第 8 章

オブジェクト指向プログラミング教育のための穴埋め問題

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, Fill-in-the-blank Questions for Object-Oriented Programming Education,in Proc. 10th Int. Congress on Advanced Applied Informatics (IIAI-AAI), pp. 116-122.（2021 年 7 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．我々は，C 言語の穴埋め問題を提供するプログラミング教育ツール pgtracer を開発した．実際の授業で pgtracer を使用して収集したデータを分析することで，C 言語プログラミング教育に有用な知見を得ることができた．本稿では，pgtracer をオブジェクト指向プログラミング用に拡張し，Java プログラム用の穴埋め問題を開発する．穴埋め問題は，プログラムとトレース表から構成される．プログラムとトレース表はそれぞれ Java のクラスとインスタンスに対応する．トレース表には，オブジェクト指向プログラムの動作を理解するために重要なインスタンス間のメッセージ送信が含まれる．さらに，学習者が解答する必要のない穴抜きを導入する．これにより，穴抜きを埋めるための学生の負担を軽減しつつ難易度設定の自由度を高めている．評価実験では，提案する穴埋め問題を実際の授業で使用し，収集した学習データを分析することで，学生の解答行動や間違いやすい箇所などを抽出する．得られた知見は，学習成果の測定やオブジェクト指向プログラミング教育の改善への活用が期待できる．

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伴い，さまざまな領域で IT を利用したサービスが提供されている．これらのサービスの高度化・複雑化に伴い，ソフトウェアの品質向上や開発効率向上のために，オブジェクト指向プログラミングの重要性が高まっている．

大学や高等専門学校では，ソフトウェア技術者を育成するためにオブジェクト指向プログラミング教育が行われているが，時間やスタッフの不足により，十分なプログラミング演習を行うことが困難である．そこで，我々はプログラミング演習を支援するための教育支援ツール pgtracer [1, 2]を開発し，収集したデータを活用した学習分析（LA）を行っている．LA から得られる知見を活用することで，学習効果の向上が期待できる．

本研究の目的は，Java プログラミング教育における知見を得ることである．そのために，pgtracer を Java 用に拡張し，LA 法を用いて収集したデータを解析する．得られた知見を活用することで Java プログラミング教育を支援できる．本稿では，Java プログラミングを教えるための穴埋め問題を提案する．

pgtracer の穴埋め問題は，Java プログラムとトレース表から構成されている．トレース表はインスタンスに対応し，そのインスタンス変数と出力値を表現する．我々の研究の革新的な点は，穴抜きをプログラム内だけでなく，トレース表内でも定義していることである．これまでに行った C プログラミング版の pgtracer を用いた評価実験から，トレース表の理解度が高い学生はプログラムの達成度も高いこと，変数値の変化をトレース表で可視化することがプログラムの習得に有用であることがわかっている[3, 4]．Java の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トレースすることが重要であるため，メッセージの送受信を表現できるようトレース表を拡張する．

プログラムやトレース表の中に穴抜きにおいて，プログラムやトレース表の他の部分がヒントとなることがある．このようなヒントを隠すために，多くの穴抜きを定義することが必要な場合がある．しかし，これでは類似した穴抜きが増え，学習者の学習意欲を低下させる可能性がある[4]．そこで，学生が解答する必要のない特殊な穴抜きを導入する．これにより，ヒントを隠しつつ，問題の難易度をより柔軟に制御することが可能となる．

作成する問題は，GoF のデザインパターンに対応する 12 個の Java プログラムを使用する．これらのプログラムは，Java プログラミング言語とオブジェクト指向設計の機能を十分に活用したものであり，オブジェクト指向プログラミング教育に適している．これらのプログラムを 3 つのレベルに分類し，穴抜きの場所を工夫することで穴埋め問題の難易度を調整する．

また，提案した穴埋め問題を実際の授業で活用し，収集したデータに対して LA 法を適用する．授業を受ける学生は，大学で 1 年半 Python や C++を使った構造化プログラミングを学んでおり，Java プログラミングは初めて学ぶ．

本稿の構成は以下の通りである．2 節では関連研究について述べる．3 節では，pgtracer

の機能を述べる．4 節では，プログラムの表現方法とプログラム内の穴抜きの定義について述べる．5 節ではトレース表の表現方法とトレース表内の穴抜きの定義について述べる．6 節では穴埋め問題の難易度を調整するための戦略を提案する．最後の節で結論を述べる．

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数多く存在するが，それぞれのツールにはいくつかの問題点がある．

