Processing によるオブジェクト指向プログラミング入門教育の実践

土肥紳一[†1]

概要：2017 年 4 月にシステムデザイン工学部が東京千住キャンパスに開設した．システムデザイン工学部は情報環境学部が改組され，情報システム工学科とデザイン工学科から構成されている．筆者が所属するデザイン工学科では，プログラミング言語に Processing を導入した．Processing はスケッチブックに絵を描く感覚でプログラムできる特徴があり，デザイン工学科のプログラム言語として最適であった．長年，情報環境学部で Java 言語を教えてきたノウハウを活かしながら，オブジェクト指向の入門を指導している．情報環境学部の受講者はコンピュータ，プログラミングそしてネットワークに興味を持っていた．しかしながら，デザイン工学科の受講者はプロダクトデザインに興味を持っている．本論文では，教室環境や受講者が大きく変化する中，プログラミング教育の工夫について述べる．

キーワード：オブジェクト指向，教室環境，プログラミング教育，モチベーション

Practical Object Oriented Computer Programming Education by using Processing Language

SHINICHI DOHI[†1]

Abstract:
School of System Design and Technology was opened at Tokyo Senju campus in April 2017. School of System Design and Technology has reorganized School of Information Environment, it consists of Department of Information System Engineering and Department of Design Engineering and Technology. In the Department of Design Engineering to which the author belongs, Processing was introduced into the programming language. Processing has a feature which can be programmed with the sense of drawing a picture in a sketchbook, and it was the best as a programming language of the Department of Design Engineering. For many years, while taking advantage of know-how that I taught Java language at the School of Information Environment, I teach introduction of object oriented programming. While taking advantage of know-how that I taught Java language at the School of Information Environment for many years, I teach introduction of object oriented programming. Students of the School of Information Environment were interested in computers, programming and networks. However, students of the Department of Design Engineering and Technology are interested in product design. In this paper, I describe the idea of programming education while the classroom environment and students are changing greatly.

Keywords: Object Oriented, Classroom Environment, Programming education, Motivation

1. はじめに

2001 年 4 月に千葉県印西市の千葉ニュータウンキャンパスに開講した情報環境学部(定員 180 名)は，情報環境工学科(定員 95 名)と情報環境デザイン学科(定員 85 名)の 2 学科で始まった．2001 年は IT バブルが崩壊した年であり，情報環境学部にとって厳しい門出となった．情報環境学部のプログラミング入門教育は，当初，C 言語(2 クラス)と Java 言語(1 クラス)を選べるように 3 クラスに分割し，授業を開始した．その後，新入生が最初に学ぶ言語として Java が適切であると判断し，2005 年から Java 言語に統一し，4 クラス分割で授業を実施するようになった．2006 年から 2 学科を情報環境学科に統一し，ネットワーク・コンピュータ工学コース，先端システム設計コース，メディア・人間環境デザインコースの 3 コース制とした．2008 年から 240 名に定員増を行い，コミュニケーションデザインコースを加えた 4 コース制になり，Java を学習する学生の数が大幅に増えた．

近年，ビッグデータやディープラーニング等が注目を浴びるようになり，情報環境学部は 2016 年 4 月の入学者を最後に募集を停止し，2017 年 4 月から東京千住キャンパスにシステムデザイン工学部を開設した．この学部は，情報システム工学科(定員 130 名)とデザイン工学科(定員 110 名)の 2 学科である．筆者はデザイン工学科に所属し，プログラミング言語は Processing を採用することになった．担当科目は，手続き型の入門を学習する「コンピュータプログラミングⅠ」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Ⅱ」である．本論文では，情報環境学部で培ったノウハウを活用しながら，「コンピュータプログラミングⅡ」の実践における工夫について述べる．

†1 東京電機大学システムデザイン工学部
Tokyo Denki University School of System Design and Technology

© 2018 Information Processing Society of Japan

2. 学習環境の変化

2.1 千葉ニュータウンキャンパスの教室

情報環境学部では，図 1 に示す様にグループ学習が行い易い教室を使った．各机には電源と情報コンセントが付いた装置が設置されている．その他，無線 LAN が使える．プログラミングの授業では，受講者の顔よりも，ノート PC の画面が見えることが重要であり，受講者が何に躓いているのか，直ぐに分かる背面指導のし易い教室であった．教室のレイアウトは図 2 に示す．赤い矢印は図 1 の撮影方向である．各テーブルは可動式で，8 名着席できる机が 8 台用意されており，64 名を収容できる．座席は自由席とした．原則として TA(Teaching Assistant)が 2 名，SA(Student Assistant)が 2 名付いた．授業は 4 名の教授者でクラス分割を行い，1 クラスは約 60 名である．

この教室は断熱性に乏しい教室で，空調を入れても夏場は 30 度近くまで室温が上昇し，冬場は 20 度程度までしか気温が上がらない．また，狭い教室で，ほぼ，定員の受講者が授業を受けるため，教室内の空気が悪いことに気付いた．教室内の二酸化炭素濃度，気温，湿度等を測定し，その空間分布を表示する等，分析を行った[1].

