プログラミング初学者を対象としたオブジェクト指向プログラミング教育システムの提案 -オブジェクト指向の基本概念の理解に基づいたプログラムの作成・実行支援機能を中心として-

大城 正典[1,a)]　永井 保夫[1,b)]

概要：著者らは，視覚化機能を持ったEclipseプラグインおよびこれを利用したプログラミング教育支援システムを開発している．この視覚化機能はプログラムの構造や動作の直感的理解を助けるが，視覚化機能のみではプログラムの作成能力を直接向上させることは期待できない．ソースコードを書かせる演習では，ソースコードの各文法的構造を正しく書かなければ静的視覚化による理解向上の効果は望めない．またオブジェクト指向言語の視覚化に関する先行研究の多くが，学習者がオブジェクト指向の基本を理解済みであることを前提としている．そこで筆者らは，仕様を学習者に示し，構造毎に段階的にソースコードを書かせてモニタリングするという演習システムを提案した．さらに本論文では，オブジェクト指向プログラミングの基礎部分を題材として，事前に教材を提示することによって基本的な考え方を学習できるようにし，目的となるプログラムの想定される振る舞いをあらかじめ動的視覚化によって観察した上で，段階的なガイドに従ってソースコードを書き，最終的に自分の書いたソースコードを実行してその動きを観察することができる学習システムを提案する．

キーワード：プログラミング教育，オブジェクト指向，視覚化，Java，Eclipse

The proposition of an object-oriented programming education system for programming learners : focused on making and executing programs based on understanding basics of object-oriented

Ohshiro Masanori[1,a)]　Nagai Yasuo[1,b)]

Abstract: Authors have developed the Eclipse plug-in with the visualization function and the programming educational support system. The visualization function makes the structure and the behavior of the program easy to understand, but you can't expect to make them improve the making ability of the program directly only by the visualization. In lesson for writing source code, the visualization feature that helps understanding to work properly, but only when a student writes every grammatical structure correctly in a source code. In addition, a number of previous studies on the visualization of an object-oriented language, is based on the premise that the learner is already understanding of basics of object-oriented programming. Therefore, we introduced a practice system that shows a simple class specification to learners and makes them write source code structure by structure referring to step by step guides. In this paper, we propose a new improved practice system. We choice a basics of object-oriented programming as a theme. In the system, at the beginning of practices, teaching materials are displayed to learners to make them understand a basic way of thinking. When a question of making program is shown, they can watch the behavior of a correct code using dynamic visualization. Then Learners are instructed to write code structure by structure referring to step by step guides. They can run and observe the behavior of their source codes using dynamic visualization.

Keywords: programming education, object-oriented, visualization, Java, Eclipse

1. はじめに

オブジェクト指向プログラミングの教育において，学習者を対象としたプログラミング学習環境の提案 [1], [2] や初中等教育におけるオブジェクト指向プログラミング言語を利用した教育に関する研究 [3] が精力的におこなわれている．このようなプログラミング教育においては，プログラムの理解を支援することが不可欠であり，静的な手法で解析したプログラムの振る舞い（プログラムのソースコードからフローチャートや実行パスの生成）やプログラムの実行履歴を取得することによる動的な振る舞いを視覚化情報として提供することで学習者の理解を支援する手法が提案されている [4], [5], [6].

われわれは，このようなオブジェクト指向プログラミングの教育を支援するために，プログラミング初学者を対象としたアプローチと，UML やオブジェクト指向による分析設計（モデリング）に関する知識を活かしたプログラミング学習者を対象としたアプローチに分けて研究をおこなっている [7], [8], [9], [10], [11], [12], [13]

前者の研究では，Java によって書かれたソースプログラム内に定義されたクラスとそのメンバを視覚化 (静的視覚化) したり，プログラムの動作の様子を視覚化 (動的視覚化) してきた．これらの視覚化機能は特にオブジェクト指向プログラミング教育の初期において，クラスやオブジェクトの構造の理解に有効だと思われる．

後者の研究では，プログラミング学習者がモデリングなどのオブジェクト指向分析設計を学び始める段階においては，複数のクラス・オブジェクト間の関係やメソッドの呼び出し関係，それらが動作している様子を観察することで，複雑性の高い設計の内容も理解しやすくなるものと思われる．

しかし，視覚化によってプログラムの静的・動的側面を理解できたとしても，プログラミングや設計といったソフトウェアを作成する能力の向上に直接繋がるわけではない．ソフトウェアの作成能力を向上させるには，視覚化をともなったプログラミングの適切な演習方法も必要になると考えられる．たとえば，本システムは当初からプログラムのコーディングと同時にリアルタイムに静的視覚化を行う機能を持っているが，学習者が実際にプログラムを記述するときに，文法にしたがった構造（対応する括弧で定められたクラスやメソッド，制御文などの形）から書かないと，視覚化によるサポートも効果を十分に発揮できないばかりか，こういった構造が持つ意味や働きが理解できず，プログラミング能力がいつまでも向上しないという悪循環に陥る可能性が考えられる．そこで，われわれは学習者に仕様を提示した上でプログラムを構造ごとにコーディングさせるようにガイドを行う演習機能を提案した [12]．この演習機能では，学習者がどの構造まで書けたかをネットワークを介してリアルタイムでモニタリングすることも可能になっており，演習時における実効性を高めるように配慮されている．

