Vol.2017-CE-139 No.5
2017/3/11

オブジェクト指向プログラミング学習支援アプリケーションの開発

相馬 侑弥[1]　高野 辰之[2]　小濱 隆司[3]　宮川 治[3]

概要：プログラミング言語にはさまざまな概念があり，そのひとつとしてオブジェクト指向がある．オブジェクト指向でプログラミングをするためには，オブジェクトについての理解が必要となる．ある時点のオブジェクトは UML（Unified Modeling Language）のオブジェクト図を利用することで表現することができる．本研究では，オブジェクト指向の理解を促進するために，オブジェクト図を利用してオブジェクトやオブジェクト間の関係を可視化するアプリケーションの開発をした．このアプリケーションでは，オブジェクトの変化を表現できる．本稿ではアプリケーションの詳細と試用結果に関して報告する．

Developing an Assist-Tool for Learning Object-Oriented Programming

Yuya SOMA[1]　Tatsuyuki TAKANO[2]　Takashi KOHAMA[3]　Osamu MIYAKAWA[3]

1. はじめに

プログラミング言語にはさまざまな概念があり，そのひとつとしてオブジェクト指向がある．この概念の習得は手続き型プログラミングを習得した学習者であっても非常に難しいことが指摘されている [1]．手続き型プログラミングを習得した学習者がオブジェクト指向でプログラミングをするためには，オブジェクトについての理解が必要となる．

オブジェクトはオブジェクト名，属性（属性名と属性の値）とメソッドを持ち，オブジェクトにメッセージを送る (メソッドを呼び出す) ことで，属性の値などが変化する．また，オブジェクトにメッセージを送るためには，オブジェクト間の関係も理解する必要がある．

手続き型プログラミングでは，一連の計算手順が決められた流れで実行される．それに加えてオブジェクト指向プログラミングでは，オブジェクトのクラスや属性の値によって実行される命令や実行順序が変わる．これにより，ステップごとの「属性の値の変化」や「メッセージのやりとり」，「オブジェクト間の関係の更新」が読み取れなくなり理解を困難にしている一因である考えられる．

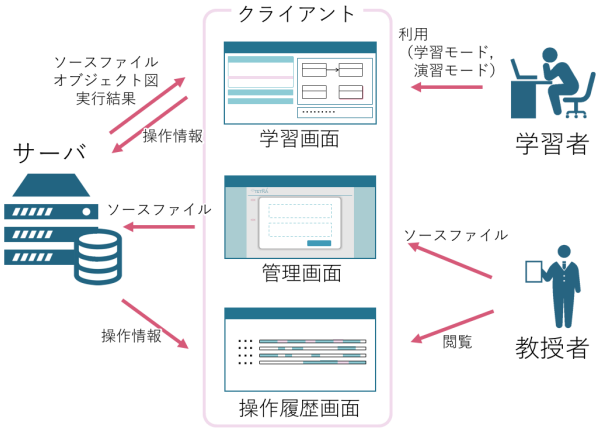

図 1　アプリケーション構成と利用者

これらの問題を解決するために，本研究では UML（Unified Modeling Language）のオブジェクト図とソースコード，実行結果を表示するアプリケーションの開発をおこなう．本アプリケーションでは，ステップに対応する各オブジェクトの値や関係を表現する．本アプリケーションを利

1　東京電機大学大学院情報環境学研究科情報環境学専攻
Graduate School of Information Environment, Tokyo Denki University
2　関東学院大学理工学部
College of Science and Engineering, Kanto Gakuin University
3　東京電機大学情報環境学部
School of Information Environment,Tokyo Denki University

