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

浅井 俊伍[1,a)]　酒井 三四郎[2,b)]

概要：オブジェクト指向言語である Java の学習は一般的に敷居が高いと言われている．そのため，プログラミングやオブジェクト指向言語の学習を支援するツールはいくつも開発されている．しかし，その多くがクラスとオブジェクトの違いに焦点を当てており，オブジェクト指向の主要な概念である継承・カプセル化・ポリモーフィズムには触れていないという問題がある．そこで本研究では，既存のツールを改良し，オブジェクト指向の主要な概念の学習支援を目的とするワークベンチを開発した．本ツールの特徴は，1）継承関係にあるクラスを色で表現する，2）カプセル化されたクラスとされていないクラスを操作できる，3）動的振る舞いをアニメーションで表現する，の 3 点である．本ツールを利用して学習するためのワークブックを開発し，本ツールの評価を行った．被験者は Java やオブジェクト指向について学習経験のある 8 名の学生であった．被験者を 2 群に分け，従来のテキストと図を用いた学習との比較を行った．その結果，明確に本ツールが有用であったとは言えなかったが，ツールを利用することにより従来のテキストと図を用いた学習にはない人による効果の差が小さいという利点があるという結果が得られた．

キーワード：Java，オブジェクト指向，学習支援，ワークベンチ

Workbench for understanding the key concepts in object-oriented programming language

1. はじめに

オブジェクト指向言語である Java の学習は，学習者にとって一般的に敷居が高い．一般的なプログラミング言語で必要となる変数や関数（メソッド）などの概念に加えて，クラスとオブジェクト（インスタンス）の違いなどのオブジェクト指向の基本的なことからオブジェクト指向の主要な概念（継承・カプセル化・ポリモーフィズム）の理解が必要となるからである．これらの理解が難しいとされている要因の 1 つにその抽象度の高さが挙げられる．通常，これらの概念を理解するためには，書籍などの解説を読み，実際にプログラミングをしながら学習する．しかし，書籍などに記載されているコードを模倣することでプログラムを作成することはできてもその 1 つ 1 つ動作を正しく理解することは難しい．

プログラミングやオブジェクト指向言語の学習を手助けするために抽象度の高い概念の可視化といった方法などを用いて学習支援を行うツールはいくつも開発されている．しかし，その多くがクラスとオブジェクトの違いに焦点を当てており，クラスとオブジェクトの違いやオブジェクトの扱い方までの学習しか支援していない．その結果，オブジェクト指向の主要な概念を学習するためのツールがあまりないという問題がある．

従って，本研究の目的は，オブジェクト指向の主要な概念の学習を支援するツールを開発し，学習支援を行うことである．

2. 先行研究

本章では，プログラミングやオブジェクト指向の学習支援という観点から先行研究を述べる．

三浦らは，Java などの静的な型付けを行うオブジェク

[1] 静岡大学大学院総合科学技術研究科
Graduate School of Integrated Science and Technology, Shizuoka University
[2] 静岡大学情報学部
Faculty of Informatics, Shizuoka University
[a)] asai@sakailab.info
[b)] sakai@inf.shizuoka.ac.jp

ト指向言語を修得するためのワークベンチとして AnchorGarden を提案している [1]．AnchorGarden はオブジェクト指向言語を修得するうえでつまづきやすい概念のうち，抽象度が比較的高いとされている「型・変数・オブジェクトとデータの参照」に焦点を当てている．ワークベンチとは可視化による概念理解の支援に加え，そのモデルを直接操作できるインタラクティブなツールである．三浦らは，AnchorGarden がプログラム作成能力を向上させる可能性があると述べている．

石川らは，AnchorGarden をポリモーフィズムに特化させた enPoly を提案している [2]．AnchorGarden に，1）メソッド呼び出しをアニメーション表示，2） インタフェースとクラスの違いを形で区別，3） リンクの向きの変更，などの変更を加え，ポリモーフィズムの学習支援を行っている．アニメーション表示により，複数の異なるオブジェクトに対し同じメソッド名で呼び出しても，オブジェクトごとに動作が異なるというポリモーフィズムの動作を理解できるようになっている．また，ポリモーフィズムに関連しているインタフェースの概念も理解できるようになっている．

