第 11 章

オブジェクト指向プログラミング教育における穴埋め問題の改善

本章の内容は以下の論文を元に加筆・修正したものである．
Miyuki Murata, Naoko Kato, Tetsuro Kakeshita, Improvement of Fill-in-the-blank Questions for Object-Oriented Programming Education, Proc. IFIP World Conference on Computers in Education (WCCE 2022). （2022 年 8 月）.

要旨：オブジェクト指向技術は，様々な観点からソフトウェアの品質を向上させるために重要である．私たちは，Java プログラムの穴埋め問題を提供するプログラミング教育ツール「pgtracer」を開発している．また，Java プログラムの穴埋め問題の開発も行った．穴埋め問題は，プログラムとトレース表から構成されている．プログラムとトレース表は，それぞれ Java のクラスとプログラムの実行に対応する．トレース表は，オブジェクト指向プログラムの動作を理解する上で重要であるインスタンス間のメッセージ送受信を表現できる．本稿では，Moodle モジュールの「埋め込み回答（Cloze)」問題を用いて，学生に穴埋め問題を出題する試みを行った．試行の結果について考察し，分析結果に基づいて穴埋め問題および pgtracer のユーザインタフェースを改善した．

キーワード

LA(Learning Analytics)，プログラミング教育，オブジェクト指向プログラミング，Java，穴埋め問題

1. はじめに

近年の情報技術の発展により，多くのアプリケーション領域で IT を活用したサービスが提供されるようになった．これらのサービスが高度化・複雑化する中で，サービスの品質や効率を高めるためには，オブジェクト指向プログラミングの重要性が増している．

大学や高等専門学校では，ソフトウェア工学を学ぶ学生を対象にオブジェクト指向プログラミング教育が行われていますが，教育時間の不足や教員不足により，十分なプログラミング教育が行われていない．そこで，C 言語プログラミングの穴埋め問題や，pgtracer で収集したデータの各種学習分析(LA)機能を提供するプログラミング教育支援ツール「pgtracer」の開発を進めている[1, 2]．これらの LA 機能から得られる知識は，学習効果を高めるために活用できる[3, 4]．

本研究の目的は，Java プログラミング教育における知識を得ることである．この目的のために，pgtracer を Java プログラム用に拡張し，収集したデータを LA 手法で解析している．本論文では，Java プログラム用の穴埋め問題を開発する．

私たちは, Java プログラムとその動作を表すトレース表を利用した pgtracer の穴埋め問題を提案した[5]．トレース表はインスタンスに対応し，インスタンスの変数値や出力，インスタンス間のメッセージ送信を表現する．本アプローチの利点は，トレース表と Java プログラムの両方でブランクを定義できることである. Java プログラムを理解するためには，オブジェクト間のメッセージ送信を追跡することが重要であるため，メッセージ送信を表現するためにトレース表を拡張した．さらに，解答を要しない特殊な空白を導入しました．この空白を利用することで，教師はヒントを隠しつつ，問題の難易度をより柔軟に制御できる．

本論文では，Moodle のアクティビティの 1 つである小テストを用いた実際の授業で，私たちが開発した初期の穴埋め問題を用いたトライアルを行い，問題の妥当性を評価しました． また，トライアルの結果を分析し，穴埋め問題やユーザインタフェースの改良を行う．これらの考察は，Java プログラムへの pgtracer の実装に役立つと思われる．

本論文は，以下のように構成されている．次節では，関連研究について述べるする．3 節では，pgtracer の機能を説明する．4 節と 5 節では，それぞれ Java プログラムに対する穴埋め問題と穴埋め問題の作成について説明する．6 章では，講義と試行について説明し，7 章では評価結果について分析する．8 節では，穴埋め問題の改善と pgtracer のユーザインタフェースについて述べる．最後に結論を述べる．

