オブジェクト指向言語におけるクラス定義の意味とオブジェクトの振舞いを理解するためのワークベンチ

三浦 元喜 †　　杉原 太郎 ‡

概要

これまでに，Java や C# などの静的な型付けを行うオブジェクト指向言語を学習するうえで比較的抽象度が高くつまずきやすい「型・変数・オブジェクトとデータ参照」の理解を促進するためのワークベンチ Anchor Garden (AG) が提案されてきた．AG では，プリミティブ型とオブジェクト型の違いや，変数や配列との参照関係を視覚的に提示することが可能である反面，クラス定義を行ううえで必要となるメンバー変数やメソッドの概念，および実行時の振舞いを視覚的に表現することができなかった．我々は，オブジェクト指向言語初学者がクラス定義を行うにあたって必要となる上記概念知識を効果的に修得するため，AG と同様の視覚・操作モデルを設計し，例示や試行錯誤による学習支援環境を構築した．

A Workbench for Understanding Meaning of Class Definition and Behavior of Objects

Motoki Miura†　　Taro Sugihara‡

Abstract

For effective learning of object-oriented programming language (OOPL) such as Java and C#, Anchor Garden system (AG) had been proposed. AG is designed to explain concepts of type, variable, object, and its relations through visualization and manipulation to the visualized models. However, AG could not represent concepts of member fields, methods, and its behaviors through method calls. These concepts will be necessary when writing a source code that defines classes. We developed a similar workbench software that facilitates novice OOPL programmers to acquire knowledge regarding above concepts.

1　はじめに

Java や C#などのオブジェクト指向プログラミング言語は，それを修得しようとする学習者にとって一般に敷居が高い．その理由として，クラスとインスタンス，メンバ変数とメソッド，コンストラクタといった抽象度の高い概念の理解が必要になることや，それらがどのように影響しあうのかを知ることが難しいことが挙げられる．通常，これらの概念を理解するためには，書籍等の説明を読み，実際にソースコードを編集してプログラムを作りながら振舞いを調べることが必要になる．しかし，プログラムを真似ながら作ることはできても一般に振舞いを調べることは難しく，結果としてプログラムとその振舞いを対応付けて理解することが困難であることが多い．特に高抽象概念の理解に到るためには何度もソースコードを編集して，実行する試行錯誤が必要となる．

オブジェクト指向プログラミングに必要な概念や知識を初学者が修得しやすくするための方法論として，視覚的に表現したモデルを初学者に提供し，初学者がそのモデルに対する操作を行うと，振舞いに対応したソースコードを提示するものがある．Anchor Garden[1] (以降，AG) は，型・変数・オブジェクトとデータ参照の概念について，上記の操作「体験」と，対応する出力コードの「観察」によって，円滑かつ良質な試行錯誤を提供することを目的として開発されたシステムである．利用者は基本的にマウス操作を行うのみで，ソースコードを記述せずにオブジェクト指向プログラミングの理解に資する概念を修得することが可能となる．しかし，AG では型・変数・オブジェクトとデータ参照の概念のみを対象としていたため，実際のクラス定義を行ううえで必要となるオブジェクトのメンバ変数やメソッドの概念については対応していなかった．

そこで我々は，従来の AG において欠けていたオブジェクトのメンバ変数やメソッドの概念を追加することで，学習者がクラス定義を行う際に生じる概念とのギャップを埋めることを提案し，その意義について議論する．

† 九州工業大学
Kyushu Institute of Technology
‡ 北陸先端科学技術大学院大学
Japan Advanced Institute of Science and Technology