提案した段階的コーディングによる演習は，実際のプログラミング技能修得者のコーディング時における思考を学習者に追体験させることになり，十分な数の問題に対して演習を行わせることで，その思考の流れを身につけることが期待できる．しかしながら課題の仕様をよく理解せずに，提示された段階的なガイドに形だけしたがってソースプログラムをただ受動的に書いているだけでは，プログラム上の構造の理解や経験者の思考の追体験といった効果はあまり期待できない．そこで著者らは本システムに用意されている動的視覚化を使用し，課題として出されたプログラムの内容を意識し，よく理解するための助けとして，完成させたプログラムが実際に動作する様子を観察できるようにしている．

本論文では，前者のプログラミング初学者を対象としたアプローチに対して，新たなプログラミング教育システムの提案をおこなう．

現状のオブジェクト指向プログラミングの教育では，学生はプログラミングの初期段階からオブジェクト指向の概念に直面することになる．オブジェクト指向の概念，たとえば「オブジェクトとは何か?」,「クラスとは何か?」などの本質的な概念の理解をすることは容易ではないのが現状である．

われわれが実施しているオブジェクト指向プログラミングの教育では，構造化プログラミングの知識で理解できるプログラムを先に提示するために，オブジェクト指向の基本的な概念の把握が不十分なまま，クラスを用いたプログラムを教える場合が多い．その結果，なぜ,「オブジェクト」と「クラス」が必要となるかが理解されないまま，プログラミングの教育が進められることになってしまう．

われわれは，このような問題点に対応するために，視覚化機能を持つプログラミング学習支援システムを Eclipse プラグインとして開発してきた．しかしながら，視覚化のみでは学習者のプログラムを能動的に作成する能力を高めることは難しいと考えている．そこで，ガイドをに基づき構造ごとに段階的なコーディングを行わせる演習機能を提案してきた．本論文では，新たにオブジェクト指向の基本的な考え方を学習させることで，クラスの定義，オブジェクトの生成・初期化・利用を行うプログラムを作成し，実行確認までを一環して行えるように，演習を取り扱うプログラミング教育システムについて説明する．

[1] 東京情報大学
Uniersity of Information Sciences, 4–1 Onaridai Wakaba-ku, Chiba, Chiba, Japan
[a)] ohshiro@rsch.tuis.ac.jp
[b)] nagai@rsch.tuis.ac.jp

2. 関連研究

ここで，先行研究によって提案・実用化されている視覚化をともなう教育システムの特徴をあげ，著者らが提案するシステムとの相違を述べる．

Jelliot3 では，オブジェクト指向プログラミングの初学者が学習する際に，オブジェクトや，変数とオブジェクトの参照関係などの抽象的な概念を正しく理解するのが困難であるため，アニメーションなどの可視化機能を提供することで，初学者がオブジェクトのデータ構造を理解しわかりやすくことを目的としていた [14].

AnchorGarden では，Java 言語を対象として，型，変数，オブジェクトを視覚化し，さらに視覚化した各要素を操作することによって抽象的な概念理解を促進させることを目的に開発された [15].

AnchorGardenPlus では，AnchorGarden を改良して，オブジェクト指向の主要概念である，継承，カプセル化，ポリモーフィズムの学習を支援している [16].

BlueJ は，Java プログラミング用の統合開発環境であり，クラス図の生成に関連付けながら，プログラムが作成できるようになっている．さらに，main メソッドなしにクラスからオブジェクトを作成し，オブジェクトをインタラクティブに操作できる機能も備えている [17].

上記の研究においては，初学者である学生は初期段階からオブジェクト指向の概念とそれを支える文法上の構造について理解をしていることが前提になると思われる．つまり，オブジェクト指向の概念，たとえば「オブジェクトとは何か？」,「クラスとは何か？」などの本質的な概念については，既に理解がおこなわれているものとして利用されるように設計されており，それらを表現するソースコードを学習者が正しく書けることまで前提となっていると考えている．

一方，今回のわれわれの提案では，構造化プログラミングまでの知識を前提とし，オブジェクト指向の基本的な概念自体をまだ理解しておらず，オブジェクト指向を支える文法構造も実際に書けない段階からクラスを用いたプログラムを教える場合に対して，なぜ「オブジェクト」と「クラス」が必要となるかをしっかり理解させてから，対応するソースコードはもちろん，main() メソッドなども含む完全なプログラムを書ける力を修得できるようにプログラミング教育を進めていくことを目的としている．