Hsiao らは，Java のための Web ベースのパラメータ化された問題を提案した[5]．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．最終的な正しさだけでなく，各ステップでの変数値もトレースできることは，プログラムの理解度を推定するのに有効である．

Truong らは Java プログラムの静的解析フレームワークを紹介した[6]．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，想定以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[7]．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[8, 9, 10]．これらの問題では，プログラム文の一部，文全体，出力値のいずれかを穴抜きとして定義しているが，プログラム実行の各段階におけるメソッド送信や変数値を扱うことはできない．本提案問題では，プログラムや出力値だけでなく，メッセージ送受信や変数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．

Java プログラムを用いた LA に関する研究の多くは，プログラムエラーやコンパイル時のエラーを分析している．Edwards らは学生が作成した Java プログラムで発生した静的解析エラーを分析し，Checkstyle と PMD を用いて検出を行った[12]．彼らは，最も一般的なエラーなどの知見を得ている．また，McCall らは，学生用プログラムのエラーを解析し，エラーの分類とその頻度について研究している[13]．Altadmri らは，25 万人以上の学生の 1 年間のコンパイル実行履歴を用いて，エラーの頻度，修正までの時間，ユーザー間でのエラーの分布などを分析している[14]．

pgtracer は，最終的な解答だけでなく，解答中に学生が入力した解答も収集する．これらのデータを利用することで，最終解答に至るまでの学生の行動を分析できる

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

pgtracer は，穴埋め問題の作成機能，問題の提供・評価機能，ログ収集機能，ログ解析機能などを提供する．

1) 穴抜きを埋める問題

穴埋め問題は，プログラム，トレース表，プログラム用マスク，トレース表用マスクの 4 つの XML ファイルから構成される．プログラム中の穴抜きの候補は，トークン，連続したトークン列，式，文である．トレース表の穴抜き候補は，変数値，ステップ番号，変数名である．我々は，穴抜きの位置によって問題の難易度を制御できることを検証している[1, 2]．

2) pgtracer が提供する機能

pgtracer は様々な機能を提供する．一つ目の機能は，穴埋め問題の作成機能である．プログラム，トレース表，プログラムマスク，トレース表の XML ファイルを生成する機能を提供する．二つ目は，穴埋め問題を学生に提供し，学生の解答に対する自動採点機能である．

また，pgtracer は，各穴抜きを埋めた直後の学生の解答，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，様々な観点から分析機能を提供する．

3) pgtracer を用いたプログラミング教育モデル

図 1 は，pgtracer を利用したプログラミング教育の流れを示す．教師は正解プログラムと入力ファイルを用いて穴埋め問題を作成する．学生は問題データベースから問題を選択して解答する．pgtracer は学生の解答を自動的に採点する．同時に，pgtracer は解答ログを収集する．学生と教師は，解析機能を使って解答ログを解析できる．教師は，問題データベースを改善し，学生またはクラス全体にフィードバックできる．