Kolling らは，Java の学習を支援するための統合開発環境（IDE）として BlueJ を提案している [3]．BlueJ の特徴は，クラス図を作成・編集しながらコーディングができ，main メソッドなしに自身が作成したクラスからオブジェクトを生成し，インタラクティブに操作できることである．この特徴により，Kolling らがプログラミング学習のアプローチとして採用している「object-first」の問題点である最初の構文を解決している．「object-first」とは，プログラミング学習をし始める際にオブジェクトの動作から学習をするというアプローチである．「object-first」には最初の構文というオブジェクトを生成するための main メソッドなどの構文をどうするかという問題がある．BlueJ では main メソッドを書かずにオブジェクトを生成し，操作できるためこの問題を解決し，「object-first」のアプローチで学習をすることができるようになっている．

Ben-Ari らは，プログラムの実行の様子をアニメーションで表示するビジュアルデバッガとして Jeliot を提案している [4]．Jeliot の特徴は，自分で作成したプログラムのオブジェクトの生成や消滅，参照の代入などの様子をアニメーションで表示できることである．一連の処理を流れるようにアニメーション表示する以外にも，処理１つ１つを順に実行することができる．この特徴により，学習者は自分の書いたソースコード１行１行の動作を視覚的に見ることができ，その動作を理解することができる．実行時のある時点での変数に保持されている値や，オブジェクト内の変数に保持されている値，オブジェクトの数などを把握できるため，学習支援だけでなくデバッガとしても扱うことができる．

Esteves らは，プログラムの実行の様子をクラスとオブジェクトに絞って表示するツール OOP-Anim を提案している [5]．OOP-Anim の特徴は，Jeliot と同様にプログラム実行時のある時点でのオブジェクトの状態を表示することである．これにより，オブジェクト指向の基本的な概念であるクラスとオブジェクトの違いを理解することができる．

Gestwicki らは，初学者用ビジュアルデバッガとして JIVE を提案している [6]．JIVE は Java で記述されたプログラムを DebugInterface を用いて動作させ，そのビジュアル化を自動で実施するツールである．シーケンス図やオブジェクト図が自動で作成されるため，利用者は視覚的にプログラムの状態を確認することができる．

以上の通り，プログラミングやクラスとオブジェクトの違いといったオブジェクト指向の基本的な概念に焦点を当てたツールは既に開発されている．しかし，オブジェクト指向の主要な概念である継承・カプセル化・ポリモーフィズムに焦点を当てたツールは少ない．焦点を当てたツールもポリモーフィズムのみに焦点を当てており，オブジェクト指向の主要な概念をまとめて学習できるようなツールは開発されていない．そこで本研究では，オブジェクト指向の主要な概念の学習を支援するツールを開発する．

3. AnchorGarden について

本研究では，先行研究で示した AnchorGarden をベースにオブジェクト指向の主要な概念を理解するためのツールの開発を行った．本章では，ベースとなった AnchorGarden の詳細を述べる．

AnchorGarden とは，三浦らが提案したオブジェクト指向言語を学習するためのワークベンチである．AnchorGarden のインタフェースを図１に示す．

AnchorGarden の特徴は次の 3 つがあげられる．

(1) プリミティブ型とオブジェクト型やクラスとオブジェクトを区別できる
(2) オブジェクトをインタラクティブに操作できる
(3) 操作に対応したソースコードを生成することができる

特徴（1）では，クラスとオブジェクトを判子と印鑑のような関係で表し，その違いを表現している．オブジェクト型に代入される参照をアンカー（図１参照）で表現しており，アンカーの有無によりプリミティブ型とオブジェクト型の違いを表現している．特徴（2）では，メソッドの呼び出しなどが行え，メソッドの呼び出し前後でのオブジェクトの状態の変化を見ることができる．また，オブジェクトへの参照が１つも無くなった状態になるとオブジェクトが消滅し，ガーベージコレクションが行われていることを表現している．特徴（3）では，オブジェクトの生成やそのオブジェクトに対する操作に対応したソースコードが生成され，学習者はコードの１つ１つの動作を理解することがで