3. 演習の流れ

本システムでの演習の基本的な流れは次の様になる．（1）Eclipse 上で演習開始メニューを選択すると初めて使用する演習問題の場合には，関連する教材が表示される．教材は HTML ベースで作成されており使用されている図式も視覚化プラグインで利用されているものと統一されている．学習者は問題に取りかかる前に十分に教材で学習することができるように，授業時間外でも表示させることができる．（2）教材内容を見た後，学習者にはプログラム作成問題が提示される．表示されるのは問題文だけでは無く，完成させるべきプログラムの動作を動的視覚化によってあらかじめ観察することができる．問題の意味を把握した上で，問題作成にとりかかる．（3）プログラムの作成時間では，段階的に示されるガイドに従って，構造毎に書いてく．完成した構造はその都度，静的視覚化機能によって視覚化される．（4）最後の段階まで書き終わったら，コンパイルし実行して正しく動作するか確認する．その際，動的視覚化によって自分の書いたプログラムの動作を視覚的に確認することができる．

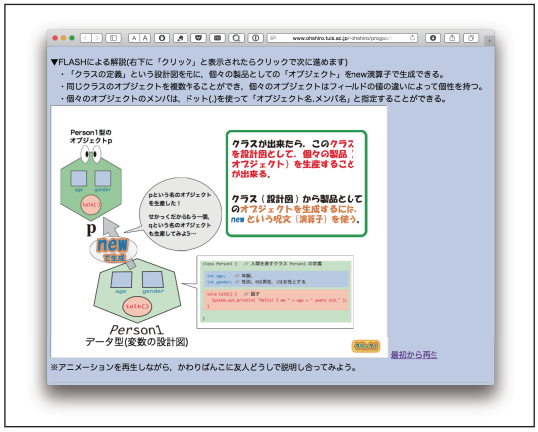

図 2 オブジェクト生成の考え方を説明する Flash 動画.
Fig. 2 A flash movie for expression of object creation.

以下に，各段階について詳しく述べる．

4. 教材の提示

本演習システムでは，学習者がその演習問題に初めてとりかかる前に，関連する教材が提示される．**図 1**〜**図 3** がその例である．図 1 はクラスについての考え方を解説している教材の一部であり，クラスの定義からオブジェクトを生成できることについては，**図 2** のような Flash ムービーも用意されている．図 3 は，オブジェクトを利用するにはどのように書けば良いかを説明した資料と，サンプルプログラムの動作を解説つきで再現する Flash ムービーの例である．このような関連教材は，最初の提示以降はシステムの GUI から任意に参照できる．これらの教材によって，「プログラム上に必要となる概念をクラスによって表現する」こと，「動作するプログラムの中でクラスという設計図で表現された概念の実例であるオブジェクトが個々の働きと役割を持つ」ことなど，オブジェクト指向の基礎的な概念を理解できるように配慮する．