図 1 千葉ニュータウンキャンパスの教室の様子

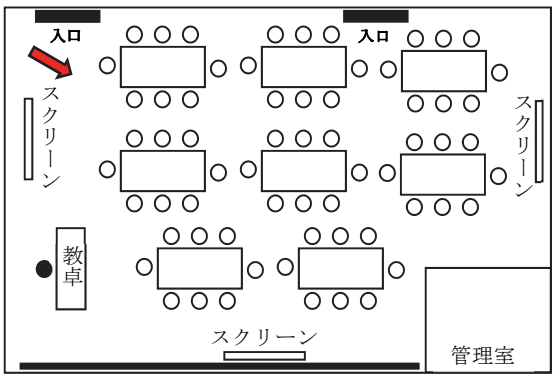

図 2 千葉ニュータウンキャンパスの教室のレイアウト

2.2 東京千住キャンパスの教室

一方，東京千住キャンパスは，縦長の講義室に変わった．この様子を図 3 に示す．各机の前方には AC コンセントと情報コンセントが埋め込まれている．さらに無線 LAN を利用できる．残念ながら，教卓からは受講者のノート PC の画面を見ることができない．また，机間指導を行うにも，教卓から教室の後方までは，15m 近い距離があり，図 1 と比較して，机間指導が難しい．教室のレイアウトの様子は図 4 に示す．赤い矢印は図 3 の撮影方向である．

教卓側にプロジェクターのスクリーンが 2 面，天井からモニターが 1 箇所付いている．机は固定式の 3 人掛けのものが 4 列，さらに教卓から教室後方に向かって 15 列並んでいる．教室の定員は，180 名である．座席は自由席とした．原則として TA が 2 名付き SA は居ない．授業は 2 名の教授者でクラス分割を行い，1 クラスの受講者は約 60 名である．教室の後方からプロジェクターのスクリーンやモニターは文字の判読が難しいため，前の方へ来るよう指導している．教室の窓は開閉できない構造になっており，完全空調である．広い教室内に，定員の 1/3 程度の受講者しか居ないため，教室内の空気は良好である．

図 3 東京千住キャンパスの教室の様子

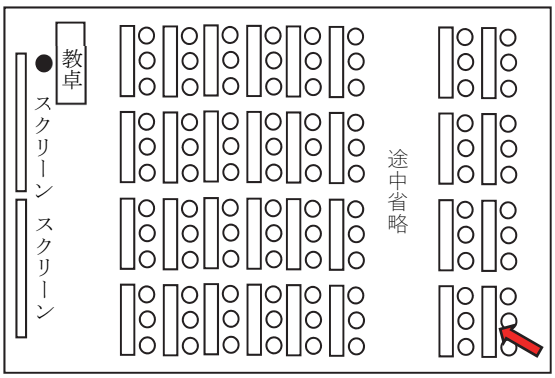

図 4 東京千住キャンパスの教室のレイアウト

© 2018 Information Processing Society of Japan

3. コンピュータプログラミングの授業内容

3.1 Processing 導入の経緯

デザイン工学科は，工学の基礎を学び，デザインと結び付け，物づくりを目指す学科である．このような学科にとって，適切なプログラミング言語は何かを探ったところ，他大学の導入例も参考にしながら Processing に辿り着いた[2]．元々，Processing は，スケッチブックに絵を描く感覚でプログラミングできるように設計されており，デザイン工学科のプログラミング教育に最適であった．

筆者が 2013 年から 2016 年に担当した高大連携型教育の 1 コマの中で,「プログラミング入門体験」も決め手になった．近隣の高校生に土曜日，千葉ニュータウンキャンパスの情報環境学部へ来てもらい，10:00～12:00 の 2 時間，講義を受ける内容である．ノート PC を原則として 1 人 1 台準備し，体験してもらった．Processing によるプログラミングは，タイピング量は少ないものの，グラフィカルに結果が表示されることに，高校生の興味を喚起できた[3].

さらにデザイン工学科に所属する研究室いくつかは，Processing を使ったアプリケーションの開発を行っており，研究への継続性の点でも有利であった．Processing は Java で記述されており，Java 本来のオブジェクトを活用でき，前述した通り，情報環境学部で培ったノウハウが継承できるメリットがあった．

3.2 授業の実施形態

BYOD(Bring Your Own Device)によって，学生が持参したノート PC に Processing 3.3.7 をインストールし，授業で使っている．ノート PC の OS は，大半が Windows 10 であり，少数であるが Mac OS を使っている人がいる．授業の 1 コマは, 2018 年 4 月から 1 コマ 100 分授業になり，半期 15 回から 14 回に減った．これはハッピーマンデー対策である．大学の行事が色々増える中で，半期 15 回の授業を開講するために，昨年までは，休日や祭日を潰して授業を行ってきた.「コンピュータプログラミングⅡ」の授業は，月曜日の 3 時限(13:40～15:20)に実施している．情報環境学部では 1 コマ 50 分，2 コマ連続で途中 10 分の休憩を入れ，休憩を含め計 110 分の授業を実施してきた．教授者にとって 1 コマ 100 分授業への移行はスムーズであった．一方，学生にとっては 90 分から 100 分に伸びたことにより，帰宅時間が遅くなったとの声が出始めている．

3.3 教科書

教科書は，オライリーの「Processing をはじめよう 第 2 版」を使っている[5]．巻末にはクイックリファレンスも付いており，重宝している．この書籍は，1 年生後期に「コンピュータプログラミングⅠ」で，1 章～9 章，11 章配列(オブジェクトを除く)までを学ぶ. 残りの章を，コンピュータプログラミングⅡで学習する．この書籍の「はじめに」には，「『Processing をはじめよう』は教科書ではありません」との記載がある．簡潔な例題と説明が，初学者にとって学びやすい構成になっているが，説明不足な点もある．補足資料を適宜準備し，練習問題も準備することで対応している．教科書は 13 章まで掲載されているが，情報環境学部での経験から，コレクションクラス(ArrayList，HashMap)，インタフェース，継承を理解することが重要と考え，14 章から 17 章を独自に執筆した．授業では，第 11 章まで学んだ後に, 14 章から 17 章を先に学び, その後，12 章と 13 章に戻る流れを組んだ．教科書の章立ては，表 1 に示す．なお，筆者にて執筆した章は，※で示した．