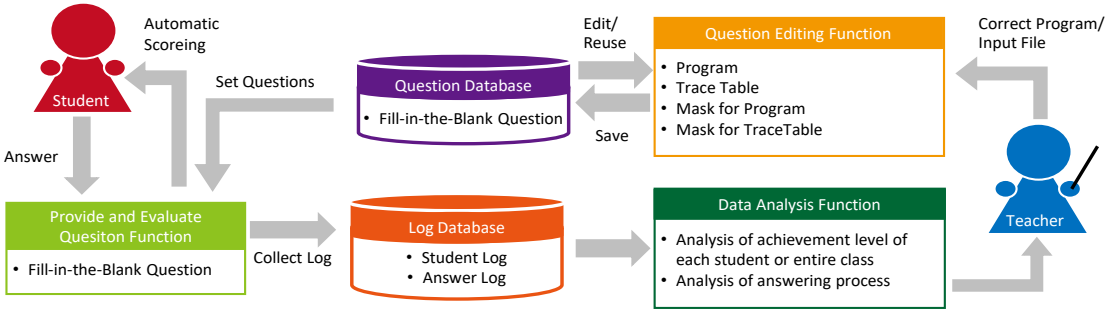

図 1：pgtracer を活用したプログラミング教育プロセス

4. 穴埋め問題のためのプログラム

ここでは，プログラムの表現方法とプログラム内の穴抜きの定義について説明する．図 2 は問題プログラムの例である．

```
    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）

A. プログラムの表現

Java プログラムの最上位の構成要素はクラスであるため, 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．各文にはステップ番号を付す．初期化文が含まれる場合があるため，インスタンス変数やローカル変数の定義などにもステップ番号を付す．ステップ番号は以下の規則に従って割り当てられ，第 5 節で説明するトレース表のステップ番号に対応する．

- インスタンス変数定義を含む各文に連続したステップ番号を割り当てる．
- メソッド定義内の各文に 1 から始まる連続したステップ番号を割り当てる．
- 代入，メソッド呼び出し，if, else, switch, case, default, while, return など，Java で定義された各文に連番を付与する．
- 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．

プログラムは正常にコンパイルされなければならない．また，Java のプログラムについて，ガイドラインを以下に定義する．

- 入れ子構造を正しく表現するためにインデントを使用する．
- クラス定義の前には，そのクラスの概念や特徴を説明する独立したコメントを付け

る．

- メソッド定義の前には，そのメソッドが提供する機能を説明する独立したコメントを付ける．
- 変数定義には，変数に格納された値を説明するコメントを同じ行に付ける．
- 一連の記述の前に，その意図やアルゴリズムを説明する独立したコメントを付ける．
- 独立したコメントの前には，空行を入れる．
- 1 つの文は，可能であれば 1 行で記述する．
- 変数名と関数名はキャメルケースの規則に従う．ただし，クラス名の最初の文字は大文字にする．

B. 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，それ以上分解できない最小の文字列のことである．コメントは 1 つのトークンとして定義される．pgtracer では，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．

穴抜きは 2 種類ある．一つは解答を必要とするもの，もう一つはそうでないものである．後者の穴抜きを「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．

異なるインスタンスに対して，同じメッセージを繰り返し送る場合を考える．文中に穴抜きが定義されていると，穴抜きの近くにある文が穴抜きを埋めるヒントとなってしまう．ヒントとなる部分に無視できる穴抜きを適切に定義することで，学生が解答すべき穴抜きの数を増やさずに，ヒントとなるトークンを隠すことができる．これにより，問題の難易度をより柔軟にコントロールできる．

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．コメントはヒントとなるため，これを隠すことで問題の難易度をコントロールできる．穴抜きと無視できる穴抜きを区別するために，穴抜きの背景を異なる色で塗りつぶしている．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．

5. 穴埋め問題のためのトレース表

A. 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受信によって実行される．このとき，トレース表を実行ステップの順に表現すると，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し，メッセージ送受信をトレース表で表現する．この方法は，シーケンス図のようなオブジェクト指向プログラミングの概念にも適合する．提案するトレース表を以下に定義する．図 3 はその例である．

表 1：各オブジェクトのトレース表の列

列	サブ項目	値
メソッドの呼び出し元	オブジェクト	インスタンス名またはクラス名
	メソッド	メソッド名
	ステップ	ステップ番号
メソッドを呼び出すステップ	クラス	メソッドが定義されているクラス名
	メソッド	メソッド名
	ステップ	ステップ番号
メソッドの引数	上段：データ型，クラス名 下段：引数名	対応するコードを実行したときの引数の値
インスタンス変数	上段：データ型，クラス名 下段：変数名，インスタンス名	値またはオブジェクト識別子
メソッドのローカル変数		
外部オブジェクト	上段 ：クラス名 下段：インスタンス名	オブジェクト識別子
出力/メソッドの戻り値	メソッドの戻り値	戻り値または返されたオブジェクトの識別子
	出力	出力文字列

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		>>

1) インスタンス識別子の表現

トレース表は，インスタンスそれぞれと Main メソッドについて定義する．インスタンスに対応するトレース表の名前はそのインスタンス名とし，Main メソッドを表すトレース表の名前は「Main」とする．

インスタンスは，1 つのクラスに対して複数定義される場合がある．それらのインスタンスを区別するために，インスタンスの名前を"ClassName#X "と表現する．ここで，"ClassName "はクラス名の最初の文字を小文字にした文字列，"X "は生成順を表す連続した

番号である．例えば，最初に生成されたクラス "CharDisplay" のインスタンス識別子は "charDisplay#1" となる．

2) トレース表の列

オブジェクトのトレース表は表 1 に示す列を持ち，Main メソッドのトレース表は「メソッドの呼び出し元」を除いて同じ列を持つ．「メソッドの呼び出し元」列は，呼び出し元のオブジェクトを記述する．オブジェクトはコンストラクタ呼び出しによって生成される．オブジェクトが生成された後，他のオブジェクトはオブジェクトにメッセージを送り，何らかの処理を実行することがある．

トレース表では，現在のオブジェクトがオブジェクトを作成するか，オブジェクトにメッセージを送信すると，すべてのオブジェクトが「外部オブジェクト」列のサブアイテムとしてリストアップされる．サブ項目の値は，呼び出されたメソッドの名前である．そして，プログラム内で生成された各オブジェクトの生存期間とメッセージの流れを理解できる．

静的メソッドを含むクラスのトレース表は，クラス名と同じ名前になる．トレース表には，表 1 に示す項目の他に，「クラス変数」という項目と「データ型」というサブ項目がある．

3) 複数のメソッド呼び出し

オブジェクトは複数回呼び出されることがある．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が引かれる．具体的には，return 文に対応する行の下に二重線が引かれる．

4) 複数の演算を含む文の表現

1 行のコードに複数の操作を含めることができる．例えば，次のコードは 2 つの操作を含んでいる．