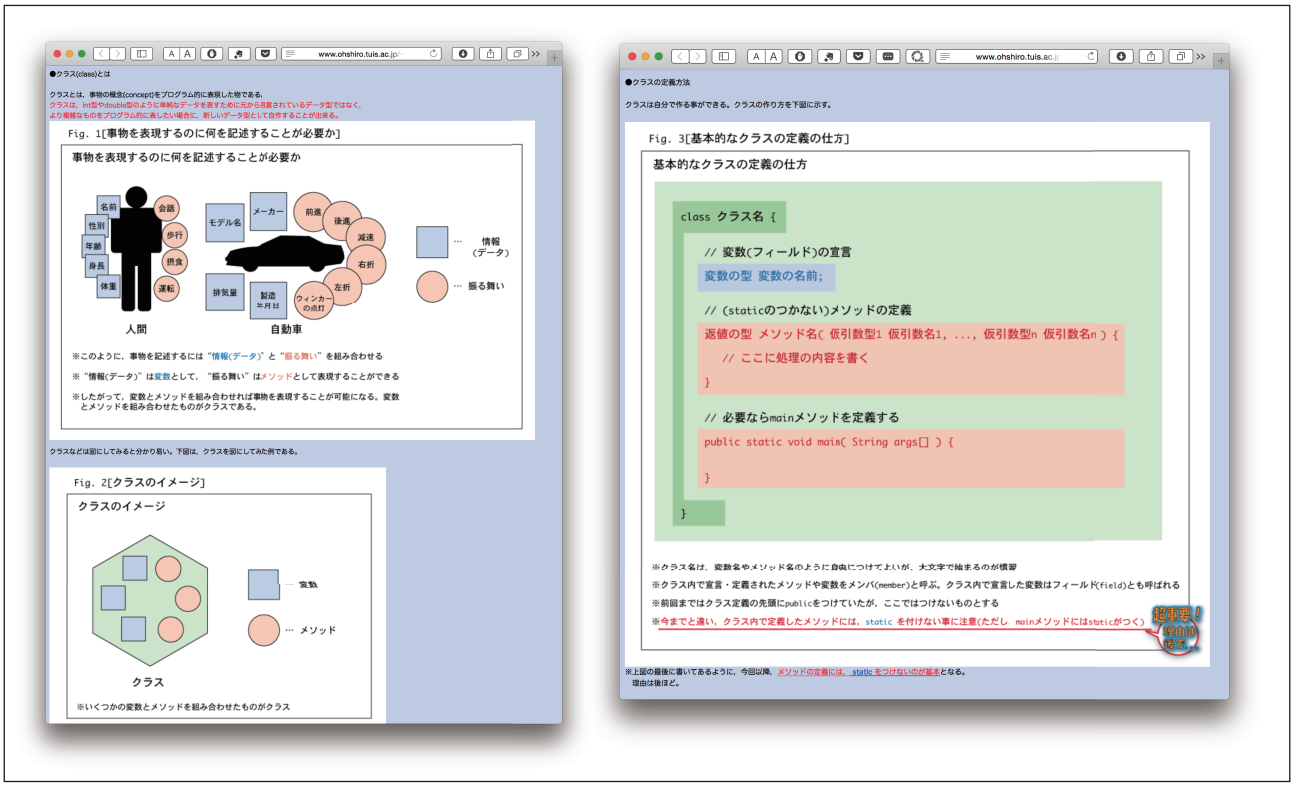

図 1　クラスに関する教材の例.

Fig. 1　A example of teaching materials of class.

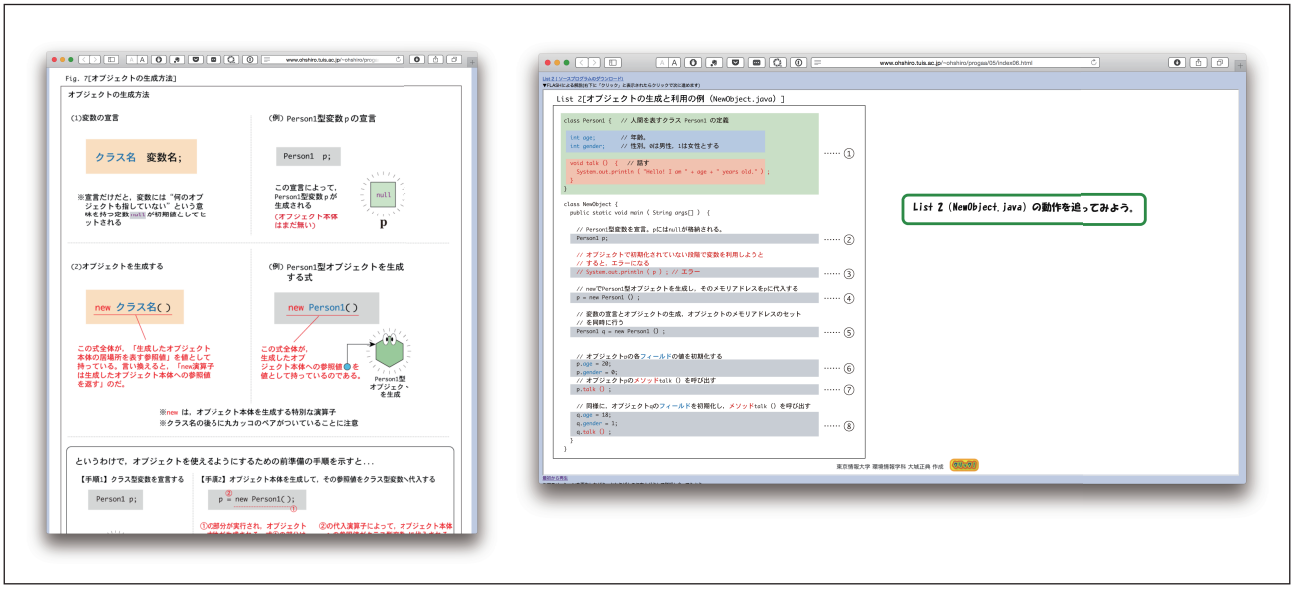

図 3　オブジェクトの利用に関する教材の例.

Fig. 3　A example of teaching materials of object usage.