表 1　教科書の章立て

はじめに	11 章 配列
1 章 ようこそ Processing へ	12 章 データ
2 章 コードを書いてみよう	13 章 拡張
3 章 描く	14 章 インタフェース(※)
4 章 変数	15 章 ArrayList オブジェクト(※)
5 章 反応	16 章 HashMap オブジェクト(※)
6 章 移動，回転，伸縮	17 章 継承(※)
7 章 メディア	付録 A　コーディングの心得
8 章 動き	付録 B　デバッガ
9 章 関数	Processing クイックリファレンス
10 章 オブジェクト	

3.4 授業実施スケジュール

授業の実施回数毎に内容を示したものを表 2 に示す．本論文の原稿執筆時点で,「8 回　中間試験」が進行している．1 回と 2 回で実施する「10 章　オブジェクト」は，専門用語がたくさん出てくる. 初学者にとって最初の関門である．特に変数にオブジェクトを代入する意味，すなわち参照の概念が重要である．3 回で実施する「11 章　配列」は,「コンピュータプログラミングⅠ」で学ばなかったオブジェクトの配列を取り上げる．参照の概念を正しく理解していないとオブジェクトの配列は理解できない．4 回で実施する「14 章　インタフェース」は，異なるオブジェクトを同一視するために必要な概念である．5 回で実施する「15 章 ArrayList」オブジェクトは, 要素数を気にしないで配列の様にオブジェクトを管理できる．6 回で実施する「16 章 HashMap」オブジェクトは, オブジェクトをキーで管理できる．7 回で実施する「17 章　継承」は，親子関係を説明する．中間試験以降は，再び，教科書に戻る．9 回と 10 回で実施する「12 章　データ」は，公開されているデータを可視化する．11 回と 12 回で実施する「13 章　拡張」は，オブジェクト指向のクラス拡張ではなく，音や画像，Arduino 等の活用が述べられている. Arduino については，教授者がデモを見せる程度になる予定である．13 回は，プログラミングコンテストを予定している．テーマは自由テーマとし，1 分／人程度の時間で全員に発表を行ってもらい，採点は Excel のワークシートを配布し，学生諸君による 5 段階で相互評価を行う．コンテスト終了後，集計を行い，ベスト 3 を発表する．14 回は，期末試験を実施する．成績は, 授業時の成果物 30%, 学力考査 70%で評価する．

© 2018 Information Processing Society of Japan

表 2　授業実施スケジュール

	内容
1 回	ガイダンス，10 章 オブジェクトオブジェクトを利用するためには，フィールド，メソッドの概念を理解し，クラスを定義する事を学びます．専門用語(フィールド，メソッド，インスタンス，コンストラクタ，オブジェクトの生成等)について学びます．
2 回	10 章 オブジェクト複数のオブジェクトを活用することについて学びます．
3 回	11 章 配列 複数のオブジェクトを扱うために，オブジェクトの 1 次元配列の活用について学習します．一例として，JitterBug オブジェクトを複数生成し表示します．その他，PImage オブジェクトを取り上げ，パラパラ漫画の要領で複数の画像を表示する例を取り上げ，オブジェクトの配列について学習します．
4 回	14 章 インタフェース インタフェースを定義することによって，異なるオブジェクトを同一視できるようになります．前回取り上げた，1 次元配列にインタフェースを取り入れることによって，異なるオブジェクトを配列で利用できるようにします．
5 回	15 章 ArrayList オブジェクト 配列は複数のオブジェクトを管理する上で，便利なオブジェクトです．しかし，あらかじめ配列の大きさを決めておかなければならない欠点があります．ArrayList オブジェクトは，add，get，remove 等のメソッドを使うことによってオブジェクトを管理でき，管理するオブジェクトの個数を気する必要がなくなります．
6 回	16 章 HashMap オブジェクト 配列や ArrayList オブジェクトは，複数のオブジェクトを扱う上で便利です．しかし，配列の場合は添字，ArrayList オブジェクトの場合はオブジェクトを入れた順番で管理する必要があります．HashMap オブジェクトは，put，get 等のメソッドを使い，順番を意識することなくオブジェクトをキーで管理できるようになります．
7 回	17 章 継承 親クラス（スーパークラス），子クラス（サブクラス）について学習します．
8 回	中間試験
9 回	12 章 データ データビジュアライゼーションに必要なデータセットの扱い方を学習します．CSV 形式のデータを Table オブジェクトに入力し，折れ線グラフの表示を行い，これに必要なオブジェクトについて学習します．
10 回	12 章 データ JSON 形式のデータを JSONObject オブジェクトに入力し，その内容を表示することを学習します．これに必要なオブジェクトの扱いを学習します．
11 回	13 章 拡張 PDF 形式のファイルの出力，サウンドの再生を学習します．これに必要なオブジェクトの扱いを学習します．
12 回	13 章 拡張 Arduino との連携を学習します．
13 回	プログラムコンテスト：これまで学習してきた内容を基に，プログラムコンテストを実施します．
14 回	期末試験