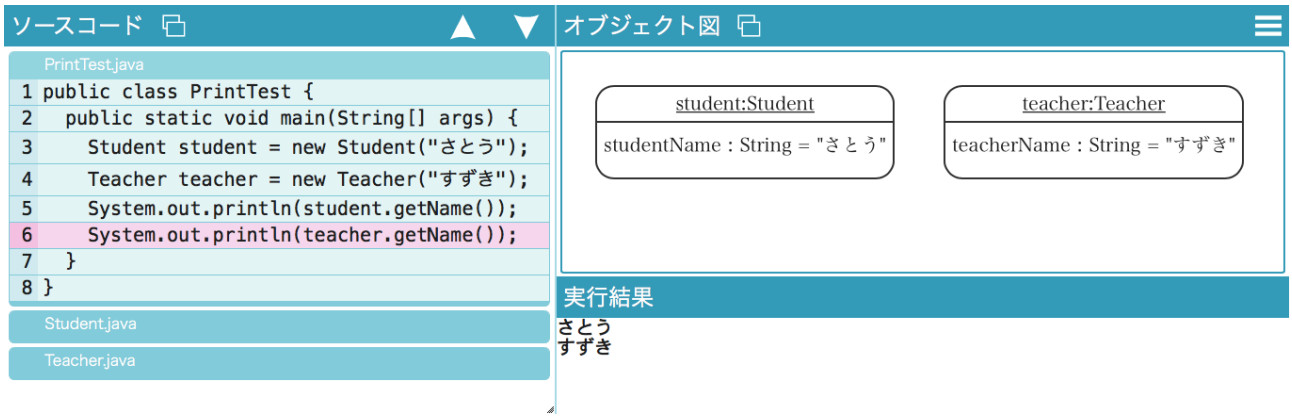

図 2　学習モードの学習画面

用することで，プログラムの実行順序に対応したオブジェクトや実行結果についての学習を支援する．

また，教授者が学習者の利用状況を把握することも支援する．

2. アプリケーション概要

本アプリケーションは，ステップごとにオブジェクトがどのような属性の値を持ち，オブジェクト間にどのような関係があるかを可視化し，オブジェクト指向言語（Java）を学習対象として扱う．

アプリケーションの構成と利用者の関係を図 1 に示す．本アプリケーションはクライアントとサーバで構成される．利用者は学習者と教授者である．クライアントは学習画面と管理画面，操作履歴画面があり，学習者が利用する画面として学習画面（学習モード，演習モード）がある．また，教授者が利用する画面として管理画面と操作履歴画面がある．

学習画面の内容は教授者がソースコードを追加することで自動的に作成される．また，ソースコードとオブジェクト図，実行結果のエリアで構成される．

管理画面はソースファイルをサーバへアップロードし，学習画面を作成するために使用する．ソースファイルは main メソッドが記述されたソースファイルと関連するソースファイルをアップロードする．

操作履歴画面は学習画面での操作を可視化するための画面である．

サーバは学習画面で表示するソースファイルと学習者の操作履歴などを管理している．管理画面から送られてきたソースファイルにより，オブジェクト図の描画に必要な情報を作成する．

3. アプリケーション実装

本アプリケーションのクライアントとサーバの実装について記述する．

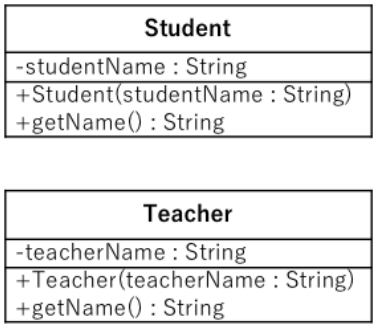

図 3　Student と Teacher のクラス図

3.1 クライアント

クライアントの画面のレイアウトを HTML で実装し，外観やアニメーションなどを LESS [*1] と JavaScript で実装した．ソースコードとオブジェクト図を描画するための情報をサーバから XML 形式で取得している．

3.2 サーバ

アップロードされたソースコードを Java Compiler API を利用してコンパイルする．この時にコンパイルオプションとして-g を利用している．このオプションを利用することでオブジェクトが代入された変数名などを取得できる．次に，コンパイルして生成された class ファイルを実行し，JDI(Java Debug Interface) を使用してオブジェクト図の情報を生成する．このオブジェクト図の情報にはステップやオブジェクトの情報などが記録されている．これらの情報の管理はデータベースの Apache Derby を使用している．

4. アプリケーション詳細

クライアントには学習画面と管理画面，操作履歴画面がある．学習画面の学習モードと演習モードは共にソースコードとオブジェクト図および実行結果のエリアで構成される．モードの違いとして，学習モードでは実行結果の内容がアプリケーションによって表示され，演習モードでは

[*1] 変数や演算，関数を利用して動的な処理を CSS に追加拡張できる言語とその技術．

```
1 public class PrintTest {
2   public static void main(String[] args) {
3     Student student = new Student("さとう");
4     Teacher teacher = new Teacher("すずき");
5     System.out.println(student.getName());
6     System.out.println(teacher.getName());
7   }
8 }
```