このような資料をよく見た後，実際の問題が提示される．

5.　演習問題の提示

ここでは演習問題の例として，オブジェクト指向プログラミングの初歩に理解すべきであるオブジェクトの生成と初期化・メンバの利用という典型的な処理をあげる．

図 4 は，クラスの定義，オブジェクトの生成，コンストラクタの定義と呼び出しまで学習した段階で想定される演習問題の例である．元となるソースコードのコメント内に記述された特殊なコマンドと一定のルールにより自動生成される問題文が，クラスの静的視覚化画面と同時に表示される [12]．しかし，問題文だけでは学習者がその問題文の意図自体を十分に理解できていない可能性も高いと思われる．そこで問題文だけではなく，このソースコードの

```
/**
 * 人間を表すクラス.
 *
 * @q#0 人間を表す Person クラスを定義しよう.
 * このクラスは以下のメンバを持つ.
 * @q#mlist
 */
class Person {
  /**
   * 年齢を表す.
   */
  int age;

  /**
   * 身長を表す.
   */
  double height;

  /**
   * age フィールドと height フィールドを初期化するコンスト…
   */
  Person(int age, double height) {
    // @q#begin 前段階で作成したコンストラクタの中身を適切…
    this.age = age;
    this.height = height;
    // @q#end
  }

  /**
   * すべてのフィールドの値を表示するメソッド.
   */
  void print() {
    // @q#begin 前段階で作成したメソッドprint() の中身を適…
    System.out.println(age + "," + height);
    // @q#end
  }

  /**
   * main メソッド.
   */
  public static void main(String args[]) {
    Person p1 = new Person(19, 172.0); // @q# mainメソッド…
    p1.print(); // @q# mainメソッドの2番目の処理として
  }
}
```

図 4 問題の元となるソースコードの例.
Fig. 4 A sample source code for a question.

main() メソッドを実行した様子も繰り返し表示される．その様子を**図 5**〜**図 8** に示す．図 5 は，main() メソッド実行直前の表示である．床に広げられているように敷かれているのは，オブジェクトの設計図とも言えるクラスの定義であり，このように動的視覚化は基本的にメインクラスを見下ろすような視点から開始される．なお，main() メソッドは static メンバであるために，クラスの上にすでに実体として置かれている．**図 6** は，Person 型変数 p1 が宣言された時点の表示であり，その値はデフォルトの初期値である null となっている．**図 7** は，Person 型オブジェクトが new 演算子によって生成される様子である．設計図であるクラスから浮き上がるようにオブジェクトが生成される．この時点でこのオブジェクトは初期化が完了していない状態であることを，目玉を白く描写することで表現している．

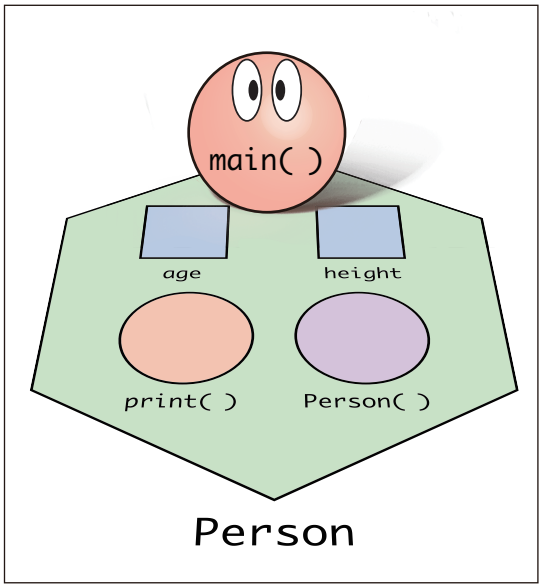

図 5 動的視覚化の最初の状態．main() メソッド実行直前.
Fig. 5 The first state of dynamic visualization. Just before the execution of main() method.

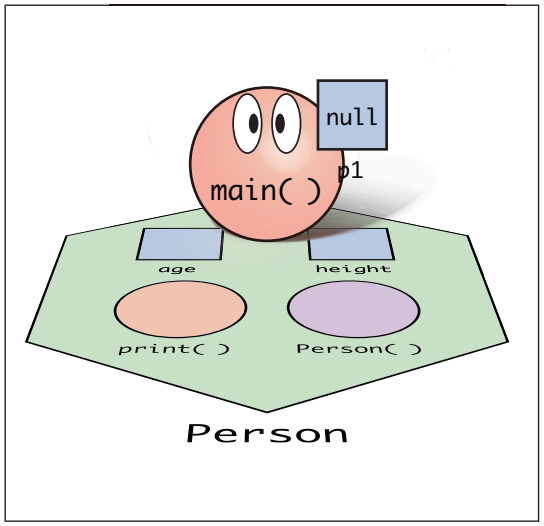

図 6 Person 型変数 p1 の宣言文が実行された状態.
Fig. 6 Declaration of a variable p1 of type Person.

なお，この時点でオブジェクトで利用できるメンバはコンストラクタのみのため，コンストラクタについている目玉だけは黒くなっている．図 8 は，生成された Person 型オブジェクトのコンストラクタが呼び出され，age フィールドと height フィールドが初期化され，このオブジェクトへの参照値によって変数 p1 が初期化された時点の表示であ