2. 関連研究

オブジェクト指向プログラミングの教育ツールは、教育目的が異なるものが多く存在している．

Hsiao らは，Java のための Web ベースの問題を提案した[6]．パラメータ化された問題は，変数の最終値や表示されるテキストを解答する．pgtracer は，最終値だけでなく，プログラムの各ステップの変数値にも穴抜きを設定できる．これは学生の理解度を正しく推定するために有効である．

Truong らは Java プログラムの静的解析フレームワークを提案した[7]．学生は与えられた問題のプログラム全体を解答する．このシステムでは，模範解答とあらかじめ定義された差分を解析に利用する．pgtracer では，プログラム全体を穴抜きと定義することで，同様の問題を作成できる．さらに，pgtracer では，全学生の解答ログを分析することで，想定外の誤答の傾向を発見できる．

Hauswirth らは，Java プログラミングを教えるために Informa クリッカーシステムを提案した[8]．Informa は，多肢選択問題などプログラムコードに関連する問題を数種類提供しているが，変数のトレースには対応していない．

Funabiki らは，穴埋め問題を利用した Java のプログラミング教育システムを提案している[9]．しかし，それらはメッセージ送信やプログラム実行の各ステップにおける変数の値を扱うことができない．我々の提案する問題では，これらの部分についても穴抜きを定義できるため，より柔軟な穴抜きの設定が可能である．

3. プログラミング教育支援ツール Pgtracer の穴埋め問題の活用

C言語用pgtracerは，学生に対して穴埋め問題を出題し，学生の解答を自動的に評価する機能を提供する[1]．また，pgtracerは，穴抜きを埋めた直後に，学生の解答，正解，所要時間などのログを自動的に収集する．pgtracerは，収集したログに対して，学生ごとの分析機能，問題ごとの分析機能，穴抜きごとの分析機能，学生の詳細な学習過程の分析機能など，様々な観点からの分析機能を提供する[2]．分析機能は学生の到達度や学習過程の分析に有用である．図1にpgtracerを活用したプログラミング教育プロセスを示す．

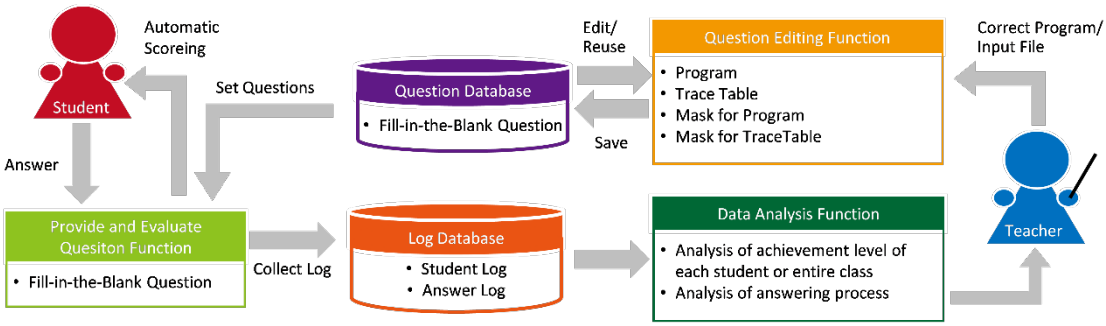

図 1：pgtracer を活用したプログラミング教育プロセス

4. Java プログラムの穴埋め問題穴埋め問題のプログラム

A. Java プログラムの穴埋め問題穴埋め問題のプログラム

ここでは，Java プログラムの穴埋め問題について説明する．図 2 に例を示す．

```
    public class Main {
        public static void main(String[] args) {
            // 'H'を持った CharDisplay のインスタンスを 1 個作る。
1           AbstractDisplay d1 = new    (1)    ('H');

            // "Hello, world."を持った StringDisplay のインスタンスを 1 個作る。
2           AbstractDisplay d2 = new           ("Hello, world.");

            // "こんにちは。"を持った StringDisplay のインスタンスを 1 個作る。
3           AbstractDisplay d3 = new    (2)    ("こんにちは。");

            // d1,d2,d3 とも、すべて同じ AbstractDisplay のサブクラスのインスタンスだから、
            // 継承した display メソッドを呼び出すことができる。
            // 実際の動作は個々のクラス CharDisplay や StringDisplay で定まる。
4           d1.  (3)  ();
5            (4)    ;
6           d3.display();
        }
    }
```