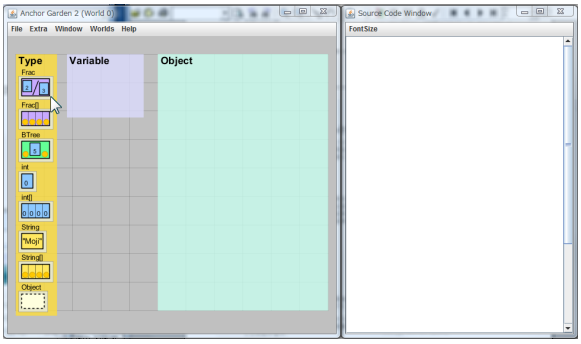

図 1: 初期画面

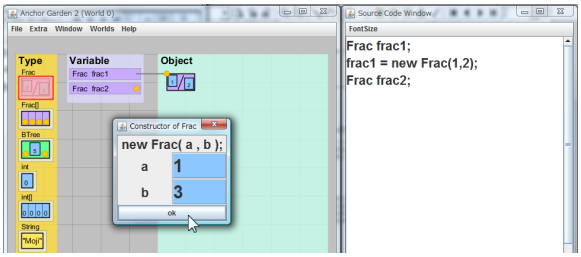

図 2: オブジェクトの生成

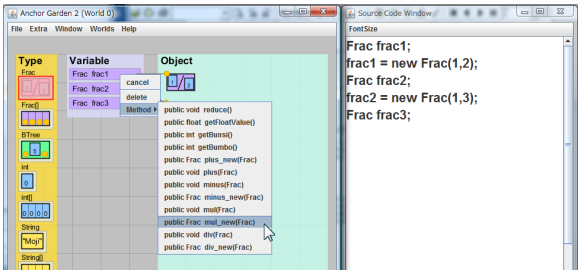

図 3: Variable 右クリックメニュー→メソッド選択

2　学習者に提示するモデルと体験

本節では，学習者に提示する視覚モデルとその体験，および，それによってどのような概念獲得を期待しているかについて，具体的な指導手順の例とシステムの動作を追いながら説明する．

2.1　分数クラスを用いた指導例

最初に，C 言語で分数を表現する構造体を用いる例 (**リスト** 1) を提示し，演算が増えると記述量が増える，もし関数化したとしても構造体と関数が独立であるため管理しにくい，分母に 0 が代入される可能性があることについて説明する．その後，オブジェクトは構造体のメンバに加え，自身の情報を管理する「メソッド」が付属したもの，という概念的な説明を先行して加えておく．

次に，オブジェクトとメソッドの概念を理解するため，同じ計算を提案システムで体験的に実行し，振舞いを観察する．またオブジェクト指向の観点を用いて表現した場合，どのような対応ソースコード

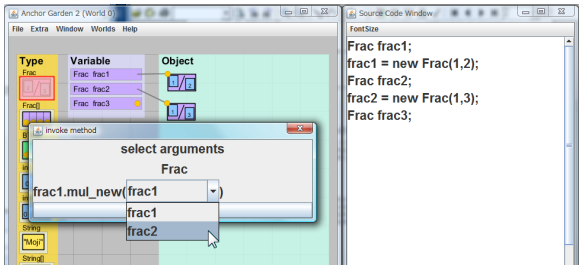

図 4: メソッド引数オブジェクトの選択

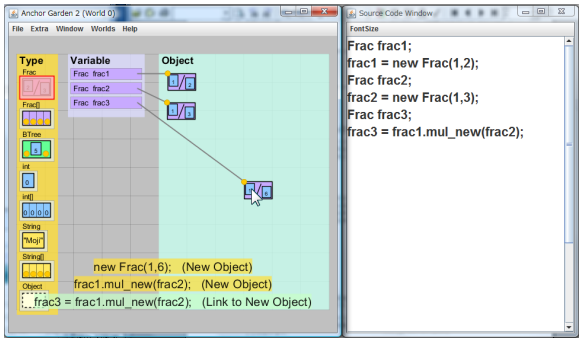

図 5: 返却値のオブジェクトへリンク

が出力されるかを確認する．システム起動直後の初期画面 (**図** 1) において，まず Type の Frac をクリックして選択する．次に中央の Variable 領域をクリックして変数 frac1 を作成し，つづけて Object 領域を SHIFT キーを押しながらクリックして分数オブジェクトを作成する．SHIFT キーを押しながらクリックすると，**図** 2 で示す画面が表示され，コンストラクタに与える引数を指定できる*．分数オブジェクト生成後は，変数 frac1 の右のリンクタブをドラッグして分数オブジェクトにドロップし，オブジェクトにリンクを張る．これを繰り返して frac1 → [1/2]，frac2 → [1/3]，frac3 → (なし) という状況を作成する．右のソースコード画面には，対応するコードが随時表示される．