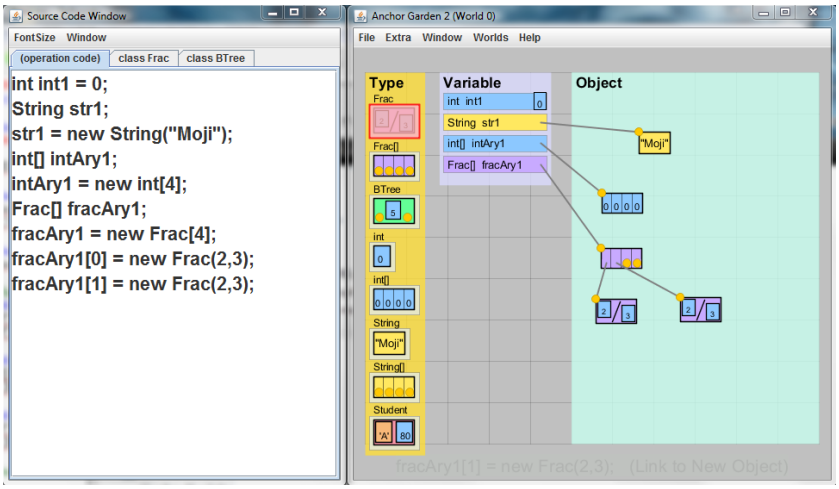

図 1　AnchorGarden のインタフェース
Fig. 1　AnchorGarden's interface

きる．

このような特徴により，オブジェクト指向言語を学習することができるようになっている．

4.　提案ツール：AnchorGardenPlus

4.1　設計目標

本研究の目的は，オブジェクト指向の主要な概念（継承・カプセル化・ポリモーフィズム）の学習を支援するツールの開発である．提案するツール AnchorGardenPlus は，AnchorGarden をベースとし，継承・カプセル化・ポリモーフィズムそれぞれの概念の理解が促進される機能を実装することが目標である．

4.2　対象

AnchorGardenPlus を用いる学習者は，以下のようなレベルの学習者を想定して開発した．

(1) Java の学習をし始め，クラスなどを作成することができ，これからオブジェクト指向を学習しようとしている学習者
(2) オブジェクト指向について学習したが，あまり理解できていない学習者

4.3　実装

オブジェクト指向の主要な概念を学習するために継承・カプセル化・ポリモーフィズムそれぞれの理解を促す機能を追加した．本節では，それぞれの機能について述べる．

4.3.1　継承

継承の概念を理解するための機能の実際の表示例を図 2，図 3 に示す．継承の概念を理解するためには，フィールドがどのクラスで定義されたかを明確にできればいいと考え，図 2 のように，AnchorGarden のオブジェクトに色付けを行った．何も継承していないクラスは単色で表示されるようにし，継承したクラスはその親クラスの色と子クラス独自の色の 2 色で表示されるようにした．これにより，オブジェクトが持つフィールドがどのクラスで定義されたものかが直感的にわかるようになる．そして，子クラスは親クラス＋α という印象をあたえることができる．色付けにより，アクセスできるフィールドが変更されることも表現した．親クラスの変数に子クラスのオブジェクトを代入した場合，子クラスで定義されたフィールドにはアクセスできなくなる．このような状態になったとき，図 3 のように，子クラスの部分が暗転することでアクセスが出来ないことを表現した．

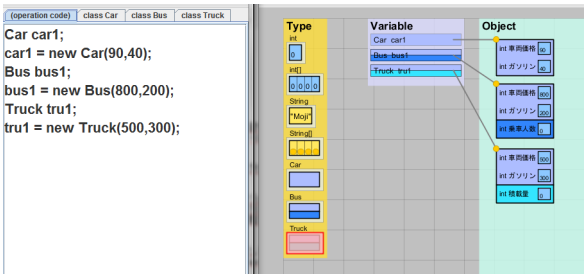

図 2　継承を学習するための機能の表示例
Fig. 2　Examples of the function to learn the Inheritance

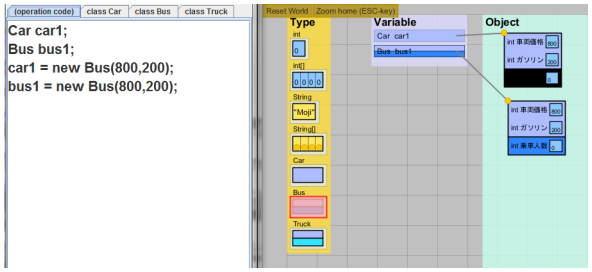

図 3　親クラスの変数に子クラスのオブジェクトを代入した時の暗転例
Fig. 3　Example of when you assign an object of child class to a variable of the parent class