図 2 ：穴抜きを含むプログラムの例 （テンプレートメソッド，メインクラス）

プログラムの表現

Java プログラムの上位コンポーネントはクラスなので, 1 つのクラスに対して 1 つのソースファイルが存在する．したがって，Java プログラムの穴埋め問題では，一般に複数のソースファイルが含まれる．また，インスタンス変数やローカル変数の定義など，初期化文が含まれることがあるため，各文にステップ番号を付与している．ステップ番号は以下の規則に従って割り当てられ，4.B 節で説明するトレース表のステップ番号に対応する．

- Java プログラムで定義されている代入文，メソッドコール，if, else, switch, case, default, return などの制御文やインスタンス変数定義に連番のステップ番号を付す．
- ステップ番号は，各メソッド定義で 1 から始める．
- 入れ子構造を持つ複合文に "x.y "のようなステップ番号を割り当てる．

プログラム内の穴抜き

トークンとは，変数名，クラス名，演算子，キーワードなど，それ以上細分化できない文字列である．コメントはトークンとして定義する．pgtracer は，任意のトークン列に対して穴抜きを定義できる．したがって，文の一部，文全体，連続した文の一部を穴抜きとして定義できる．

穴抜きには 2 種類ある．一つは解答を必要とするもので，もう一つは解答を必要としないものである．後者のタイプの穴抜きを「無視できる穴抜き」と呼ぶ．無視できる穴抜きを定義することで，解答が必要な穴抜きの前後に類似のコードがある場合など，その手がかりとなるコードやコメントを隠すことができる．無視できる穴抜きを導入することで，穴抜きの難易度をより柔軟に制御できる．

また，無視できる穴抜きを使用してコメントを隠すことができるため，問題の難易度を制御できる．解答する穴抜きと無視できる穴抜きを区別するために，これらの穴抜きの背景を異なる色で塗りつぶす．図 2 において，数字のない穴抜きは無視できる穴抜きを表している．

B. トレース表内の穴抜き

トレース表の表現

Java プログラムは，オブジェクト間のメッセージ送信によって実行される．このとき，実行ステップの順序でトレース表を表現すると，表が複雑になる．そこで，オブジェクトごとにトレース表を定義する．各インスタンスは，クラスのインスタンスを区別するために，"ClassName#X "と表現する．例えば，1 番目に生成されたクラス "CharDisplay" のインスタンスの識別子は "charDisplay#1" であり，これがトレース表の名前となる．提案するトレース表は，以下のように定義する．図 3 に例を示す．

Caller of the Method			Step of Called the Method			Argument of the Method	Instance Value	Local Variable of the Method	Output / Return Value of the Method
Caller of			Class	method	step	char	char	int	output
object	method	step				ch	ch	i	
Main	main	1	CharDisplay		1				
Main	main	1	CharDisplay	CharDisplay	1	H	H		
Main	main	(1)	AbstractDisplay	display	1		H		
			(2)	open	1		H		<<
			AbstractDisplay	display	2		H	0	
			AbstractDisplay	display	2.1		H	0	
			(3)	print	1		H	0	H
			AbstractDisplay	display	2		H	1	
			AbstractDisplay	display	2.1		H	1	
			(4)	print	1		H	1	H
(omitted)									
			AbstractDisplay	display	2		H	5	
			AbstractDisplay	display	3		H		
			(5)	close	1		H		>>