次に，かけ算を実行し，返却値として新しい分数オブジェクトを返すメソッド mul_new() を呼び出す．変数 frac1 を右クリックしてメニューを表示し，method → public Frac mul_new(Frac) を選択する (**図** 3)．すると，**図** 4 のようにメソッドを呼び出す際に必要となる引数オブジェクトの選択画面となる．ドロップダウンリストから frac2 を選択して OK を押す．すると Object 領域にメソッド実行によって生成された，新しいオブジェクト [1/6] が生成されるので，変数 frac3 からリンクする (**図** 5)．なお，new が付かない public void mul(Frac) を選択すると，メソッドを呼び出した分数オブジェクト (例えば frac1) の値が変更される．ほかには引数をとらないメソッド public void reduce() を呼び出すと，その分数オブジェクトの約分をおこなう．

*SHIFT キーなしの単純なクリックのみの場合は，引数指定なしのデフォルトコンストラクタを利用する．

リスト 1: 構造体による分数の表現と，その計算例

```
#include <stdio.h>
typedef struct {
  int a; //分子
  int b; //分母
} frac;

int main(void){
  frac half, onethird, mulres;
  half.a = 1; half.b = 2;
  onethird.a = 1; onethird.b = 3;

  // 分数のかけ算
  mulres.a = half.a * onethird.a;
  mulres.b = half.b * onethird.b;

  printf("mulres␣=␣%d/%d\n", mulres.a,
      mulres.b);

  return (0);
}
```

この一連の操作と，右側の画面に出力されるソースコードを対比することによって，(1) オブジェクトへのメソッド呼び出しは記述が簡潔になること，(2) 変数の宣言は型名+変数名で行うこと，(3) コンストラクタの呼び出しは new 型名 (引数) で行うこと，(4) 変数からのオブジェクトへの参照は=で行うこと，(5) public Frac mul_new(Frac) では計算結果が入った「新しい分数オブジェクト」が生成されるが，public void mul(Frac) や public void reduce() のように，呼び出したオブジェクトの自分自身の値を直接書き換えるメソッドも存在すること，を体験的に理解できる．その後，Frac クラスの定義 (付録**リスト** 2) を見せることで，コンストラクタや，メソッドにおける引数・返却値の定義はどのように行えばよいのかを具体的な例を通じて理解することが可能となる．

2.2 二分木クラスを用いたデータ構造の図示

従来の AG では，オブジェクトのメンバ変数に，自分と同じクラスのインスタンスを指定できなかったが，提案システムでは，それが可能となった．そのため，双方向リンクや二分木といったデータ構造を表現したり，再帰を含むメソッドを実行できるようになった．

図 6 は，二分木 (BTree) の動作例である．この例では，ユーザが (1) BTree の変数 btre1 を作成 (2) ノードオブジェクト [5] を生成 (3) 変数からノードオブジェクト [5] にリンク (4) ノードオブジェクトを追加するメソッドを 6 回呼び出し (5) ノードオブジェクト [4] を生成しリンク (6) [4] の存在と [6] の存在を問い合わせ (7) 変数 btre3 を作成し，[3] にリンク という一連の操作を行っている．ユーザは，BTree クラスの定義 (付録**リスト** 3) を参照しながらこのように操作することにより，addNode(int) メソッドが実際にどう動作するのかを体験的に修得することが可能である．この例では，addNode(int) メソッドが新しく生成したオブジェクトに自動的にリンクを張るため，ユーザ (学習者) が操作するのはメニューからのメソッド呼び出しと，引数としての整数値の入力のみとなる．**図** 6 では，最後の [4] のオブジェクトだけユーザが Object 領域クリックで作成し，[3] から手作業でリンクしている．この場合にも，参照を表す該当コード btre1.left.right.right = new BTree(4); が正しく生成される．また，マウスカーソルをメンバ変数のリンクタブ (黄色の丸) に合わせることで，そのリンクを張ったオブジェクトがどのような表記で表現できるのかを，ユーザはいつでも確認することができる．**図** 6 では，例として [4] のオブジェクトの左リンクは btre3.right.left および btre1.left.right.right.left であることをユーザが確認している状況を示している．

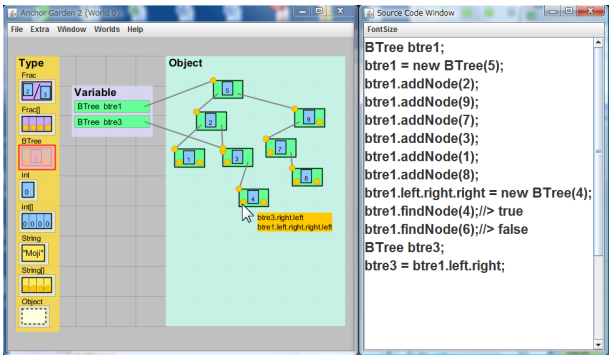

図 6: 二分木 (BTree)