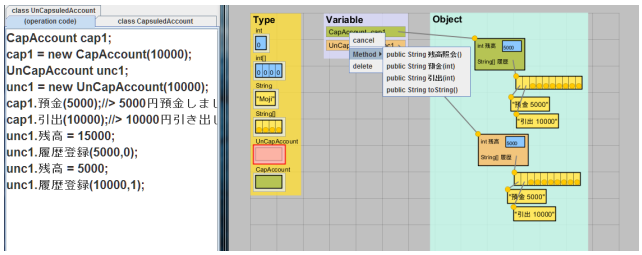

図 4　カプセル化されたクラスの呼び出せるメソッド一覧

Fig. 4　Method summary that can be called a encapsulated class

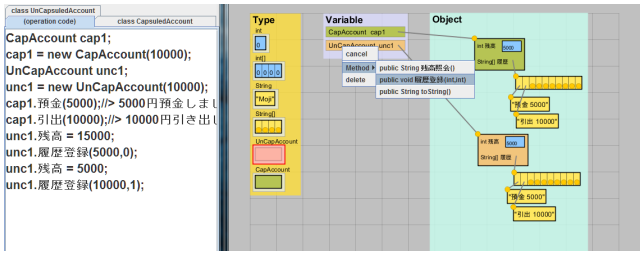

図 5　カプセル化されていないクラスの呼び出せるメソッド一覧

Fig. 5　Method summary that can be called a class that has not been encapsulated

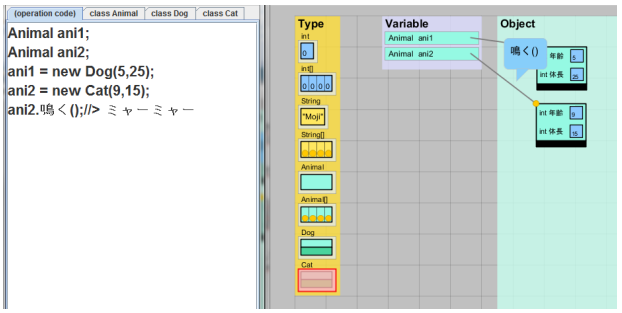

図 6　ポリモーフィズムを学習するための機能の表示例

Fig. 6　Examples of the functions for learning Polymorphism

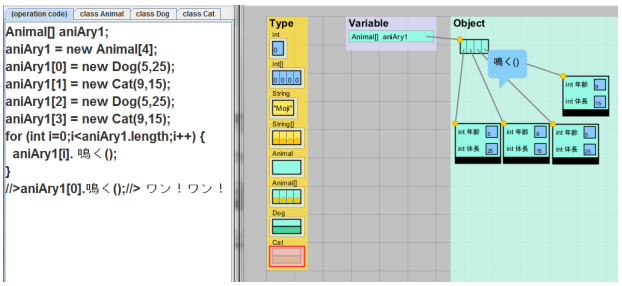

図 7　ポリモーフィズムの有効な利用方法の表示例

Fig. 7　Example of effective usage of Polymorphism

4.3.2　カプセル化

カプセル化の概念を理解するための機能の実際の表示例を図 4，図 5 に示す．カプセル化の概念を理解するためには，カプセル化されたクラスとされていないクラスを操作できればわかりやすいと考え，カプセル化されたクラスとされていないクラスを用意した．カプセル化されたクラスでは，クラスのメンバ変数に直接値を代入できないようになっており，変数の値を変更するためのアクセスメソッドが用意されている．カプセル化されていないクラスでは，アクセスメソッドがなく，クラスのメンバ変数に直接値を変更できるようになっている．上手にカプセル化されたクラスは，不適切な操作ができないようになっており，エラーが起きにくくなっている．カプセル化されたクラスとされていないクラスをそれぞれ操作し，どちらが適切に使われやすいかを考えてもらうことで，カプセル化のメリットを知ってもらう．

4.3.3　ポリモーフィズム

ポリモーフィズムの概念を理解するための機能の実際の表示例を図 6 に示す．ポリモーフィズムの概念を理解するためには，その動作のイメージを持てればいいと考え，メソッド呼び出しをアニメーション表示にした．これは AnchorGarden をポリモーフィズムに特化させた enPoly の機能を参考にした．これにより，どのオブジェクトに対しどのメソッドを呼び出しているかを強調し，異なるオブジェクトに対し同一メソッドを呼び出しているにもかかわらずその動作が異なるというポリモーフィズムの動作を表現する．