図 3：穴抜きを含むトレース表の例
（テンプレートメソッド，インスタンス ID=charDisplay#1）

また，図 3 には含まれていないが，インスタンスが別のインスタンスを作成したり，インスタンスにメッセージを送ったりすると，「外部オブジェクト」のサブ項目として，すべてのインスタンスが列挙される．「メソッドの呼び出し元」欄には，メソッドの呼び出し元のオブジェクト，メソッド名，およびステップ番号が表示される．「メソッドの引数」，「インスタンス値」，「メソッドのローカル変数」列のサブ項目の上段には，データ型またはクラス名が

表示される．サブ項目の最下段は，変数名である．各セルには，基本データ型として格納されている値が表示される．クラスの場合は，インスタンス識別子が表示される．プログラム上，変数に領域が割り当てられていない場合は空欄となる．

オブジェクトは複数回呼び出すことができる．各メソッド呼び出しの一連の処理を区別するために，トレース表には二重線を引く．具体的には，return 文に対応する行の下に二重線を引く．

関数呼び出しの結果を変数に代入するなど，1 つのコードに複数の操作が含まれる場合がある．この場合，トレース表は 2 行で表現するが，これらのステップ番号は同一である．プログラムを完全にトレースできる．トレース表のある行で表される操作は，インスタンス変数と外部オブジェクトの列を参照することで認識できる．

トレース表の穴抜き

トレース表では，各セルの値を穴抜きとして定義する．具体的には，変数値，出力値，ステップ番号，オブジェクト識別子，クラス名，メソッド名に対して穴抜きを定義できる．また，データ型，クラス名，変数名，オブジェクト名にも穴抜きを定義できる．また，プログラムの場合と同様に，トレース表で解答を必要としない無視できる穴抜きを定義できる．

5. 穴埋め問題の作成

本節では，穴埋め問題の開発方針を提案する．作成した問題は，2021 年度前期に「プログラミング演習Ⅲ」で出題する．本講義は佐賀大学理工学部情報部門の 3 年次で実施している講義である．

5.1 穴埋め問題のトピック

被験者は，1 年次に Python，2 年次には C++を学習している．Java を用いたオブジェクト指向プログラミングの学習は，対象講義から始まる．対象講義は，Java 言語を用いたソフトウェア開発の実務で使用される様々なツールの目的や使い方を学ぶことを意図している．授業では，教科書[12]に沿って，Gamma ら[13]が紹介した 23 のデザインパターンを解説する．また，講義では Java 言語に関する詳細な解説は行わない．そこで，Java プログラミングの習得を支援するために，本研究で提案する穴埋め問題を出題する．

穴埋め問題は，教科書から 12 のトピックを選択し，サンプルプログラムを用いて作成した．トピックは授業で扱う内容を考慮して選定し，問題のレベルはトピックの内容や教える順番によって初級，中級，上級とした（表 1）．

表1：出題する穴埋め問題のトピック

レベル	初級	中級	上級
デザインパターン (教科書の項)	TemplateMethod (3) FactoryMethod (4) Iterator (1)	Decorator (12) Strategy (10) AbstractFactory (8)	Adapter (2) Builder (7) Command (22)

B. 穴埋め問題の開発方針

被験者は，これまでの講義でプログラミングの基礎は学習しているため，オブジェクト指向の基礎やデザインパターンに関する問題を作成する．

プログラム内の穴抜きは，主にクラス定義など Java 特有の文法の理解度と，メッセージ送信の実行フローを確認するために設定する．また，トレース表内の穴抜きは，主にクラス，メソッド名，ステップ番号などの呼び出し側オブジェクトに関する項目，呼び出し側オブジェクトに関する項目，メッセージ送信に伴って変化する値について設定する．これらを考慮し，このテーマについて適切なレベルの問題を作成するため，以下の方針を立てる．