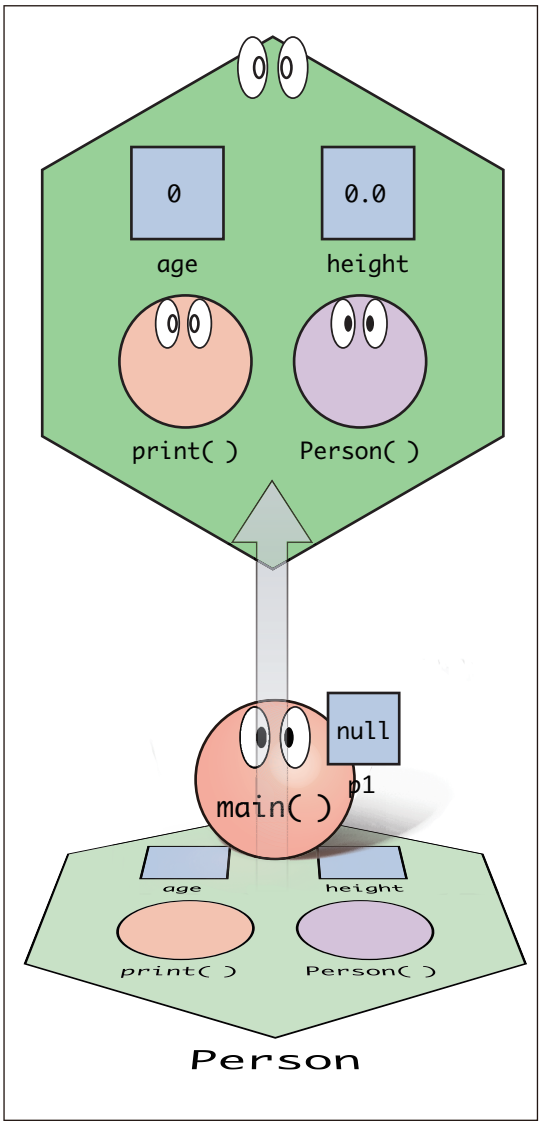

図 7 Person 型オブジェクトが生成される様子.
Fig. 7 This figure shows creating an object from class Person.

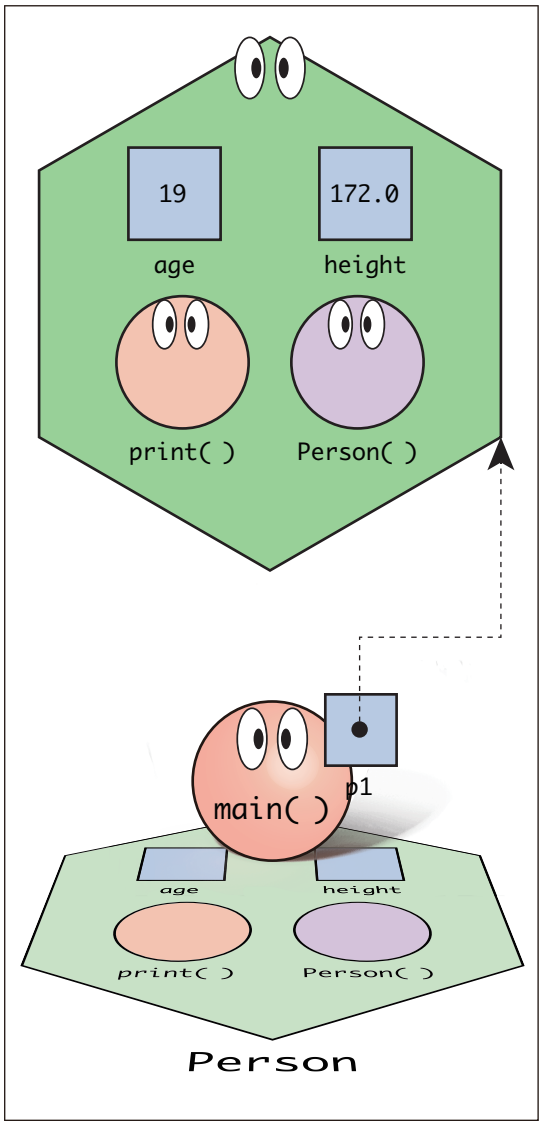

図 8 変数 p1 が初期化された直後の様子.
Fig. 8 Just after initialization of p1.

る．以降，main() メソッドの終了までの動作が表示され，それが繰り返される．

学習者は，この動的視覚化の表示によってこれから作成することになるプログラムの動作を確認をしながら，ガイドにしたがって段階的にコーディングを行う [12]．すべてのガイドにしたがってコーディングを終えたことが確認されると，完成したプログラムを実行し，動的視覚化によってその振る舞いを観察することができる．その際は，**図 9** に示すようにソースコードのどの部分が現在実行されているかを同時に見ることができる．