```
AbstractDisplay d1 = new CharDisplay('H');
```

一つはクラス "CharDisplay "のインスタンス生成，もう一つは生成されたインスタンス変数 d1 への代入である．

それぞれの操作を理解することは，プログラムを理解するために必要である．そこで，トレース表では，この 2 つの操作を実行順に 2 行で表現する．ここで，ステップ番号はすべて同じである．トレース表のある行が表す操作は，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．

5) 変数値の表現

変数のデータ型が整数などの基本データ型の場合，格納されている値が表示される．インスタンスが格納されている場合，セルにはインスタンス識別子が表示される．

プログラム内で変数に領域が割り当てられていない場合や，不定な状態の場合は空欄と

なる．この 2 つの場合は，プログラムの概念上は厳密には異なるが，運用実験の対象となるクラスでは，厳密に区別することを求めていない．オブジェクト指向プログラミング教育の初級レベルでは，このような微妙な違いを提示すると，学習者の混乱を招き，理解の妨げになる可能性があるため，2 つのケースは同じ表記を用いる．

6) 反復ステップの略

繰り返し回数が多くなると，トレース表が非常に大きくなり，把握が困難となる可能性がある．そこで，中間ステップを省略して反復処理を記述することにした．省略した行を明確にするために，（省略）と記述した行を挿入した．

B. トレース表内の穴抜き

トレース表では，各セルの値を穴抜きとして定義できる．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，メソッド名などである．また，メソッド引数，インスタンス変数，ローカル変数，外部オブジェクトについては，その小項目，データ型やクラス名，変数名，オブジェクト名などを穴抜きとして定義できる．

プログラムの場合と同様に，トレース表においても，解答を要しない無視できる穴抜きを定義できる．トレース表では，新しい値が代入されるまで，変数の状態は変化しない．そのため，プログラムから値を推測させたい場合でも，前後の行の値から推測してしまうことがある．これを防ぐためには，目標値の前後の値も穴抜きとして定義する必要がある．しかし，穴抜きが増えることで入力回数が増えるため，問題の難易度と解答の面倒さを混同する学生がおり，問題の難易度を検証する上では適切ではなかった．解答を要としない無視できる穴抜きを導入することで，この問題を解決できる．

6. 穴埋め問題の作成

ここでは，穴埋め問題の開発方針を提案する．作成した問題は，佐賀大学情報システム工学科 3 年次に開講される「プログラミング演習 III」にて提供する予定である．学生は 1 年次には Python を，2 年次には C++を用いた構造化プログラミングを学び，プログラミングの基本的な手法や，基本的なソートや整列のアルゴリズムを学習済みである．「プログラミング III」では，Java を用いたオブジェクト指向プログラミングの学習を開始する．

A. 対象クラス "プログラミング演習 III"

対象クラスでは，ソフトウェア開発の実務現場で使用される各種ツールの用途や使い方を学び，ソフトウェア開発の効率化と高品質化を目指すことを目的する．具体的には，統合開発環境（IDE）である Eclipse を用いて Java 言語でソフトウェアを開発し，JUnit や Jenkins などのツールの使用方法を学ぶ．

最初の 1 週間は，IDE についての講義と Java プログラミングの基本的な演習を行う．その後，Git と GitHub への登録，ソフトウェアテスト技術とユニットテスト設計，JUnit，Jenkins，UML 図，デザインパターンについて学習する．最後に開発演習を 4 週間行う．この授業計画と並行して，本章で提案する穴埋め問題を提供し，Java プログラミングの習得を支援する．

B. 穴埋め問題のトピック

演習の目的は，講義内容に対応した演習を行うことにより，学生の理解を深めることである．問題は「 Java で学ぶデザインパターン入門 」から 12 題を選び，そのサンプルプログラムを用いて作成する[11]．トピックは講義で扱う内容を考慮して選択し，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，上級と設定する（表 2）．

表 2：出題する穴埋め問題のトピック

ご注文	レベル	デザインパターン	教科書に掲載されている箇所
1	初級	TemplateMethod	3
2		FactoryMethod	4
3		Iterator	1
4		Composite	11
5	中級	Decorator	12
6		Strategy	10
7		AbstractFactory	8
8		Observer	17
9	上級	Adapter	2
10		Builder	7
11		Command	22