1. 1 問につき 10 個程度の解答が必要な穴抜きを設定する．
2. 1 つのテーマに対して，難易度の異なる 2 つの問題を作成する．
3. 各設問の教育目的を明確にする．
4. 各穴抜きの意図を明確にする．
5. プログラム内の穴抜きを設定する際，初級レベルでは個々のトークンを基本として使用する．中級・上級レベルでは，より長いトークンの列を使用する．
6. トレース表内の穴抜きを設定する際，初級レベルでは主に変数値を使用する．中級・上級レベルでは，インスタンス識別子やメソッドなど，より多くの穴抜きを使用する．

これまでの研究[3, 4]より，問題中の穴抜きの数が多すぎると，学生のモチベーションが低下し，継続的に pgtracer を使用できなくなることがわかった．前回の実験では，対象となる学生が C 言語の初心者であり，プログラムの総行数が 20 行程度であったため，1 問あたり 5 つの穴抜きを設定した．しかし，今回の実験では，対象学生のプログラミング習熟度が高いと予想されるため，1 問あたり 10 個程度の穴抜きを設定している．また，同じ理由でプログラムの総行数も多くなっている．

方針 2 は，問題の種類による難易度の違いを調査するために定義する．さらに，方針 3, 4 では，明確にした問題の教育目的，各穴抜きの意図をもとに，学習ログを分析する．方針 5，6 は，初級・中級・上級の違いを明確にするために定義する．

6. 講義での穴埋め問題利用の試行

A Moodle を使った試行

pgtracer の Java 版はまだ開発中のため，Moodle の質問機能を用いた．穴埋め問題のプロ

グラムとトレース表は手動で作成し，画像として表示した．穴埋め問題には複数の Java プログラムとトレース表が含まれるため，それぞれをブラウザ上の別のタブに表示した．解答欄の表現には Moodle の埋め込み回答（cloze）機能を使用した．

本講義では毎週，表 2 に示す問題で試行を行った．問題 ID の PR, TR などの接尾辞は，それぞれ問題の種類がプログラムであること，トレース表であることを示す．右の列は，初回受験者数を示す．Ex01_PR，Ex01_TR，Ex08_SP は，解答方法やトレース表の概念を説明するサンプル問題である．したがって，これらの問題は議論から除外した．

表2：出題する穴埋め問題と受験者数

週	デザインパターン	レベル	問題の種類	問題 ID	穴抜きの数	受験者数
1	穴埋め問題やトレース表を理解するための練習問題		プログラム	Ex01_PR	6	67
			トレース表	Ex01_TR	4	
6	TemplateMethod	初級	トレース表	Ex06_PR	12	71
		初級	プログラム	Ex06_TR	12	
7	FactoryMethod	初級	トレース表	Ex07_PR	9	58
8	簡単な例	初級	プログラム	Ex08_SP	2	67
	Iterator	初級	トレース表	Ex08_TR	10	
10	FactoryMehod	初級	トレース表	Ex10_TR	9	61
11	Composite	初級	トレース表	Ex11_PR	11	69
13	Strategy	中級	プログラム	Ex13_PR	6	66
14	Observer	中級	プログラム	Ex14_PR	10	65

B. 解答結果

表 3 は，問題の種類が「プログラム」の穴埋め問題において，穴抜きが 1 つのトークンからなるタイプと 1 つ以上のトークンからなるタイプの 2 つに分類し，それぞれの穴抜き数と正答率の平均を示す．初級，中級ともに，複数のトークンを含む穴抜きの正答率の平均は，1 つのトークンを含む穴抜きの正答率より低い．これは，学生にとって，複数のトークンを含む穴抜きを解答する方が困難であることを示している．また，1 つ以上のトークンを含む穴抜きについては，いずれも初級レベルより中級レベルの方が正答率の平均が低い．中級では，プログラムの処理フローを理解していないと正解にたどり着けないような穴抜きを設定した．そのため，トークンを含む穴抜きであっても，出題者が意図した難易度を反映できたと考えられる．