4. 授業の工夫と実践結果

4.1 専門用語を理解させる工夫

オブジェクト指向には，初学者が最初に遭遇する関門がある．それは，オブジェクト指向固有の専門用語との格闘である．情報環境学部の時代に，この点は十分承知しており，専門用語が意味するものは何かを，具体的にイメージさせる小道具を活用しながら授業を行っている．図 5 はクラスとオブジェクトの違いを理解させるために利用した．SSS2015 でも紹介したものである．粘土の型がクラスであり，型に粘土を流し込んで取り出したものがオブジェクトであることを説明する．授業アンケートの自由記述の中に「例えは理解しやすかったが実際どのように使えばいいのかがわからない」との指摘があった．この方法は，クラスとオブジェクトをイメージさせ易いが，プログラムを書くことは初学者にとって難しいことが窺がえる．つまり，プログラムを入力する段階になると，クラスとオブジェクトの関係が混乱する．授業毎に，何度も説明を繰返しながら指導することになる．

図 5 付箋，紐，型と粘土による補足説明の小道具

クラスの定義の中には，フィールドが存在する．このフィールドは，状態，インスタンス変数等，別な言い方が存在し，初学者にますます混乱を与える．同様に，メソッドもそうである．振舞いや関数などの表現がある．オブジェクトを生成する時に実行される特別なメソッドがコンストラクタであるが，初学者に混乱を与える部分である．コンストラクタに仮引数があると，仮引数とフィールドの関係が混乱を招く．粘土を使ってイメージさせることは容易であるが，プログラムとして記述できるようになるまでは，専門用語を意識させながら，授業は特にゆっくり進めるように注意している．授業内容および授業毎のアンケートによる理解度調査結果は，web サイトにて公開している[4].

4.2 オブジェクトを理解させる工夫

最初に出てくるオブジェクトは，第 10 章の JitterBug クラスである．〇が乱数で上下左右に少しずつ移動し，その軌跡を残すことで，JitterBug(じれったい虫)らしさを表現している．ソースプログラムと実行例を図 6 に示す．ソースプログラムは，教科書の p149 の Example 10-1 を引用した．実行結果は，グラフィカルに表示されるため，オブジェクトの様子を理解させ易い．Java でオブジェクト指向を教えた経験では，生成したオブジェクトを活用するプログラムを実行しても，ゲッターによって得られた情報をテキストで表示することが常であった．オブジェクトの状況を頭の中で描きながら理解させる必要があったが，Processing はオブジェクトの反応をグラフィカルに表現できるため，受講者にイメージさせ易い．JitterBug オブジェクトが少しずつ動く様子は，オブジェクトが実装している move メソッドによって JitterBug オブジェクトのフィールド x と y が乱数によって少しずつ変化し，その座標に〇を表示することによって動いて見えるという説明が，イメージと一致する．サンプルコードの使用については，

© 2018 Information Processing Society of Japan

「一般に，本書に掲載しているコードは各自のプログラムやドキュメントに使用してかまいません．コードの大部分を転載する場合を除き，私たちに許可を求める必要はありません.」との記載（抜粋）に基づき，活用している．

```
JitterBug bug;

void setup() {
  size (480, 120);
  bug = new JitterBug(width * 0.66, height / 2, 10);
}

void draw() {
  bug.move();
  bug.display();
}
```

```
class JitterBug {
  float x;
  float y;
  int diameter;
  float speed = 2.5;

  JitterBug(float tempX, float tempY, int tempDiameter) {
    x = tempX;
    y = tempY;
    diameter = tempDiameter;
  }

  void move() {
    x += random(-speed, speed);
    y += random(-speed, speed);
  }

  void display() {
    ellipse(x, y, diameter, diameter);
  }
}
```

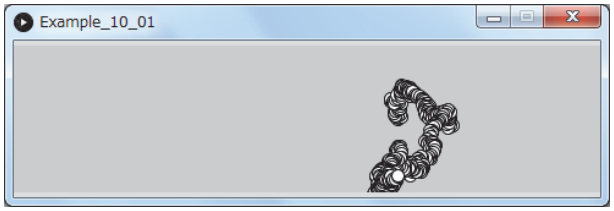

図 6 JitterBug オブジェクトの実行例

4.3 参照を理解させる工夫

「コンピュータプログラミング I 」で，int 型や float 型など，基本データ型の変数を学んできた受講者にとって，変数にオブジェクトを代入することの理解は，難易度が高い．変数自体にオブジェクトが格納されるように誤解しがちである．生成されたオブジェクトへの参照が変数に格納されることを理解しないと，今後の学習は難しい．Java で教えた経験から，付箋と紐を使って説明すると理解され易い．付箋が変数，紐が参照を意味しており，複数の変数にオブジェクトを代入した場合は，複数の変数がオブジェクトを参照していることを見せる．

9 章でフクロウを表示する Owl 関数が取り上げられている．授業では 10 章のオブジェクトの中で，Owl クラスを独自に定義し，Owl オブジェクトを生成し，Owl 関数と同様にフクロウを表示できることを学ばせた．さらに複数の Owl オブジェクトを生成する例として，複数の Owl 型の変数 owl0，owl1, owl2 を宣言し，複数の Owl オブジェクトを表示するプログラムと実行例を図 7 に示す．