図 4 PrintTest.java のソースファイル

```
1 public class Student {
2   private String studentName;
3   public Student(String studentName) {
4     this.studentName = studentName;
5   }
6   public String getName(){
7     return this.studentName;
8   }
9 }
```

図 5 Student.java のソースファイル

```
1 public class Teacher {
2   private String teacherName;
3   public Teacher(String teacherName) {
4     this.teacherName = teacherName;
5   }
6   public String getName(){
7     return this.teacherName;
8   }
9 }
```

図 6 Teacher.java のソースファイル

実行結果の内容を学習者が記述する．操作履歴画面では学習者が表示しているソースファイルを切り替えた時刻と解答を可視化する．

学習モードの学習画面の例を図 2 に示す．この画面では，左側にソースコード，右側にオブジェクト図と実行結果のエリアがある．ソースコードエリアでは main が記述されている PrintTest.java，関連するソースファイルの Student.java，Teacher.java を表示している．Student と Teacher のクラス図を図 3 に示す．これらの 2 つのクラスでは，クラスの 2 段目からインスタンス変数を 1 つ宣言していることが読み取れる．また，インスタンス変数の値はソースファイルの記述からコンストラクタによって変更されることがわかる．

PrintTest.java のソースファイルを図 4 に，Student.java を図 5 に，Teacher.java を図 6 に示す．今回のプログラムでは PrintTest.java の main メソッドを実行する．このプログラムでは，Student クラスと Teacher クラスのオブジェクトがそれぞれ 1 つずつ生成される．また，コンストラクタの引数として “さとう” と “すずき” が設定されているためインスタンス変数に値として代入される．その後，オブジェクトのインスタンス変数の値がそれぞれ出力命令を用いて出力される．

図 2 のオブジェクト図はソースコードを 6 行目まで進めた状態を表している．3 行目で new キーワードを利用して Student オブジェクトを生成し，オブジェクト図に表示される．そして，4 行目で同様に Teacher オブジェクトが生成され，オブジェクト図に表示される．オブジェクトの表示の例として，Student オブジェクトでは上段にオブジェクト名 student，クラス名 Student が，下段に属性名 studentName，属性の値 “さとう”，属性の型 String が記述されている．

実行結果では 5 行目の出力命令によって studentName の「さとう」が，6 行目の出力命令によって teacherName の「すずき」が出力されている．

演習モードの例を図 7 に示す．演習モードを利用する場合，教授者は「PrintTest の最終的な出力結果を記述してください」などの指示をする．それに対して，学習者はソースファイルを読み，実行結果へ「さとう」，改行，「すずき」を入力する．

図 8 に管理画面を示す．この画面は学習画面の内容を作成するために利用される．学習画面を作成するためにビューを作成する．新規ビュー生成のテキストエリアにビュー名を入力し生成ボタンを押す．生成ボタンを押すと図 9 のアップロード画面が表示される．この画面で main メソッドがあるソースファイルと関連するソースファイルをそれぞれアップロードする．決定ボタンを押し問題がなければアップロード済みビューに追加される．

図 7 に対応する操作履歴画面の例を図 10 に示す．この画面で学習画面を利用した際の操作履歴を可視化する．この画面では各ソースコードの開かれた時刻と開いていた時間が表示されている．また，解答内容の変更を見ることができる．この画面ではまず PrintTest.java のソースコードを開き，次に Student.java，PrintTest.java，Teacher.java の順番で開いたことが分かる．また解答では 18 分頃に記述し，27 分頃に記述の変更または追加をしていることが読み取れる．ただし，この例では横軸の時刻はわかりやすいように 0 時 00 分 00 秒から開始している．

これより，学習画面と操作履歴画面の詳細な説明を述べる．

4.1 学習画面

学習画面には学習モードと演習モードの 2 つのモードがある．これらのモードではソースコードエリアとオブジェクト図エリアで同一の表示をし，実行結果エリアはモードによって機能を変えている．以下では，学習画面の各エリ