またユーザは，addNode(int) による二分木の構築に加えて，findNode(int) によるノードの探索もメソッド実行により体験できる．**図** 6 では，btre1.findNode(4) が true を返し，btre1.findNode(6) が false を返す例を示している．指定した値をもつノードの有無を true/false で返す以外に，ノードを発見した場合に返却値としてそのオブジェクトを返す (発見できなかったときは null を返す) メソッド findNodeObj(int) メソッドも実装できる．発見できた場合，返却値となる画面上のオブジェクトが点滅しながら拡大するアニメーションにより，ユーザに強調表示する．

このような視覚的表現そのものは，アルゴリズムアニメーションを実行する可視化システムやシミュレーションツールでは一般的に行われている．しかし，任意のオブジェクトに対して任意のタイミングでメソッドを実行したり，ユーザが自由にリンク (参照) を変更できる環境はあまり例がない．BlueJ ではメソッド実行の結果を調査することが可能であるが，1 つのオブジェクトのメンバを 1 つのウィンドウで

表示し，オブジェクトの参照についてはクリックして順次辿っていく方式のため，提案システムのようにオブジェクト間の参照関係を視覚的に表現することはできない†．またソースコードを入力しながら試行錯誤する環境に比べて，初学者の作業負担が少ないため，初学者は本来の学習内容に集中でき，効果的に修得できると予想される．

2.3 文字列オブジェクトのequals()と==の違い

従来の AG では，Java における文字列オブジェクト同士の == による比較 (オブジェクトが一致していれば true) と，equals(String) メソッド (文字列の内容が一致していれば true) の違いについて，画面中のオブジェクトが一致しているかどうかをユーザ自らが目視で確認するしかなかった．提案システムはメソッド呼び出しができるため，equals(String) メソッドを直接呼び出して，文字列オブジェクト同士を比較し，その結果を画面で確認できるようになった．ただし，提案システムは (str1 == str2) のような式をユーザが直接入力して評価することができないため，別名のメソッド (**図** 7 の例では，eqeq_isSame(String)) を通じて確認することになる．

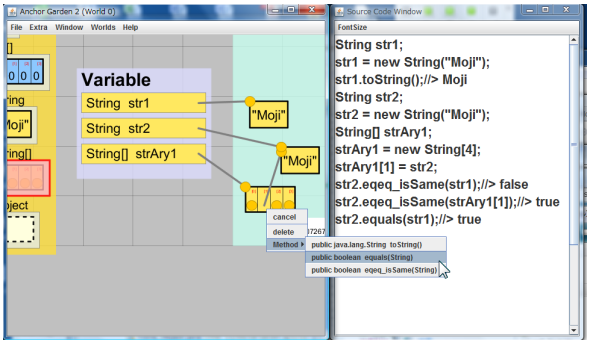

図 7: 文字列オブジェクトの比較：equals() と==の違い

このように，最初に分数や二分木の例について操作しながらオブジェクト指向の概念を獲得し，その後実際のクラス定義を参照することで，段階的なプログラミング能力の向上が見込まれる．

3 実装と制限

本節では，提案システムの実装と，現状における制限事項について述べる．

3.1 実装

提案システムは Java アプリケーションとして実装しており，Java Web Start で動作する．オブジェクトへのメソッド呼び出しは画面内に表示されているオブジェクトに対し，リフレクション API を用いて行う．変数メニューから表示するメソッド一覧も，リフレクション API を用いている．すべてのメソッドを表示すると煩雑になるため，規約として covis_ ではじまる名前のメソッドのみを提示している．メソッド呼び出しの引数にオブジェクトが必要な場合は，変数から参照可能なオブジェクトを列挙し，それをドロップダウンリストで選択させる．引数の型が整数 (int) や文字列 (String) の場合は，テキストフィールドを提示し，直接ユーザに入力させる．

メソッドの返却値については，提案システムの視覚化オブジェクトのルートクラス Covis_Object のインスタンスであれば，オブジェクト領域に追加して表示する．プリミティブの場合は，ラッパークラスのインスタンス (例えば int の場合は Integer) が返されるので，toString() メソッドを使って文字列化し，メソッド呼び出しコードの右にコメントの形 (//>) で表記する (**図** 8 参照)．返却値の型が void 以外のときに null が返された場合は，明示的に null を表示する．