表3：問題の種類「プログラム」の穴抜き数と正答率

問題のレベル	問題数	1つのトークンを含む穴抜き		複数のトークンを含む穴抜き	
		穴抜き数	平均正答率(%)	穴抜き数	平均正答率(%)
初級	3	22	76.5	10	57.7
中級	2	11	67.2	9	41.9

表 4 は，問題の種類が「トレース表」の穴埋め問題を，穴抜きの種類別に分類して，穴抜きの数と平均正答率を示す．表 4 から，メソッド呼び出し元のステップ番号とインスタンスの変数値を解答する穴抜きの平均正答率が低いことがわかる．このことから，オブジェクト指向特有のメソッド呼び出しやインスタンスに関する問題は，学生にとって難しいことがわかる．

表4：問題の種類「トレース表」の穴抜き数と正解の数

穴抜きの種類	穴抜き数	平均正答率(%)
コンストラクタやメソッドの呼び出し	16	72.5
呼び出したメソッドのステップ番号	3	46.0
呼び出されたクラス	2	61.7
戻り値（インスタンス）	5	64.0
変数値(インスタンス)	1	38.8
変数値（基本型）	4	71.1

C. アンケート結果

講義の最終日には，今回の試行に関するアンケートを実施した．回答は 69 件であった．穴埋め問題については，63.8%が「どちらかといえば難しい」と回答している．これは，問題の難易度が適切であることを示唆している．穴埋め問題の数は，87.0%の人が適切であったと回答している．無視できる穴抜きによって，ヒントとなりうるコードや変数の値を隠しつつ，適切な穴抜きの数を設定できたと考察される．

さらに，学生から 14 件のコメントを得た．プログラムおよびトレース表の表示方法に関する回答は 5 件あった．今回の試行では，プログラムとトレース表を複数のタブで画像として表示したが，プログラムやトレースメッセージの送受信を理解する際にタブを切り替える必要があり，学生の理解を妨げた可能性がある．また，解答の書き方に関する回答が 3 件ある．入力時に大文字と小文字の違いで不正解になることに対する不満であった．しかし，実際のプログラミングでは，小文字と大文字の違いは致命的となるため，教育上合理的と考え，正しい入力を求めるようにする．

7. 穴埋め問題の改善

本節では，穴埋め問題の改良について述べる．なお，4 節で説明した穴埋め問題のプログラムおよびトレース表を旧プログラムまたは旧トレース表と呼ぶ．

A　プログラムの改善

旧プログラムでは，インスタンス変数定義文にステップ番号も付加されていた．しかし，Java では，インスタンス変数はコンストラクタが呼ばれたときにデフォルト値で初期化され，コンストラクタ内に書かれたステートメントによって値が代入される．このデフォルト値は暗黙のうちに設定されるものであり，プログラム中に明示的に記述されることはない．このため，インスタンス変数定義文へのステップ番号の付与，トレース表へのデフォルト値の記載は，学生を混乱させるため行わないことにした．また，インスタンス変数の代入文でインスタンス変数の値を明示することは，変数値のような不定な状態は存在しないという Java プログラムの方針と一致する．

旧プログラムの作成に参照したサンプルプログラムは，2004 年に発行された教科書のものであり，総称のサポートなど Java 言語の進化に対応したものではなかった．この度，2021 年に教科書の第 3 版が発行され，Java 言語の進化に合わせてサンプルプログラムも更新されている．第 3 版のサンプルプログラムを使用することで，最新の Java に対応した問題を提供できる．

不正解を調べると，準備されている正解以外の正解で穴抜きを埋めてしまい，不正解になっている答案があることがわかった．例えば，「i < 5」が正解の場合，「5 > i」や「i <= 4」と答えてしまい，不正解となるケースがあった．

これを避けるため，正解が一意に定まるように穴抜きの位置を調整した．先ほどの例を改良し，「i <」の部分を穴抜きとせず，「5」の部分にのみ穴抜きを設定することで，もう一つの正解をなくすことができる（図 4）．

```
        public  (1)  class AbstractDisplay {
            // open, print, close はサブクラスに実装をまかせる抽象メソッド
            public  (2)  void open();
            public       void print();
            public       void close();

            // display は AbstractDisplay で実装してるメソッド
            public  (3)  void display() {
1               open();
2               for (int i = 0;  (4)  5; i++) {
2.1                 print();
                }
3               close();
            }
        }
```