ポリモーフィズムを有効に利用する方法を表示する実際の表示例を図 7 に示す．ポリモーフィズムを有効に利用する方法を紹介するために，配列に代入されたすべてのオブジェクトに対し同じメソッドを呼び出す繰り返し実行機能を付け加えた．これにより，複数種類の子クラスのオブジェクトを親クラスの配列で一括に操作できるというポリモーフィズムの有効な利用方法を知ってもらう．

4.4　AnchorGardenPlus のインタフェース

AnchorGardenPlus の各部の名称を図 8 に示す．AnchorGardenPlus を起動すると，図 8 のようにウィンドウに Type，Variable，Object の 3 つのフィールドが現れる．Type フィールドは使用するクラスを選択するフィールドであり，メニューバーの Extra からモードを変更することでフィールドに表示するクラスを変更することができる．Variable フィールドは変数を宣言するフィールドである．Object フィールドは，オブジェクトを生成するフィールドである．

4.5　AnchorGardenPlus での学習方法

4.5.1　継承

継承を学習するには，用意されているクラスからオブジェクトを生成し，それらの持つ変数がどのクラスで定義されているかを確認してもらう．用意されたクラスは親クラスは車を，子クラスはバスとトラックをモデルに作成されており，それぞれのクラスで変数が数個定義されている．生成されたオブジェクトはクラスごとに色付けがされ

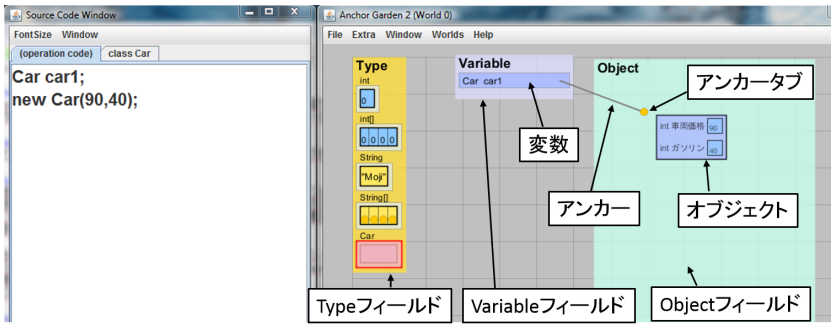

図 8　AnchorGardenPlus の各部の名称
Fig. 8　Nomenclature of AnchorGardenPlus

ているため，変数がどこで定義されているかが視覚的に分かるようになっている．2 種類の子クラスで同じ色の部分がある場合，その部分は共通の親クラスであるということが分かる．親クラスの変数に子クラスのオブジェクトを代入し，子クラスで定義された部分が暗転することを確認してもらう．暗転することでその部分の変数にアクセス出来ないことを表現し，アクセスできなくなるということを理解してもらう．

ここで示した学習方法はあくまで一例である．本ツールはワークベンチとして開発しているため，様々な試行錯誤ができるようになっている．ワークベンチ上でできる操作はコードとして記述できる操作で，できない操作はコードとして記述できない（コンパイルエラーになる）操作となる．学習者はここで示した学習方法だけでなく，自由に試行錯誤をして学習することを期待している．

4.5.2　カプセル化

カプセル化を学習するには，用意されているクラスからオブジェクトを生成し，それらを操作してもらう．用意されたクラスは，銀行の口座をモデルとしたクラスになっており，変数として残高を持っている．カプセル化されてないクラスでは残高変数に直接値を代入でき，カプセル化されたクラスは代入できないようになっている．残高変数が預金・引出以外で値が変更されるという不適切な操作がカプセル化されたクラスではできないといったカプセル化されたクラスのメリットを理解してもらう．

4.5.3　ポリモーフィズム

ポリモーフィズムを学習するには，用意されている親クラスの変数を 2 つ宣言し，そこに異なる子クラスのオブジェクトを代入する．親クラスは animal クラス，子クラスは dog クラスと cat クラスとなっており，子クラスでは鳴くメソッドがオーバライドされている．そして，それぞれのオブジェクトに対し鳴くメソッドを呼び出し，代入されているオブジェクトの種類によって鳴くメソッドの動作が異なっていることを確認してもらう．実際には，dog オブジェクトは「ワンワン」，cat オブジェクトは「ミャーミャー」と表示される．これにより，同じメソッドを呼び出しているにもかかわらずその動作が異なるというポリモーフィズムの動作を理解してもらう．

