

[1] 土肥紳一，宮川　治，今野紀子：プログラミング入門教育におけるモチベーションと成績の関係，情報教育シンポジウム SSS2011 論文集，Vol.2011，no.4，pp.141-146(2011).

[2] 久野　靖，辰己丈夫，中野由章他：情報科教育法改訂 2 版，オーム社(2009)

[3] 2011 年度コンピュータプログラミング B 講義ノート http://www2.dcl.sie.dendai.ac.jp/dohi/2011/proB/

[4] 土肥紳一，宮川　治，今野紀子：SIEM を活用したプログラミング入門教育のための授業コンテンツ，PC カンファレンス講演論文集，pp.115-118 (2006)．

[5] 土肥紳一，宮川　治，今野紀子：SIEM 導入したプログラミング入門教育におけるペンタブレットの活用，大学 ICT 推進協議会，年次大会講演会講演論文集 p399-p406(2011).