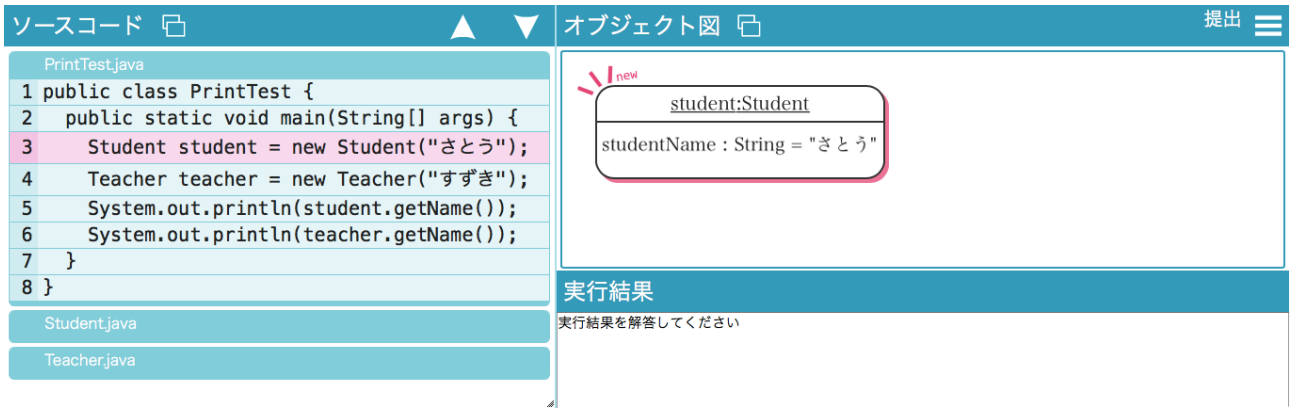

図 7　演習モードの学習画面

図 8　管理画面

図 9　アップロード画面

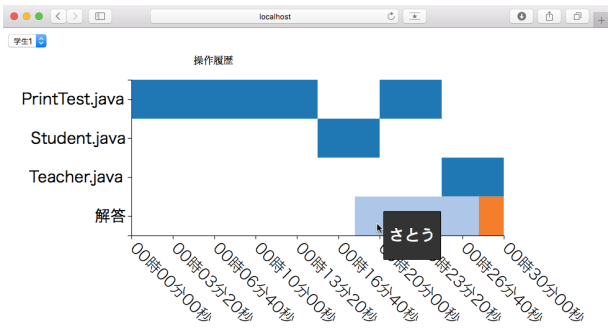

図 10　操作履歴画面の例

アとサーバに送られる情報について述べる．

4.1.1　ソースコードエリア

ソースファイルは教授者によって事前にアップロードされ，ファイルごとに表示されている．それぞれのファイルを，ファイル名とソースコードでグループ化して表示している．ファイル名をクリックすることで，ファイルのソースコードの表示/非表示をアコーディオン形式で切り替えることができる．本アプリケーションでのアコーディオン形式とは，あるファイルをクリックした際にソースコードを広げて表示できる形式である．基本的には，あるファイルが広がる時に既に開いるファイルが閉じる動きとなる．複数のソースコードを表示したい場合，Ctrl キーを押しながらファイル名をクリックする．ファイルの表示順番は教授者が指定した main メソッドが書かれているファイルが最上位の位置に表示され，それ以降はアップロード順とした．また，最上位に表示されているファイルのソースコードは学習画面を利用開始時から開かれている．

ステップは画面かキーボードの矢印ボタンを押すことで移動できる．また，ソースコードの行をクリックすると，その行がプログラムで初めて実行されたステップに移動する．さらに，ファイル名をドラッグ&ドロップすることでファイルの順番を変更することができる．

4.1.2　オブジェクト図エリア

オブジェクト図エリアにはステップに対応するオブジェクトの状態を表示している．オブジェクトはオブジェクト名とクラス名，属性名，属性の値，属性の型を表示する．また，オブジェクト間の関係はリンクの線を表示する．

前のステップからの差分を把握するために強調機能がある．強調されるタイミングと機能は以下の 3 つがある．

- オブジェクトが生成されるとオブジェクト図にオブジェクトが新たに追加される．このオブジェクトの左上に new のマークと影を追加する．
- オブジェクトの属性の値が変更された場合に属性全体を赤色に変更する．
- あるオブジェクトの属性に他のオブジェクトが代入された時にオブジェクト間に関係が生じる．この関係が