図 4：穴抜きの改良（テンプレートメソッド，AbstructDisplay.java）

また，「i＜5」が正しいのに「i<5」となっているなど，空白の有無による不正解も多く見られた．Java のコーディング規約[15]によれば，比較演算子の前後には穴抜きを入れるべきとされている．システム開発では，プログラムの再利用性が求められるが，単にプログラムが正しく動くだけでなく，ガイドラインに沿ったコーディングができることが重要である．よって，不正解として扱うことにした．

B　トレース表の改善

旧トレース表は，プログラムを手動でトレースして作成したものであるが，今回，若干の調整を加えてトレース表自動生成機能［15］が完成したので，この機能によって作成されたトレース表を使用する．この機能は，Java プログラムと入力データを与えることで，Excel ファイル形式のトレース表を出力する．各シートは，出力される Excel ファイルの各インスタンスに対応する．本機能の完成により，トレース表の開発コストが大幅に削減された．

旧トレース表では，インスタンス変数の定義も表示し，その値は不定とされていたが，7.1 節で述べた理由により，トレース表にも記載しないこととした．

C　ユーザインタフェース

旧来の穴埋め問題では，プログラムやトレース表が画像で表示された．その画像はブラウザのタブで表示されていた．学生からのアンケートによると，タブの切り替えが面倒でわかりにくいという回答が複数あった．

そこで，図 5 に示すように，穴埋め問題を出題する際には，ブラウザのフレーム機能を使って，プログラムやトレース表の一覧を表示する．その中から項目を選択することで，当該プログラムやトレース表が表示される．また，折りたたみ機能を利用して，同じクラスのインスタンスの表示・非表示を切り替えることができる．このユーザインタフェースは，pgtracer にも適用できる．

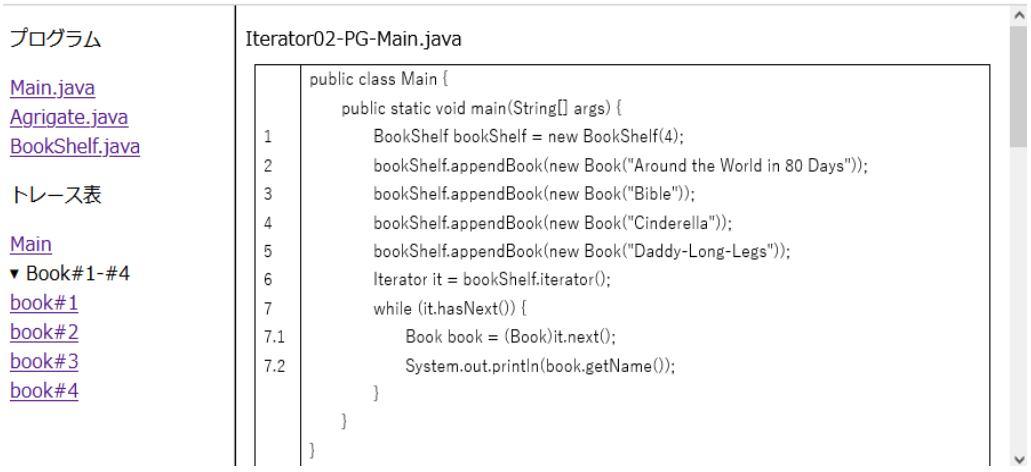

図 5：ユーザインタフェースの改善図

8. 結論と今後の課題

本稿では，穴埋め問題を用いた Java プログラミング教育支援ツール pgtracer の穴埋め問題の改良を行った．改良は試行結果をもとに行った．また，pgtracer のユーザインタフェースを改善し，学生が問題を解く際の難易度を下げる工夫をした．これらの改善により，学生がプログラムやトレース表を理解しやすくなり，継続的な学習が促進されることが期待できる．C 言語用の pgtracer では，学生が解答したプログラムをコンパイルして実行し，出力が同じであれば正解と判定している．このモジュールを Java 言語の pgtracer に組み込むことで，正解の設定をより柔軟にし，問題の難易度を制御できると考えている．