図 7 に示した複数の Owl オブジェクトの関係は，付箋と紐で説明した．この様子を図 8 に示す．左側の付箋 owl0 ，owl1，owl2，temp は変数に見立て，各変数には Owl オブジェクトへの参照が入っていることを紐で示す．特に owl2 と temp は同じ Owl オブジェクトを参照している．身近な文具を活用した工夫であるが，参照をイメージさせ易い．

```
Owl owl0;
Owl owl1;
Owl owl2;
```

```
class Owl{
  int x;
  int y;
```

```
Owl temp;

void setup(){
  size(480, 120);
  background(176, 204, 226);
  owl0 = new Owl(110, 110);
  owl1 = new Owl(180, 110);
  owl2 = new Owl(250, 110);
  temp = owl2;
}

void draw(){
  owl0.display();
  owl1.display();
  temp.display();
}
```

```
  Owl(int tempX, int tempY){
    x = tempX;
    y = tempY;
  }
  void display(){
    pushMatrix();
    translate(x, y);
    stroke(138, 138, 125);
    strokeWeight(70);
    line(0, -35, 0, -65);
    noStroke();
    fill(255);
    ellipse(-17.5, -65, 35, 35);
    ellipse( 17.5, -65, 35, 35);
    arc(0, -65, 70, 70, 0, PI);
    fill(51, 51, 30);
    ellipse(-14, -65, 8, 8);
    ellipse( 14, -65, 8, 8);
    quad(0, -58, 4, -51, 0, -44, -4, -51);
    popMatrix();
  }
}
```

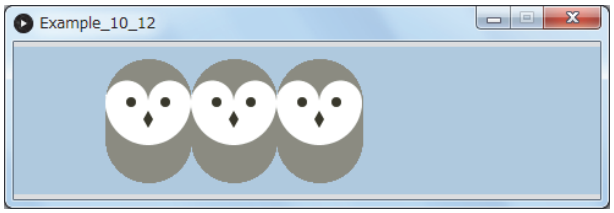

図 7 複数の Owl オブジェクトの表示

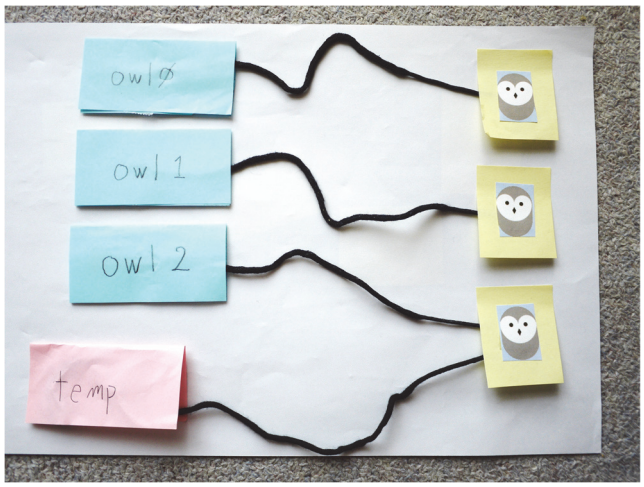

図 8 付箋と紐を使った Owl オブジェクトの参照の様子

4.4 インタフェースの概念を理解させる工夫

オブジェクトを参照する変数は，クラス名を変数の型に宣言する．したがって，クラス名だけが異なり，それ以外は全く同じ内容のクラスであっても，別なものとして扱われる．このことを理解させるために，Java で教えた経験から，変数の型は，色々な形状のコネクターが付いたケーブルを見せ，形状が異なると接続できないことを見せる．この様子を図 9 に示す．

インタフェースの概念は，初学者にとって理解が難しい内容の一つである．インタフェースは異なるオブジェクトを同一視するために欠かせない機能である．インタフェースの概念については，USB ケーブルを持ち出し，USB が付いているオブジェクトであれば，共通に接続できることを見せられる．Java で教えていた時代から取り入れている

© 2018 Information Processing Society of Japan

が，この方法は，インタフェースを理解させる良い方法である．授業アンケートの自由記述の中に「小道具を使った説明が分かりやすかったです」との指摘があり，受講者にインタフェースの概念をイメージさせ易いことが分かった．