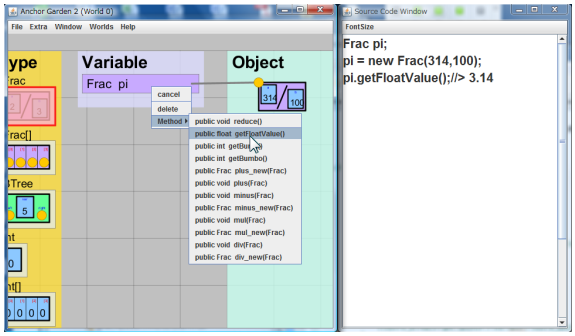

図 8: 返却値の表現

3.2 制限

現時点でのシステム上の制限について言及する．

1. テキストで記述された任意のクラスやメソッドの定義を動的に読み込み，実行する機能はまだ実装していない．提案システムの有効性をより高めるためには，ユーザがクラスを自由に定義し，それをロードして画面内で利用できることが望ましい．例えば付録**リスト** 2 や**リスト** 3 に示すような Java コード記述を与えれば，システム側で独自の視覚化クラスを継承し，必要なインタフェースを実装し，メソッド内でのメンバ変数へのアクセスやコンストラクタ呼び出しを書き替えたのち，動的にコンパイルしたクラスをロードして対応することを検討している．

†BlueJ では汎用性を重視しており，メンバの数や参照の数が多い場合には，BlueJ の「1 オブジェクト 1 ウィンドウ」表現が適していると思われる．

2. 上記の制約により，ユーザが自分でクラスを定義可能な場合でも，継承するクラス (スーパークラス) は既存の Java のクラスを自由に設定できない可能性が高い．このため，GUI 部品 (JFrame, JButton 等) を利用して画面に表示するような場合は，独自のラッパークラスを設計する必要があるかもしれない．

3. 現状では，返却値として新規オブジェクトが生成される場合，それを画面内に出現させるだけで，変数からの参照はユーザの判断で行う必要がある．もし煩雑になる場合は，自動的に参照を行うことも検討したい．関連して，プリミティブ型の返却値の場合，値をソースコード内に，コメントの形 (//>) で表記している (**図 8** 参照) が，この値をプリミティブ型の変数に代入したり，視覚的に表現する機能はない．返却値がプリミティブ型の場合，それを既存の変数に代入したり，新規変数を作成して代入できるようにすることも検討している．

4. toString() メソッドの動作は，Java における実際の動作とは異なる可能性がある．現在のシステムでは，String 型の文字列オブジェクトに対して toString() メソッドを実行すると，呼び出したオブジェクトそのものが返却される．その他 (Frac や BTree 等) への toString() メソッドの呼び出しについては，毎回，文字列オブジェクトを新規作成する．

5. static の概念 (static なフィールドやメソッド) の表現は今後の課題である．

6. ソースコード画面に表示されるような，式を自由に入力して実行する機能がない．提案システムはオブジェクト指向プログラミング初学者で，ソースコードをまだ自在に編集できない学習者を対象としているが，BlueJ[10] や Nigari[8] 等のコード入力環境への移行を円滑に行うためにも，式を直接入力して振舞いを確認するといった機能を提供することにより，段階的な学習を支援したいと考えている．

4 関連研究

BlueJ[10] は，プログラミング初心者の学習を意識して設計された Java 開発環境である．クラスやインタフェースの作成と，継承や関連の定義を UML 図から簡単に行えたり，インスタンスを生成してメソッドを呼び出すといった操作が行える．可視化機能付きのサンプルプログラムを修正して実行することで，オブジェクトの状態を確認することもできるが，基本的にはソースコードを修正しながら振舞いを確認できるユーザを対象としている．Jeliot3[9] は，Java プログラムの実行における変数や配列への代入，条件文などをアニメーションで表示するシステムである．配列も扱うことができ，ステップ/連続実行によりソースコードの意味を理解できる点では有効である．しかし可視化はメソッド実行時の引数渡しや条件分岐に注力されており，オブジェクト間の参照関係をわかりやすく表現するのに向いていない．ドリトル [5] や Alice[3]，The Java Power Tools[6]，PigWorld[7]，Scratch[4] などは，オブジェクトの状態を即座に可視化することによって，プログラムの振る舞いを理解させることに注力している．Nigari system[8] は，クラスやメソッドなどの「おまじない」を記述しなくても動作するという特徴をもつ．またオブジェクトを可視化することによって，初学者のプログラミング意欲を向上させつつ，オブジェクトの概念を理解させることができる．提案システムは，視覚モデルへの操作を通じて，オブジェクト指向の概念やソースコード表記との関連性を意識させることを目的としている．