4.6　ワークブックの作成

AnchorGardenPlus を利用して学習するためのワークブックを作成した．オブジェクト指向の主要な概念の概要や Java での書き方，メリットなどをテキストで解説したあと，AnchorGardenPlus を利用して具体例を操作しながら理解を深めてもらうようにした．なお，このワークブックには練習問題などは含まれていない．ワークブックの構成を以下に示す．

- クラスとオブジェクトのおさらい
- 継承
  - 継承とは
  - 継承の書き方
  - どんな時に役に立つか
  - AnchorGardenPlus を使っての解説
  - 親クラスと子クラスの代入関係
- カプセル化
  - カプセル化とは
  - カプセル化の書き方
  - どんな時に役に立つか
  - AnchorGardenPlus を使っての解説
- ポリモーフィズム
  - 前準備
  - オーバライド
  - 抽象クラス
  - ポリモーフィズムとは
  - ポリモーフィズムの書き方
  - AnchorGardenPlus を使っての解説
  - どんな時に役に立つか

5.　評価実験

5.1　仮説

本実験で検証する仮説を以下に示す．

仮説　AnchorGardenPlus を利用することで，従来のテキ

図 9　変更前のワークブックのポリモーフィズムの解説

Fig. 9　Explanation of polymorphism of the pre-change workbook

ストと図での学習方法より高い学習効果がある

5.2　実験計画

本実験は AnchorGardenPlus を利用して学習を行う実験群と，AnchorGardenPlus を利用しないで学習を行う統制群を用いた計画で実施された．学習効果を測定するために学習を行う前にプレテストを，学習を行った後にポストテストを実施し，そのテストの点数の差によって測定した．

5.2.1　被験者と振り分け方法

理科系大学生 8 名に対して実験を行った．被験者は Java を用いたプログラミングの講義を受講しており，オブジェクト指向という観点からの主要な概念（継承・カプセル化・ポリモーフィズム）の学習経験もある．被験者を実験群 4 名，統制群 4 名に振り分け，実験を行った．このとき，実験群と統制群の振り分けは無作為に行い，学力の違いなどは考慮していない．

5.2.2　ワークブックの変更

本実験を行う上で被験者は学習を行う．実験群は本研究で作成したワークブックに従い AnchorGardenPlus を利用して学習を行い，統制群はワークブックのみで学習を行う．このとき，AnchorGardenPlus の利用の有無による学習量の差をなくすために，統制群で用いるワークブックに変更を加えた．具体的には，ワークブックの AnchorGardenPlus を用いる部分を削除し，代わりに具体例のソースコードを載せ，その具体例を用いて解説を行うようにした．以下にポリモーフィズムの解説の部分の変更前のワークブックと変更後のワークブックを図 9，図 10 にそれぞれ示す．

5.2.3　テスト

学力の測定ををを行うためのテストを作成した．テストはオブジェクト指向の主要な概念（継承・カプセル化・ポリモーフィズム）のそれぞれの理解度を測るために概念ごとに問題を作成した．問題は基礎問題と応用問題を作成し，基礎問題は記述の是非を問う選択問題に，応用問題は概念

図 10　変更後のワークブックのポリモーフィズムの解説

Fig. 10　Explanation of polymorphism of the post-change workbook

に応じた記述問題にした．プレテストとポストテストは似たような構成にし難易度が同じになるように作成した．

5.2.4　実験手順

実験の手順を以下に示す．各テストと学習の間に休憩はなく連続して行った．括弧内は想定している時間を示す．

(1) 実験の説明 (1 分)
(2) プレテスト (5 分)
(3) 学習 (任意・15〜20 分程度を想定)
(4) ポストテスト (5 分)
(5) インタビューなど (4 分)

6.　実験結果

6.1　実験結果

6.1.1　テストの結果

被験者 8 名のプレテストとポストテストの結果とその平均点を表 1，表 2 に示す．また，概念ごとにまとめたプレテストとポストテストの結果とその平均点について，継承を表 3，表 4 に，カプセル化を表 5，表 6 に，ポリモーフィズムを表 7，表 8 にそれぞれ示す．

表 1，表 2 は本実験で行ったテストの全体の結果である．両群におけるプレテストとポストテストの差の平均に対し，有意水準 5%で t 検定を実施したところ p=0.68 ＞ 0.05

表 1　実験群のプレテストとポストテストの結果（全体）

Table 1　The results of pre-test and post-test of the experimental group(All)