表 1　学習画面から送信するデータ

項目名	内容
学習者番号	学習者を特定出来る番号
教材名	学習チャプター名
日時情報	操作が発生した時刻
ステップの移動	移動した先のステップの番号
表示形式の変更	各表示部の表示方法の切り替え
ソースコードの表示	表示されているファイル名
ソースコードの並べ替え	ファイルの表示順番の入れ替え
オブジェクトの移動	オブジェクト図の要素の移動
解答	実行結果エリアの記述内容

```
1  public class Bs100_1 {
2    public static void main(String[] args) {
3      Dog dog1 = new Dog("ポチ");
4      dog1.setWeight(20);
5      Cat cat1 = new Cat("ミルク");
6      cat1.setWeight(3);
7      Cat cat2 = new Cat("ミケ");
8      cat2.setWeight(5);
9
10     dog1.print();
11     cat1.print();
12     cat2.print();
13   }
14 }
```

図 11　Bs100_1.java のソースファイル

```
1  public class Cat{
2    private String name;
3    private int weight;
4
5    public Cat (String name) {
6      this.name = name;
7    }
8    public String getName() {
9      return this.name;
10   }
11   public int getWeight(){
12     return this.weight;
13   }
14   public void setWeight(int weight){
15     this.weight = weight;
16   }
17
18   public void print(){
19     System.out.print(this.name);
20     if (this.weight > 4) {
21       System.out.println("  重い");
22     }else{
23       System.out.println("  軽い");
24     }
25   }
26 }
```

図 12　Cat.java のソースファイル

新たに追加された場合にはオブジェクト間に新たにリンクが引かれる．このリンクの色を赤色で表示する．

また，オブジェクト図のリンクの交差などによってオブジェクトの位置を変更したい場合は図のオブジェクトをドラッグ&ドロップすることで移動できる．

4.1.3　実行結果エリア

実行結果エリアはモードによって内容が異なる．学習モードでは現在のステップまでのプログラムの出力を表示している．学習者はこの内容を見ながらプログラムとの対応関係を学習する．

演習モードの実行結果エリアは学習者がソースコードを読みプログラムの出力内容を記入するために利用される．

4.1.4　学習画面からサーバへ送られる情報

学習画面からサーバへ送られる情報を表 1 に示す．「教材名」は教授者によって決められ，「学習者番号」はログインする際に入力された情報を利用している．また，「日時情報」は操作が起きた時刻を利用した．「ステップの移動」や「表示形式の変更」，「ソースコードの表示」，「ソースコードの並べ替え」，「オブジェクトの移動」，「解答」は「教材名」と「学習者番号」，「日時情報」が付与されて送信されている．

「ステップの移動」は行のクリックや矢印ボタンでステップを移動した際に送信される．「表示形式の変更」はオブジェクト図やソースコードの全画面への表示切り替えが発生した際に送信される．「ソースコードの表示」はソースコード名のクリック後にどのソースコードを開いているかが送信される．「ソースコードの並べ替え」はソースコードの表示順番の変更が行われた際に送信される．「オブジェクトの移動」はオブジェクト図エリアのオブジェクトを移動した際に移動後の位置が送信される．「解答」は実行結果エリアをクリックした時や内容を変更したことがブラウザによって認識できた時にサーバへ解答経過内容として送信がおこなわれる．また，学習者が最終的な解答結果として送信する場合，画面右上にある提出ボタンを押すことで解答を提出できる．

4.2　管理画面

管理画面ではビューの作成と管理ができる．ビューは学習教材ごとに作成され，ビュー名とソースファイル，作成日時を管理する．

ビューの作成はビュー名を入力し，生成ボタンを押す．生成ボタンを押すことでソースファイルをアップロードする画面が表示さる．この画面で実行する main メソッドと関連するソースファイルを分けて追加する．ソースファイルのアップロード方法は各領域にドラッグ&ドロップで追加する．または，領域をクリックして選択ウィンドウを利

```
1  public class Dog {
2    private String name;
3    private int weight;
4
5    public Dog (String name) {
6      this.name = name;
7    }
8    public String getName() {
9      return this.name;
10   }
11   public int getWeight(){
12     return this.weight;
13   }
14   public void setWeight(int weight){
15     this.weight = weight;
16   }
17
18   public void print(){
19     System.out.print(this.name);
20     if (this.weight > 10) {
21       System.out.println("  重い");
22     }else{
23       System.out.println("  軽い");
24     }
25   }
26 }
```