Myers による INCENSE[11] や，jGRASP のオブジェクトビューア [12] は，プログラムを実行しているときの動的なデータ構造をクラス名やフィールド名から推測し，半自動的に表示できる．しかし，やはりプログラムの動作を確認するための位置付けであり，プログラム実行以外の方法でオブジェクトをインタラクティブに生成したり，データ構造を操作することはできない．

Campbell らの LIVE (Languate-Independent Visualization Environment)[13] は，視覚化されたオブジェクトをインタラクティブに操作してデータ構造を学習したり，対応するソースコードを生成したりすることができる点で，我々のアプローチに類似している．グラフィカルな表現を用いているが，オブジェクト生成やリンクを張る操作はポップアップメニューを使用しており，リンクを直接ドロップして張り替えるといった直感的な操作は提供していない．Anchor Garden[1] は，視覚化モデルに対する比較的自由度の高い操作を許容している．しかし LIVE も Anchor Garden もメソッド呼び出しについては対応していないため，提案システムにおける二分木の構築や探索，返却値オブジェクトの動的な生成と再利用といった内容を扱うことができない．メソッド呼び出しのバリエーション (例えば分数の場合，オブジェクト自身を書き替える場合と，新しいオブジェクトを作成して返す場合) を体験することで，ユーザが自分自身でクラスを定義する際の参考になるものと考えている．

5 試行

2011 年 5 月に，筆頭著者が所属する大学の工学部 3 年生 6 名に対して，オブジェクト指向プログラミングの基本的な考え方を指導する機会があったため，提案システムを利用した．学生らは 2 年生のときに C 言語の基本的な知識は修得していたが，オブジェクト指向についての知識はもっていない．ノートパソコンを 1 人 1 台用意し，筆頭著者が 2.1 で示した「分数クラスを用いた指導例」の流れに沿って，説明

を行いつつ，操作を行ってもらった．最後に文献 [1] の付録 A.3 の問 5(変数とオブジェクトの参照関係を図示する問題) を解かせた．回答中は，教員への質問や提案システムを利用して答えのヒントを得ようとする場面もあったが，結果として半数の学生が 4 問全部合格し，残りの 3 名も 4 問中 2 問合格した．学生から「問題がゲーム感覚で楽しい」という意見がでたため，その 1 週間後に類似の問題 (文献 [1] の付録 A.2 の問 3) を出題し，前回の復習や教員の助言，提案システムの利用無しの状況で解かせたところ，4 名が満点で，残りの 2 名も 3 問中 2 問正解していた．このことから，提案システムを用いた場合，90 分 (講義 1 回分) 程度の時間でオブジェクト指向プログラミングの基本的な考え方を理解できるといえる．今後は，実際にクラス定義やメソッド定義問題を解かせることで，オブジェクト指向独特の設計をどの程度理解できたかを調査する必要がある．

6　まとめと今後の課題

本研究では，初学者が視覚的に表現したモデルに対する操作を行うと，振舞いに対応したソースコードを提示するアプローチを拡張し，メソッド呼び出しによる動的な振る舞いを実行し，観察できるようにした．変数から参照されているオブジェクトに対して，あらかじめ準備されているメソッドをメニューから実行することで，メソッドの振舞い (メンバー変数の値の変化や，オブジェクト間の接続関係) や返却値の様子を観察できる．また，対応するソースコードが逐次表示されるため，学習者はプログラミング実行環境における概念とソースコード表記を結びつけやすくなる．これは，特にソースコードの表記に慣れていない初学者がオブジェクト指向プログラミングのメリットや基本的な発想を理解するのに有効であると考えている．

提案システムを用いると，ソースコードを記述・実行しながら動作確認する環境に比べ，良質な試行錯誤による円滑な導入学習が期待できる．今後は，初学者の概念獲得における提案システムの有効性について検証していく予定である．

提案システムは `http://goo.gl/Muz6t` から実行可能である．

参考文献

[1] 三浦 元喜，杉原 太郎，國藤 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.