6. 動的視覚化を利用した動作の確認と検証

この問題（ 図 4）の場合，学習者がよく間違える点として，コンストラクタの処理内容があげられる．すなわち，

```
this.age = age;
```

としなければいけないところを

```
age = age;
```

と書いてしまうケースである．このような場合，初心者はコンソールに出力された表示結果だけではどこが間違っていたかすぐには理解できないことが多い．通常，このような変数の状態監視はデバッガを利用して行うことができるが，初心者にはデバッガを自主的に利用することも一般的には難しい．しかし，本システムでは動的視覚化によって着目すべきフィールドとその値が見やすく図によって表示されているので，その変化を追い，コンストラクタの処理内で初期化が正しく行われていない事実を確認することは比較的簡単であると思われる．例えばこの場合，学習者はオブジェクトの初期化が終わった段階で図 8 のような表示内容を期待しているが，実際には**図 10** のように表示され，コンストラクタの実行が終わった後にもかかわらず，フィールドの値がデフォルト値のままであることに気づくと期待できる．指導者も，デバッガの専門的な使い方を説

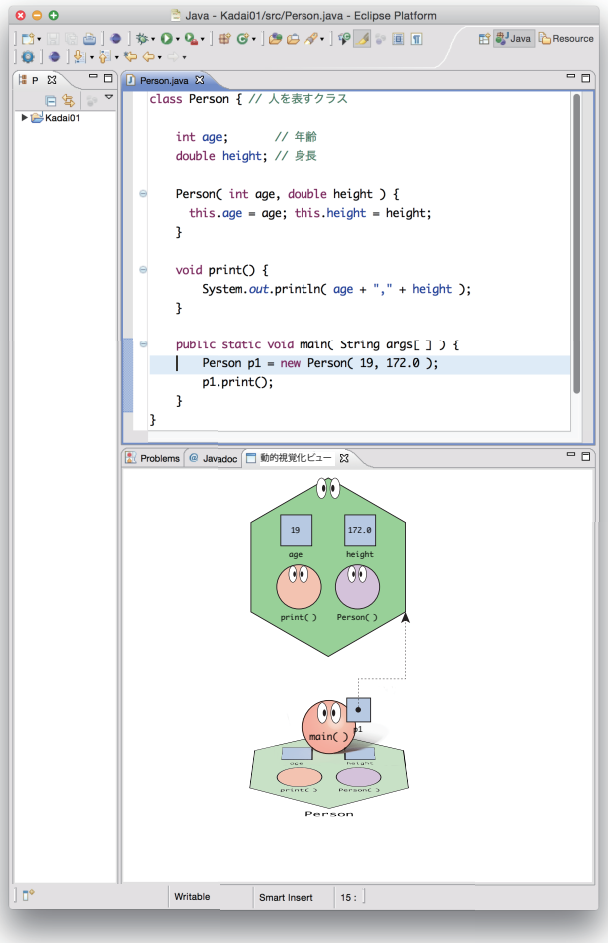

図 9 ソースコードと動的視覚化の同時表示.
Fig. 9 The source code and dynamic visualization are displayed simultaneously.

明するまでも無く，実行されているソースコードの位置と図示されているフィールドの値に着目して観察するように指導するだけで良い．もちろん，専用のデバッガツールの有用性は論を待たないが，動的視覚化による検証は学習者が十分成長するまでの間は十分に有用であると言える．

7. おわりに

本稿では，視覚化機能によってサポートされたオブジェクト指向プログラミング初学者のための一貫した演習システムを提案した．本システムは単に完成されたプログラムを視覚化するだけでなく，プログラム作成能力の向上を目的としたものであるので，実際に授業で使用して効果を検証してみる予定である．また仕様上，本システムを使った演習は反転授業に取り入れることも可能である．授業内での利用で効果が確認できれば，反転授業用のツールとして試用し効果を検証していきたい．

その他の今後の課題としては，メソッド内処理の意味的正しさの自動検証や，モジュール化（アクセス制御）の他，継承・抽象メソッド・抽象クラス・interface などの視