被験者	プレテスト	ポストテスト	差
A	40	45	5
B	30	65	35
C	50	85	35
D	40	80	40
平均	40.0	68.8	28.8

表 2 統制群のプレテストとポストテストの結果（全体）
Table 2 The results of pre-test and post-test of the control group(All)

被験者	プレテスト	ポストテスト	差
E	65	80	15
F	15	30	15
G	30	85	55
H	15	65	50
平均	31.3	65.0	33.8

表 3 実験群のプレテストとポストテストの結果（継承）
Table 3 The results of pre-test and post-test of the experimental group(Inheritance)

被験者	プレテスト	ポストテスト	差
A	0	20	20
B	20	25	25
C	10	30	20
D	10	25	15
平均	10.0	25.0	15.0

表 4 統制群のプレテストとポストテストの結果（継承）
Table 4 The results of pre-test and post-test of the control group(Inheritance)

被験者	プレテスト	ポストテスト	差
E	25	25	0
F	5	15	10
G	10	30	20
H	5	20	15
平均	11.3	22.5	11.3

表 5 実験群のプレテストとポストテストの結果（カプセル化）
Table 5 The results of pre-test and post-test of the experimental group(Encapsulation)

被験者	プレテスト	ポストテスト	差
A	10	15	5
B	0	30	30
C	10	30	20
D	10	30	20
平均	7.5	26.3	18.8

表 6 統制群のプレテストとポストテストの結果（カプセル化）
Table 6 The results of pre-test and post-test of the control group(Encapsulation)

被験者	プレテスト	ポストテスト	差
E	10	30	20
F	0	5	5
G	0	30	30
H	0	20	20
平均	2.5	21.3	18.8

となり有意差は見られなかった．プレテストの結果を見ると，実験群と統制群ともに平均点が 50 点を下回っており，

表 7 実験群のプレテストとポストテストの結果（ポリモーフィズム）
Table 7 The results of pre-test and post-test of the experimental group(Polymorphism)

被験者	プレテスト	ポストテスト	差
A	30	10	-20
B	10	10	0
C	30	25	-5
D	20	25	5
平均	22.5	17.5	-5.0

表 8 統制群のプレテストとポストテストの結果（ポリモーフィズム）
Table 8 The results of pre-test and post-test of the control group(Polymorphism)

被験者	プレテスト	ポストテスト	差
E	30	25	-5
F	10	10	0
G	20	25	5
H	10	25	15
平均	17.5	21.3	3.8

オブジェクト指向の主要な概念に対する理解が十分ではないことが分かる．ポストテストの結果を見ると，実験群と統制群ともに平均点が伸びており，オブジェクト指向の主要な概念に対する理解が深まったことが分かる．プレテストとポストテストの差をみてみると，統制群のほうが実験群よりも点数が伸びていることが分かる．このような結果になった原因は 7 章の考察で述べる．

表 3，表 4 は本実験で行ったテストの継承の部分だけの結果である．実験群の方では，全員がポストテストで 20 点を超えており十分理解できていることが分かる．また，平均点でも実験群のほうが高く，ツールを用いた学習が効果的であったといえる．

表 5，表 6 は本実験で行ったテストのカプセル化の部分だけの結果である．プレテストの結果を見ると，実験群と統制群ともに点数が低く，学習前の理解度が低いことが分かる．ポストテストを見てみると，実験群と統制群ともに点数が伸びが同程度だが，実験群は 4 名中 3 名がポストテストで満点となっており，ツールを用いた学習が効果的であったといえる．

表 7，表 8 は本実験で行ったテストのポリモーフィズムの部分だけの結果である．プレテストの結果を見ると，実験群と統制群ともに点数が高く，学習前の理解度が高いことが分かる．ポストテストの結果を見ると，実験群ではほとんど点数が伸びず，下がっている被験者もいる．統制群も実験群ほどではないが，あまり点数がのびていない．ポリモーフィズムに対する理解は実験群と統制群ともに学習前からの理解度が高かったため，ポストテストのポリモーフィズムに対する問題に点数が伸びなかった原因があると考える．

6.1.2 被験者の意見

被験者の方からいただいた意見を以下に示す．

- 講義より分かりやすい（実験群・統制群ともに）
- ツールを使ったほうが分かりやすい（実験終了後，統制群の被験者に AnchorGardenPlus を紹介したとき）
- ツールの使い方でわからないところがあった