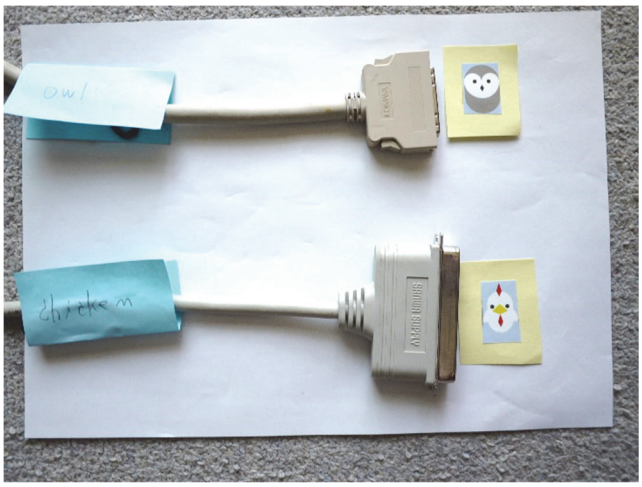

図 9 ケーブルを活用した変数の型の補足

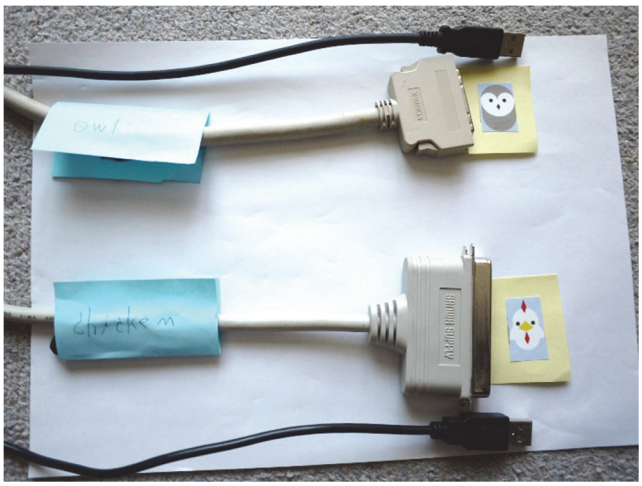

図 10 USB ケーブルを活用したインタフェースの補足

Processing では，Owl クラスと Chicken クラスを取り上げ，USB ケーブルに相当する Bird インタフェースを定義し，これを実装することで説明した．ニワトリは筆者が作成したものである．アンケート調査で，「Chicken オブジェクトは，可愛いと思いますか.」との設問に対して，48 名中 43 名(89.6%)が「はい」と回答した．表示されるオブジェクトに好感を持ってもらえた．USB ケーブルで例えた例を基に，Bird インタフェースの意味を理解させ，異なるオブジェクトであるが，同一視できることを学ぶ．さらにオブジェクトの配列で，Bird インタフェースを活用することによって，Owl と Chicken を交互に並べる例を示した．Owl型の配列では，Owlオブジェクトだけを管理できるが，Chicken オブジェクトは管理できない．一方，Chicken 型の配列では，Chicken オブジェクトだけを管理できるが，Owl オブジェクトは管理できない．二つのオブジェクトに共通な Bird インタフェースを実装することによって，Bird 型の配列で Owl オブジェクトと Chicken オブジェクトの両方を管理できるようになる．Bird インタフェースを実装したオブジェクトの実行例を図 11 に示す．Owl クラスは省略した．

```
Bird[] birds = new Bird[7];

void setup(){
  size(480, 120);
  background(176, 204, 226);
  for(int i = 0; i < birds.length; i++){
    if((i % 2) == 0){
      birds[i] = new Owl(35 + i * 70, 110);
    } else {
      birds[i] = new Chicken(35 + i * 70,
110);
    }
  }
}

void draw(){
  for(int i = 0; i < birds.length; i++){
    birds[i].display();
  }
}
```

```
interface Bird{
  void display();
}
```

```
class Chicken implements Bird{
  int x;
  int y;
  Chicken(int tempX, int tempY){
    x = tempX;
    y = tempY;
  }
  void display(){
    pushMatrix();
    translate(x, y);
    stroke(255);
    strokeWeight(50);
    line(0, -35, 0, -65);
    noStroke();
    fill(255);
    arc(0, -30, 70, 70, PI, TWO_PI);
    fill(51, 51, 30);
    ellipse(-15, -60, 8, 8);
    ellipse( 15, -60, 8, 8);
    fill(255, 255, 0);
    quad(0, -65, 15, -51, 0, -45, -15, -51);
    fill(255, 0, 0);
    quad(0, -105, 5, -90, 0, -80, -5, -90);
    quad(0, -45, 5, -30, 0, -20, -5, -30);
    popMatrix();
  }
}
```

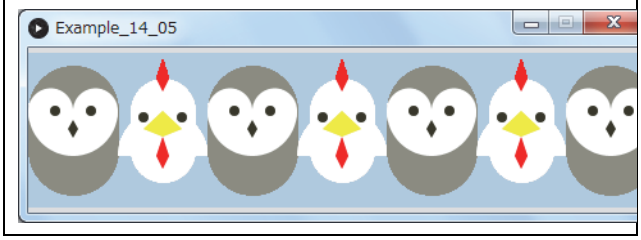

図 11 Bird インタフェースを実装したオブジェクト

物の例えであるが，Bird インタフェースの実装によって，鳥籠の中にフクロウとニワトリの両方が入れられるようになったことになる．現実に近い場面をグラフィカルに表現できるため，インタフェースの効果をイメージさせ易い．Java で教えていた頃と比較しても，非常に教え易く Processing で教えることの効果を実感している．

4.5 ArrayList オブジェクトを理解させる工夫

配列は，複数のオブジェクトを管理できるが，配列を宣言する時に，配列の大きさ(要素の最大値)を決めなければならない欠点がある．この大きさを超えると，プログラムは異常終了する．ArrayList オブジェクトは，大きさを決める必要が無い．オブジェクトを入れるときは，add メソッドを使う．最初に追加されたオブジェクトは 0 番目で区別される．さらに追加すると 1 番目で区別できる．オブジェクトを参照する場合は，get メソッドの引数に番号を指定すると，参照できる．オブジェクトの個数は，size メソッドで得られる．オブジェクトを削除したい場合は，remove メソッドの引数に番号を指定し，削除されたオブジェクトの参照が返却される．ArrayList オブジェクトは

© 2018 Information Processing Society of Japan

便利であるが，初学者にとって，難しい内容である．

ArrayList オブジェクトは 5 回目の授業で説明した．さらに 6 回目の授業の冒頭で補足説明を行った．補足説明で使用した資料は図 12 と図 13 である．小道具で複雑なオブジェクトの関係を見せることは既に限界に来ており，パワーポイントで複数のスライドを準備し，ソースプログラムとオブジェクトの関係をパラパラ漫画の要領で示した．

図 12 は Owl オブジェクトのみを管理する例の抜粋である．3 つのオブジェクトを生成し，for 文で ArrayList オブジェクトの中の 0 番目から順に Owl オブジェクトを表示する資料であり，変数 i が 2 の状態を示している．