[2] Akingbade, A., Finley, T., Jackson, D., Patel, P. and Rodger, S. H.: JAWAA: Easy Web-Based Animation from CS 0 to Advanced CS Courses, *Proceedings of the 34th SIGCSE technical symposium on Computer science education*, pp. 162–166 (2003).

[3] Powers, K., Ecott, S. and Hirshfield, L. M.: Through the Looking Glass: Teaching CS0 with Alice, *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pp. 213–217 (2007).

[4] Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M. and Rusk, N.: Programming by Choice: Urban Youth Learning Programming with Scratch, *Proceedings of the 39th SIGCSE technical symposium on Computer science education*, pp. 367–371 (2008).

[5] 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，情報処理学会誌 プログラミング，Vol. 44, No. 13, pp. 58–71 (2003).

[6] Proulx, V. K., Raab, J. and Rasala, R.: Objects from the Beginning — With GUIs, *Proceedings of the 7th annual conference on Innovation and technology in computer science education*, pp. 65–69 (2002).

[7] Lister, R.: Teaching Java First: Experiments with a Pigs-Early Pedagogy, *Proceedings of the sixth conference on Australasian computing education*, pp. 177–183 (2004).

[8] 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. 9, pp. 25–46 (2004).

[9] Moreno, A. and Joy, M. S.: Jeliot 3 in a Demanding Educational Setting, *Electronic Notes in Theoretical Computer Science (ENTCS)*, Vol. 178, pp. 51–59 (2007).

[10] Kölling, 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).

[11] Myers, B. A.: INCENSE: A system for displaying data structures, *SIGGRAPH Comput. Graph.*, Vol. 17, No. 3, pp. 115–125 (1983).

[12] James H. Cross, I., Hendrix, T. D., Jain, J. and Barowski, L. A.: Dynamic Object Viewers for Data Structures, *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pp. 4–8 (2007).

[13] Campbell, A. E. R., Catto, G. L. and Hansen, E. E.: Language-Independent Interactive Data Visualization, *Proceedings of the 34th SIGCSE technical symposium on Computer science education*, pp. 215–219 (2003).

付録：提案システムが例示に用いるクラスの定義

リスト 2: Frac クラスの定義 (一部)

```
public class Frac {
  int a; //分子
  int b; //分母

  public Frac() {
    a = 2; b = 3;
  }
  public Frac(int ia, int ib) {
    a = ia; b = ib;
  }
  public float getFloatValue(){
    return ((float)a/(float)b);
  }
  public void plus(Frac arg){
    int na = a*arg.b + b*arg.a;
    int nb = b*arg.b;
    a = na; b = nb;
  }
  public void plus_new(Frac arg){
    int na = a*arg.b + b*arg.a;
    int nb = b*arg.b;
    return new Frac(na,nb);
  }
  public void minus(Frac arg){ } // 略
  public void minus_new(Frac arg){ } // 略
  public void mul(Frac arg){ } // 略
  public void mul_new(Frac arg){ } // 略
  public void div(Frac arg){ } // 略
  public void div_new(Frac arg){ } // 略
  public void reduce(){ // 約分
    int g = gcd(a,b);
    a = a/g; b = b/g;
  }
  public int gcd(int m, int n){
    while (m != n) {
      if (m < n) { n -= m; }
      else { m -= n; }
    }
    return n;
  }
}
```

リスト 3: BTree クラスの定義

```
public class BTree {
  int val; //ノードの値
  BTree left; //左
  BTree right; //右

  public BTree() {
    val = 5;
  }
  public BTree(int ival) {
    val = ival;
  }
  public void addNode(int n){
    if (n < val){
       if (left == null)
           left = new BTree(n);
         else left.addNode(n);
    } else {
       if (right == null)
           right = new BTree(n);
         else right.addNode(n);
    }
  }
  public boolean findNode(int n){
    if (n == val) return true;
    if (n < val){
       if (left == null)
           return false;
         else return left.findNode(n);
    } else {
       if (right == null)
           return false;
         else return right.findNode(n);
    }
  }
  public BTree findNodeObj(int n){
    if (n == val) return this;
    if (n < val){
       if (left == null)
           return null;
         else return left.findNodeObj(n);
    } else {
       if (right == null)
           return null;
         else return right.findNodeObj(n);
    }
  }
}
```