7. 考察

7.1 平均点の比較

表 1，表 2 から，実験群の平均点の差が 28.8，統制群の平均点の差が 33.8 と統制群の方が差が大きいということがわかる．実験群のほうが差が小さくなった原因は，実験群と統制群ともに継承とカプセル化の点数の伸びが同程度であることに対し，実験群のポリモーフィズムの点数が下がったことである．このような結果になった原因としてポストテストのポリモーフィズムに対する問題に原因があると考えられる．本実験で用いたポストテストのポリモーフィズムに対する問題にはコードを記述する問題が用意されていた．本学部の講義で実施されるテストではこのような問題が少ないため，本学部の学生である被験者がこのような問題に慣れていない可能性がある．そして，解答をする際，統制群のほうが似たようなコードをワークブックの具体例で目にしており，見よう見まねで書けたのに対し，実験群ではコードを見る機会が少なく，動作は理解していたがコードが書けなかったのではないかと考える．また，プレテストのポリモーフィズムの結果が他 2 つの概念よりも高く，問題の難易度が低かった可能性がある．その結果，ポストテストのポリモーフィズムの点数が下がり，全体の平均点を下げ，統制群の方が平均点の差が大きくなったのではないかと考えられる．

7.2 個々の差

表 1，表 2 から，それぞれの点数の差の標準偏差を求めると実験群は 13.9，統制群は 18.8 となる．このことから，実験群は統制群に比べて標準偏差が小さく点数の伸びのバラつきが小さいことがわかる．このことから，ツールを用いた学習はテキストでの学習より向き不向きが小さいと考えられる．

7.3 今後の課題

本研究で行った実験では，被験者が少なく，学習前の学力を考慮せず 2 群に分けており，偏りのある結果になってしまっている．このため，被験者数を増やし，同じような学力になるように 2 群に分け実験を行う必要があると考える．ポストテストの問題も実験群と統制群の両方の学習内容によって影響のないような問題を考える必要があると考える．

AnchorGardenPlus の機能の拡張としては，オブジェクト指向の主要な概念だけではなく，Java の理解が難しい概念などを支援できるような機能を付け加えていきたいと考えている．1 つのツールだけで，様々な概念の理解を支援できるようになることを期待する．

8. おわりに

本研究では，オブジェクト指向の主要な概念（継承・カプセル化・ポリモーフィズム）に焦点を当てた学習支援ツールがないという問題に対し，三浦らによって提案されているワークベンチ AnchorGarden の機能を拡張し，オブジェクト指向の主要な概念の学習支援ツールの開発を行った．

AnchorGardenPlus の特徴は，1）継承関係にあるクラスやアクセスできるフィールドを色で表現される，2）できる操作とできない操作で表現したカプセル化されたクラスを操作できる，3）動的な振る舞いがアニメーション表示される，の 3 点である．

ツールの評価実験を行った結果，明確にツールが有用であり，高い学習効果があるとは言えないが，ツールを利用することにより従来のテキストと図を用いた学習にはない人による効果の差が小さいという利点があることが分かった．

今後の課題として，実験方法などを見直したり，被験者を増やしたりし適切にツールの評価を行う必要がある．また，学習効果の低かった概念の学習方法を見直す必要もある．

謝辞 三浦元喜教授に，AnchorGarden のソースコードの提供いただきました．感謝いたします．

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(2009)

[2] 石川 祐希子，松澤 芳昭，酒井 三四郎 :“enPoly:オブジェクト指向言語におけるポリモーフィズムの概念を理解するためのワークベンチ”，教育システム情報学会誌，Vol.31，No.2，pp.208-213(2014)

[3] Michael Kolling and Bruce Quig :“The BlueJ system and its pedagogy”, Computer Science Education, Vol.13, No.4，pp.249-268(2003)

[4] Andres Moreno, Niko Myller, Erkki Sutinen et al. :“Visualizing Programs with Jeliot3”, Proceedings of the Working Conference on Advanced Visual Interfaces-AVI’04, pp.99-104(2005)

[5] Micaela Esteves and Antonio Mendes :“OOP-Anim: a system to support learning of basic object oriented programming concepts”，Proceedings of the 4th International Conference on Computer System and Technologies -CompSysTech’03，pp.573-579(2003)

[6] Gestwicki，P. and Jayaraman， B. :“Methodology and architecture of JIVE”，SoftVis’05 Proceeding of the 2005 ACM Symposium on Software Visualization, pp.95-104(2005)