図 13 Dog.java のソースファイル

用する．

アップロード済みのビューから，学習画面に遷移することやビューの削除ができる．

4.3 操作履歴画面

操作履歴画面は履歴を可視化する機能と学習者を選択する機能がある．可視化する内容は学習者が学習画面で参照しているソースコードや解答内容の変更が視覚的に表現されている．この画面によって学習者の操作履歴が解答結果とどのように関連するかを提示することができる．学習者は左上のプルダウンメニューから開き，その中の学習者番号を選択することで表示する学習者の情報を切り替えることができる．

この可視化では横軸を時刻，縦軸をそれぞれのファイル名と解答とする．ソースコードが開閉された時刻を元に帯状で開かれていた時間を表現する．解答はサーバへ送信された学習者の記述内容を元に表示されている．可視化では記述内容ごとに帯を色分けし表示する．それぞれの解答の内容を見るためには，帯上でマウスオーバーすることで解答を確認できる．

5. 学習画面の利用方法

学習画面の学習モードと演習モードの使い方について記

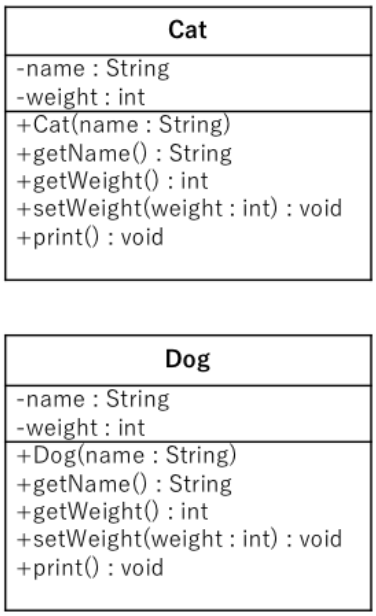

図 14 Cat クラスと Dog クラスのクラス図

述する．学習画面で学習モードを利用するのか演習モードを利用するのかの選択は教授者があらかじめ指定する．学習者は学習画面を利用するために，アプリケーションにログインする．ログインに成功すると学習画面が開かれ，学習できる状態になる．

学習モードでは学習者はソースコードを読みオブジェクト図の状態の変化や実行結果を学習することができる．また，学習者はステップを進めることで，オブジェクトの生成やどのオブジェクトに対して属性の値の変更があるかを視覚的に理解することができる．

演習モードでは，学習者は教授者に「プログラムの最終的な実行結果を記述しなさい」などの問題を出される．学習者はソースコードやオブジェクト図の情報を利用しながら演習を進める．出力命令があり実行結果を記述する場合は実行結果エリアに入力する．

6. 試験運用

授業の演習において本アプリケーションが正しく動作するかを確かめるために試験運用した．課題として「プログラムの最終的な実行結果を記述しなさい」という問題を出題した．

利用したソースファイルは図 11 の Bs100_1.java と図 12 の Cat.java，図 13 の Dog.java を用いた．また，図 14 に Cat クラスと Dog クラスのクラス図を示す．利用したプログラムでは，Dog オブジェクトの属性 name の値を決めて生成し，属性 weight の値を後から設定している．次に Cat オブジェクトも同様の手順で 2 つ生成される．最後にそれぞれのオブジェクトの print メソッドを用いて出力命令が実行される．