図 13 は Bird インタフェースを実装したオブジェクトを ArrayList オブジェクトで管理する例の抜粋である．Owl オブジェクトと Chicken オブジェクトを交互に add するように for 文で記述した．

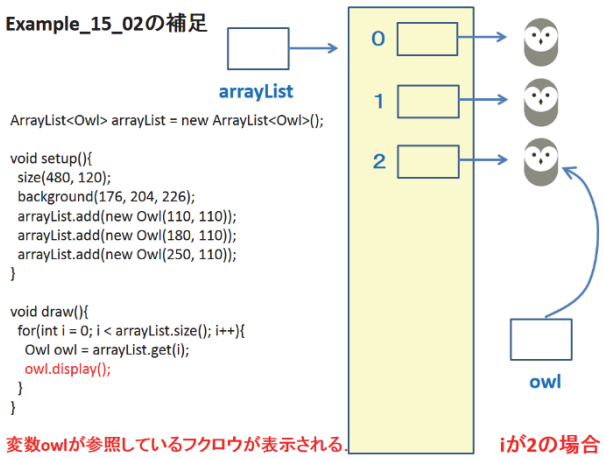

図 12 ArrayList オブジェクトの補足資料 1(抜粋)

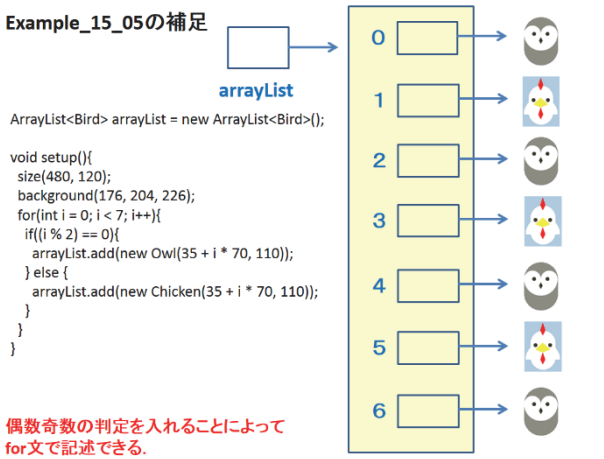

図 13 ArrayList オブジェクトの補足資料 2(抜粋)

補足資料の効果を調べるために，5 回目と 6 回目の同一調査項目に対して，理解できましたかの問いを設け，「はい」と回答した割合を図 14 に示した．図 14 では各調査項目に対して，「理解できましたか」の文言を省略している．ArrayList オブジェクトに関しては，get と add メソッドの理解が補足資料によって向上した．一方，配列と ArrayList オブジェクトの違いは，改善されていないことも分かった．80%の理解が得られるように，補足資料のさらなる工夫が必要である．

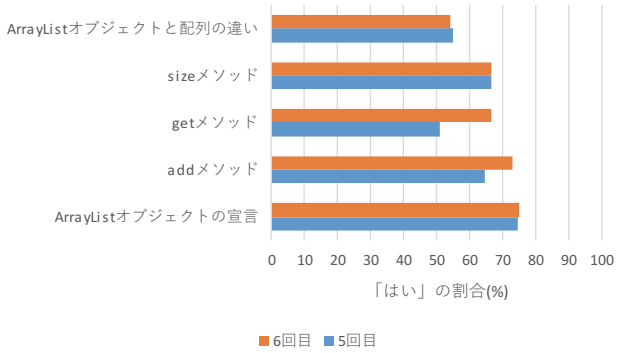

図 14 ArrayList オブジェクトの補足資料の効果

4.6 HashMap オブジェクトを理解させる工夫

HashMap オブジェクトは，6 回目の授業で説明した．ハッシュキーを使ってオブジェクトを管理することができる．工夫した点は，猫をペットに見立て，猫(Cat)クラスを定義し，ペットの名前でオブジェクトを管理するプログラムを示した．”tama”，”mike”，”shiro”等の名前で，put メソッドでオブジェクトを入れ，”tama”，”mike”，”shiro”の名前で，get メソッドでオブジェクトを参照した．get したオブジェクトが区別できるように，Cat オブジェクトの口元にその文字を表示するように工夫した．配列や ArrayList オブジェクトは，オブジェクトを番号で管理するが，HashMap オブジェクトでは，猫に名札を付け，その名札に書かれている文字で利用できることを説明した．

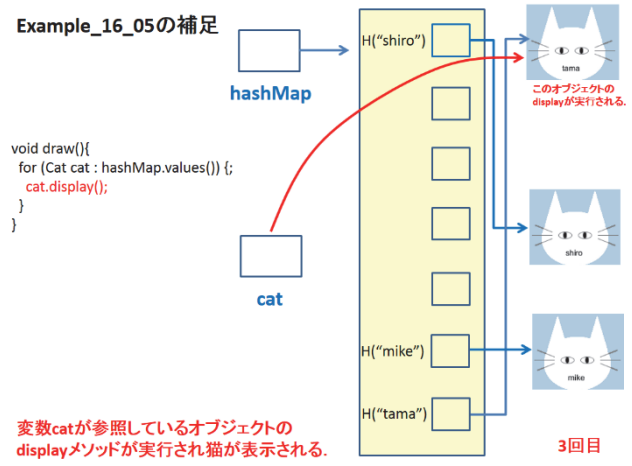

図 15 HashMap オブジェクトの補足資料(抜粋)