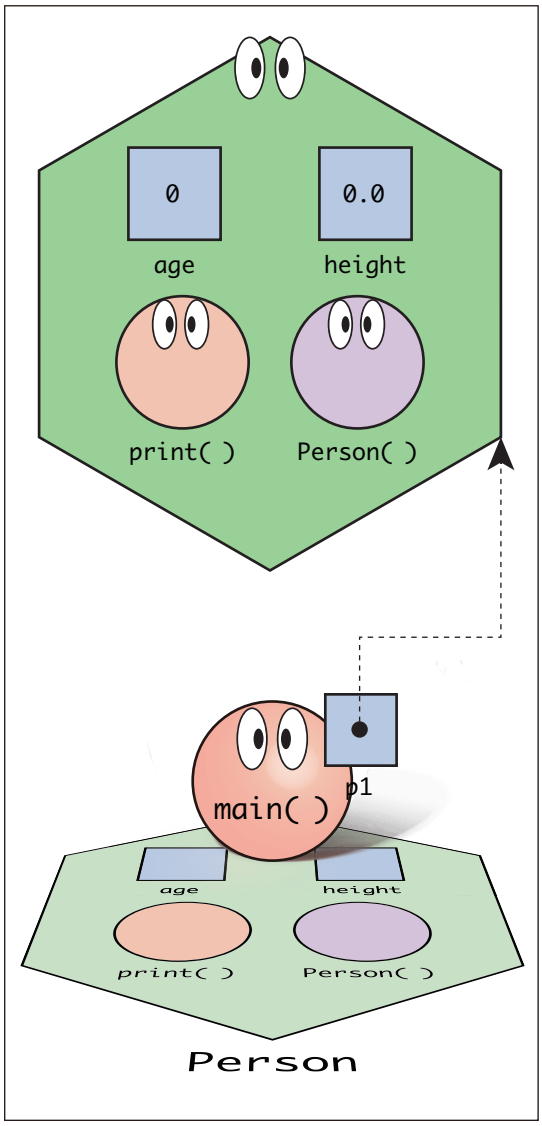

図 10 フィールドが正しく初期化されていない場合の表示
Fig. 10 The case of bad initialization.

覚化様式 [8] の改善や演習手法についての検討などがあげられる．

参考文献

[1] 萩庭 崇，永田守男：オブジェクト指向言語のための視覚的プログラム支援環境，情報処理学会ソフトウエア工学研究，Vol. 96, No. 4, pp. 25–32 (1996).
[2] 長 慎也，甲斐宗典，川合 晶，日野孝昭，前島真一，筧 捷彦：Nigari-Java 言語へも移行しやすい初学者向けプログラミング言語，情報処理学会研究報告コンピュータと教育，Vol. 103, No. CE-071, pp. 13–20 (2003).
[3] 兼宗 進，中谷多哉子，御手洗理英，福井眞吾，久野 靖：初中等教育におけるオブジェクト指向プログラミングの実践と評価，情報処理学会論文誌，Vol. 44, No. SIG 14 (PRO18), pp. 58–71 (2003).
[4] 喜多義弘，川添貴議，片山徹郎：初心者を対象にした Java プログラム自動可視化ツールの実現に向けて，信学技報 SS，Vol. 104, No. 570, pp. 19–24 (2005).
[5] 谷口孝治，石尾 隆，神谷年洋，楠本真二，井上 克：プログラム実行履歴からの簡潔なシーケンス図の生成手法，

コンピュータソフトウェア， Vol. 24, No. 3, pp. 153–169 (2007).
[6] 竹下彰人，片山徹郎：シーケンス図を用いた実行履歴の可視化による Java プログラムの理解支援に関する考察，信学技報 SS，Vol. 106, No. 426, pp. 43–48 (2006).
[7] 大城正典，永井保夫：情報視覚化を活用したオブジェクト指向プログラミング教育支援システムの提案，信学技報教育工学， Vol. 110, No. 453, pp. 131–136 (2011).
[8] 大城正典，永井保夫：初等プログラミングから設計レベルまでを対象としたオブジェクト指向教育のための支援システムの提案，情報処理学会 情報教育シンポジウム 2011 論文集 SSS2011，pp. 59–66 (2011).
[9] 大城正典，永井保夫：Eclipse を用いたオブジェクト指向プログラミング教育支援視覚化システムの設計と実装，信学技報教育工学， Vol. 112, No. 500, pp. 185–188 (2013).
[10] 大城正典，永井保夫：モニタ機能と可視化機能を持った構造指向による漸次的なプログラム作成学習システム，信学技報教育工学， Vol. 113, No. 482, pp. 31–34 (2014).
[11] 大城正典，永井保夫：段階的コーディングガイド機能およびモニタ機能を持つオブジェクト指向プログラミング教育のための視覚化支援システムの提案，信学技報教育工学， Vol. 114, No. 260, pp. 53–58 (2014).
[12] 大城正典，永井保夫：Eclipse 視覚化プラグインによる総合的なプログラミング教育支援システム，情報処理学会 情報教育シンポジウム 2015 論文集 SSS2015，pp. 23–30 (2015).
[13] 大城正典，永井保夫：視覚化機能を持つ Eclipse プラグインによる段階的コーディングから動作検証までをサポートするオブジェクト指向プログラミング学習システム，信学技報教育工学， Vol. 115, No. 492, pp. 61–66 (2016).
[14] Moreno, A., Myller, E. and Ben-Ari, M.: Visualizing Programs with Jeliot 3, *In Procs. of the International Working Conference on Advanced Visual Interfaces*, ACM, pp. 373–376 (2004).
[15] 三浦元喜，杉原太郎，国藤　進：オブジェクト指向言語における変数とデータの関係を理解するためのワークベンチ，情報処理学会論文誌，Vol. 50, No. 10, pp. 2396–2408 (2009).
[16] 浅井俊伍，酒井三四郎：オブジェクト指向言語における主要な概念を理解するためのワークベンチ，情報教育シンポジウム 2015 論文集，pp. 1–8 (2015).
[17] Kolling, M., Quig, B., Patterson, A. and Rosenberg, J.: The BlueJ system and its pedagogy, *Journal of Computer Science Education Special Issue on Learning and Teaching Object Technology*, Vol. 13, No. 4, pp. 249–268 (2003).