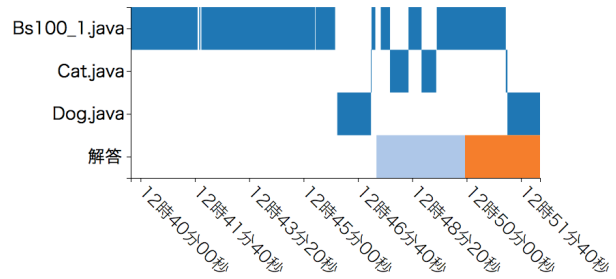

図 15　操作履歴の可視化 正答者の例

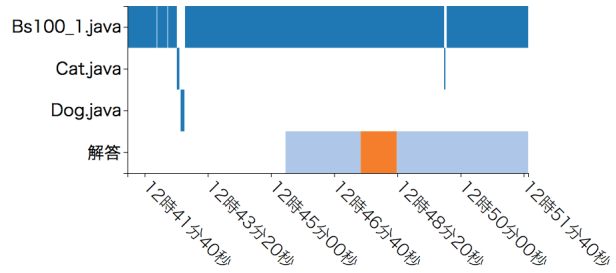

図 16　操作履歴の可視化 誤答者の例

6.1　実施環境

本演習は 2016 年 10 月 5 日に東京電機大学の情報環境学部で開講されているコンピュータプログラミング B の講義で実施した．実施した演習には 9 名が参加した．この講義を履修するためにはコンピュータプログラミング A の単位を取得している必要がある．コンピュータプログラミング A の授業では Java 言語を使って手続き型のプログラミングの考え方を学習している．また，実施日までにクラス図とオブジェクト図の読み方とオブジェクトを生成するソースコードの記述方法については学習済みである．

6.2　実施結果

試験運用では，画面のレイアウトが崩れることやアニメーションが動かないなどの不具合は特になくスムーズに使用できた．演習の結果は，正答した学生は 6 名，誤答した学生は 3 名であった．正答者の例の操作履歴の可視化を図 15 に，誤答者の例の操作履歴の可視化を図 16 に示す．操作履歴画面を見ると学習者の解答やどのソースコードを表示したかを確認することができた．

6.3　考察

操作履歴画面の可視化から学習者のソースファイルの参照順序を読み取れた．これにより，教授者が学習者の学習状況を把握するための機能を提供できたと考えられる．

今回作成した正答者の操作履歴の可視化からプログラムの流れをたどりながらソースファイルを参照していることがわかった．また，誤答者の操作履歴の可視化から読むべきソースファイルがわからないまま解答を記述していることが推測される．

7.　関連研究

本研究では，プログラムの実行順序に対応したオブジェクトの可視化を行った．プログラムの可視化アプリケーションとして Jeliot3[2] が提案されている．Jeliot3 は，ユーザがコーディングした Java プログラムの実行を可視化するものであり，さまざまな概念をアニメーションを用いて表現できる．特に初学者のデータ構造に対する理解を支援している．アニメーションでは，実物の物などを用いて説明を行っている．Jeliot3 と異なり，本研究のアプローチはプログラム実行時のオブジェクトの理解を支援することが目的である．

オブジェクトの理解を促す研究として，プログラム実行時にオブジェクト図を用いて可視化する手法の研究が提案されている [3]．オブジェクト図を用いる点や可視化に関する手法は，本研究と同様のアプローチである．しかし，本研究において注目しているのは，学習者による本アプリケーションの操作履歴であり，その詳細を知ることである点が異なる．

8.　まとめ

オブジェクト指向プログラミングにおいてプログラムの実行順序に対応したオブジェクトや実行結果についての学習支援を目的として，本研究ではソースコードのステップに従って UML のオブジェクト図，実行結果を表示するアプリケーションの開発を行った．

本アプリケーションは，クライアントとして学習者が利用する学習画面と教授者が利用する管理画面と操作履歴画面を提供する．また，サーバではそれらの情報を管理している．学習画面には学習モードの他に演習モードがあり，後者では実行結果エリアが解答欄となり，教授者の出題に対して実行結果を解答し提出する機能がある．

今回，実際のプログラミング講義の演習において本アプリケーションの動作とデータの取得の確認を目的として演習モードを利用した試験運用を行った．その結果，学生 9 名が参加し演習を行った結果ではクライアントの操作や表示に不具合は見られなかった．また，取得した学習者の操作履歴から正答者と誤答者に違いがあることが示唆された．

今後は，本アプリケーションを利用した演習を行い，学習者の操作履歴と解答から特徴を分析していく．

参考文献

[1] Soly Mathew Biju. Difficulties in understanding object oriented programming concepts. In *Innovations and Advances in Computer, Information, Systems Sciences, and Engineering*, pp. 319–326. Springer, 2013.

[2] Andrés Moreno, Niko Myller, Erkki Sutinen, and Mordechai Ben-Ari. Visualizing programs with jeliot 3. In *Proceedings of the working conference on Advanced*

visual interfaces, pp. 373–376. ACM, 2004.
[3] 中原 進, 紫合 治. オブジェクト指向プログラムの動作の可視化. 研究報告ソフトウェア工学（SE）, Vol. 2010, No. 9, pp. 1–8, mar 2010.