HashMap オブジェクトは 6 回目の授業で説明した．さ

© 2018 Information Processing Society of Japan

らに 7 回目の授業の冒頭で補足説明を行った．補足資料を図 15 に示す．ArrayList オブジェクトと同様に，補足資料を見たことによる理解度を調査した．「はい」と回答した割合を図 16 に示す．HashMap オブジェクトに関しては，put メソッドとキーで管理できることの理解が向上したが，size メソッドに関しては僅かながら逆効果の結果となった．

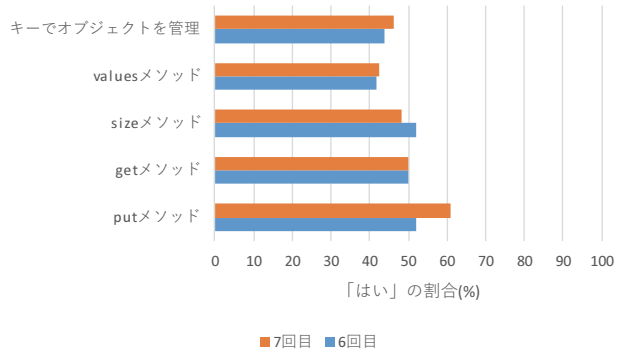

図 16 補足資料の効果

猫は筆者が作成したものである．アンケート調査で，「Cat オブジェクトは，可愛いと思いますか．」との設問に対して，48 名中 34 名(70.8%)が「はい」と回答した．Cat オブジェクトは Chicken オブジェクトほど好感を得られなかった．好感度と理解度に関係があるかも知れない．

```
SubJitterBug jit;
SubJitterBug bug;

void setup() {
  size (480, 120);
  jit = new SubJitterBug(width * 0.33, height / 2, 50);
  bug = new SubJitterBug(width * 0.66, height / 2, 10);
  jit.setColor(color(0, 0, 255));
  bug.setColor(color(255, 0, 0));
}

void draw() {
  jit.move();
  bug.move();
  jit.display();
  bug.display();
}

class SubJitterBug extends JitterBug{
  color c;
  SubJitterBug(float tempX, float tempY, int tempDiameter){
    super(tempX, tempY, tempDiameter);
  }
  void setColor(color tempC){
    c = tempC;
  }

  @Override
  void display(){
    fill(c);
    ellipse(x, y, diameter, diameter);
  }
}
```

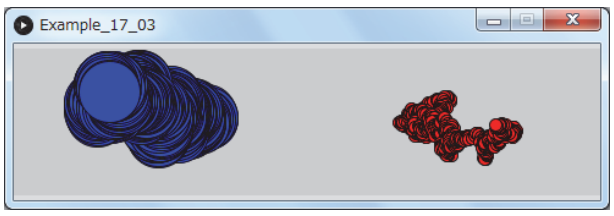

図 17 クラス拡張の例

4.7 継承を理解させる工夫

親クラスにない機能を子クラスに与える例として，JitterBug を取り上げた．親クラスを JitterBug に，子クラスは SubJitterBug クラスにした．具体的には子クラスには，色を付けられるフィールドを追加し，セッターとして setColor メソッドを追加した．この様子を図 17 に示す．

Java で教えた経験から，「受講者に置き換えて説明した．皆さんは，親から誕生しており，親の性質を引き継いでいます．皆さんは情報システム工学という学問を学んでおり，親はそのことは知りません．親には無い性質を皆さんが身に付けていることになります．このように親の性質を継承しつつ，親に無い機能を身に付けることを，クラス拡張と言います」と説明する工夫を取り入れ，70%を超える理解が得られた．

5. まとめ

本論文では，「コンピュータプログラミングⅡ」の中間試験までの内容について，Processing によるオブジェクト指向プログラミング入門教育の実践による工夫等を述べた．情報環境学部の時代に Java でオブジェクト指向を教えた経験を活かし，Processing を使ったデザイン工学科のオブジェクト指向プログラミングの入門にノウハウを取り入れた．筆者は，Java よりも Processing の方がオブジェクト指向を教授し易いとの感想を持った．受講者がどのように感じているかは，比較対象が無いため結論付けるのは難しいが，授業毎に実施しているアンケートの理解度調査を継続的に実施し，分析を行う予定である．

さらに，教室の形状がグループ学習形式から縦長の講義形式に変わったことの影響，実行結果に表示されるキャラクタの様子が理解度に影響を与えている可能性がある．今後の分析で，この辺りの関係を明らかにしたい．中間試験以降は，教科書の「12 章 データ」と「13 章 拡張」を学習しながら，中間試験までに学んだ事を応用し，受講者にとって興味を喚起するプログラムを示すことによって，受講者のモチベーションの向上にも努めたい．

参考文献

[1] 土肥紳一，プログラミング教育における教室内の空気の調査，情報処理学会，情報教育シンポジウム SSS2017 論文集，Vol.2017，pp271-276，2017
[2] 三浦元喜，Processing Web IDE を用いたプログラミング基礎教育の試み，情報処理学会，情報教育シンポジウム SSS2013 論文集，Vol.2013，2017
[3] 土肥紳一，今野紀子，Processing による高校生を対象としたプログラミング入門体験，情報処理学会，情報教育シンポジウム SSS2013 論文集，Vol.2013，no.2，pp217-224，2013
[4] コンピュータプログラミングⅡ，web サイト，https://dohi.chiba.dendai.ac.jp/~dohi/computer-programming-2/ad/ 2018 年 6 月 4 日閲覧
[5] Casey Reas，Ben Fry 著，船田 巧 訳，https://www.oreilly.co.jp/books/9784873117737/ 2018 年 6 月 4 日閲覧

© 2018 Information Processing Society of Japan