今後は，pgtracer を完成させ，作成した穴埋め問題を用いて実験を行う予定である．収集したデータを用いて，学生の解答傾向や，LA 技術を用いた間違えやすい問題の傾向などを抽出することができ，得られた知見は Java プログラミング教育に貢献できると考えている．

謝辞

本研究は，日本学術振興会科研費補助金（課題番号：20K03265，#20K03233）の支援を受けている．

参考文献

[1] 掛下，柳田，太田: 穴埋め問題を活用したプログラミング教育支援ツール pgtracer の開発と評価，情報処理学会誌:コンピュータと教育, Vol.2, No.2, pp.20-36, (2016).

[2] 掛下，太田: ウェブプログラミング教育支援ツール pgtracer における学生ログ解析機能. 情報処理学会論文誌コンピュータと教育, Vol.5, No.2, pp.456-468, (2019).

[3] Kakeshita, T.， Murata, M. : Application of Programming Education Support Tool pgtracer for Homework Assignment”, International Journal of Learning Technologies and Learning Environments, Vol. 1, No. 1, pp. 40-61, (2018).

[4] Murata, M.， Kakeshita, T. : Analysis Method of Student Achievement Level utilizing Web-Based Programming Education Support Tool pgtacer, In:5th International Conference on Learning Technologies and Learning Environment (LTLE 2016), pp.316-321.（2016）.

[5] Murata, M.， Kato, N.， Kakeshita, T. : Fill-in-the-blank Questions for Objected-Oriented Programming Education, In 10th International Congress on Advanced Applied Informatics (IIAI-AAI), pp.116-122 (2021).

[6] Hsiao, I., Brusilovsky, P., Sosnovsky, S.: Web-based parameterized questions for object-oriented programming. E-Learn’2008: World Conference on E-Learning (2008).

[7] Truong, N., Roe, P., Bancroft, P.: Static analysis of students' Java programs. Sixth Australasian Computing Education Conference (ACE 2004) (2004).

[8] Hauswirth, M., Adamoli, A.: Teaching Java programming with the Informa clicker system. Science of Computer Programming, 78(5), 499-520 (2013).

[9] Funabiki, N., Matsushima, Y., Nakanishi, T., et al.: A Java programming learning assistant system using test-driven development method. IAENG International Journal of Computer Science 40(1), 38-46 (2013).

[10]Zaw, K. K., Funabiki, N., Kao, W.-C.: A proposal of value trace problem for algorithm code reading in Java programming learning assistant system. Information Engineering Express 1(3), 9-18 (2015).

[11]Kyaw, H. H. S., Funabiki, N., Kao, W.-C.: A proposal of code amendment problem in Java programming learning assistant system. International Journal of Information and Education Technology 10(10), 751-756 (2020).

[12]Yuki, H.: An Introduction to Design Patterns using Java Programming Language. 3rd edition. SoftBank Creative (2021). (in Japanese)

[13]Gamma, J., Helm, E., Johnson, R., Vlissides, R.: Design Patterns Elements of Reusable Object Oriented Software. Addison-Wesley Professional (1994).

[14]Code Conventions for Java Programming Language, https://www.oracle.com/java/technologies/javase/codeconventions-contents.html, last accessed 2021/02/25.

[15]Kishikawa, M., Ohtsuki, M., Kakeshita, T. : Generating Trace Table for Java Programs, In 11th International Congress on Advanced Applied Informatics (IIAI-AAI), pp.200-207, 2022.