12		Visitor	13

C. 穴埋め問題の開発方針

これまでの研究により，C 言語の穴埋め問題に関して以下の知見が得られている[3, 4]．

- トレース表の中だけで定義された穴抜きを含む問題が最も簡単で，次いでプログラムの中だけで定義された穴抜きを含む問題である．トレース表とプログラムの両方に穴抜きが定義されている問題は，学生にとって最も困難である．
- プログラム内の穴抜きの難易度は，個々のトークン，トークンの並び，文全体の順に高くなる．

- トレース表内の穴抜きの難易度は，ステップ番号，変数値，変数名の順で高くなる．
- コメントは問題の難易度を下げる．
- 問題で定義された穴抜きの数が増えると，学生の解答意欲が低下する．

2 年次までの講義でプログラミングの基礎は学んでいるので，オブジェクト指向プログラミングの基礎やデザインパターンを学べるような問題と位置付けた．

プログラム内の穴抜きは，主にクラス定義などの Java 特有の文法の理解度チェックと，メッセージ送信の実行フローの確認を行うために定義する．さらに，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トピック，呼び出し側オブジェクトに関する項目，メッセージの送受信に伴って変化する値について定義する．

これらを考慮し，以下の方針を立て，適切なレベルの設問を作成した．

1. 1 問あたり 10 個程度の穴抜きを定義する．ここで，無視できる穴抜きはカウントしない．
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定義するために，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シーケンスを使用する．
6. トレース表内の穴抜きを定義するために，初級レベルでは主に変数値を使用する．中級になると，インスタンス識別子やメソッドの穴抜きを多く使う．

方針 1 については，これまでの経験から，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，pgtracer を継続的に使用できなくなることが分かっているために設定した．前回の実験では，対象者が C 言語の初心者であること，プログラムの総行数が 20 行程度であることから，1 問あたり 5 つの穴抜きを定義した．しかし，今回の実験では，学生のプログラミング習熟度が高いと予想されるため，10 個の穴抜きを定義した．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，難易度の違いを検討するために設定する．さらに，方針 3，4 で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ために設定する．方針 5，6 は，初級，中級，上級の違いを明確にするために設定する．

D. 穴埋め問題のトピック

表3に,「TemplateMethod 」で説明したプログラムをもとに作成した問題の教育目的と,問題の一部の穴抜きの意図を示す.

表 3：問題の教育目的および穴抜きの意図（テンプレートメソッド）

教育目的	抽象クラスと抽象メソッドの使い方を認識できる。	
穴抜き番号	正解	穴抜きの意図
(1)	CharDisplay	コンストラクタの呼び出しを理解している
(2)	StringDisplay	コンストラクタの呼び出しを理解している
(3)	display	メソッド呼び出しができる
(4)	d2.display()	出力よりインスタンスとメソッドを導くことができる

穴抜き(1), (2)について考える．変数 d1 と d2 は抽象クラス「AbstractDisplay」の変数であるため，学生は，「AbstractDisplay」のサブクラスである「CharDisplay」または「StringDisplay」のいずれかのコンストラクタが呼ばれることを理解し，その引数から適切なコンストラクタを導出しなければならない．コメントから適切なコンストラクタを導出するのは比較的容易である．ステップ 2 の記述は，穴抜き(2)と同じ処理であるため，無視できる穴抜きとしている．

穴抜き(3)と(4)は，メソッド呼び出しの記述である．穴抜き(4)は出力値からインスタンスを導出する必要がある．さらに，トークンの列であるが，コメントとステップ 6 の文からも導出できる．

このように，トークン，トークンの並び，文全体といった穴抜きの長さと，コメントの表示・非表示を利用して，穴抜きの難易度を制御できる．この問題は，穴抜きをそのままにしてコメントを非表示にすると，より難しくなる．

7. 考察

プログラム内の穴抜きやトレース表を作成する際に，それぞれの穴抜きの意図を明確にした．その結果，解答を導くためには，Java の文法，デザインパターン，処理を組み合わせて考えることが必要であるとわかった．この知見をさらに分類し，学生の解答記録と合わせて分析することで，各穴抜きの難易度を推定することができると考えている．

問題作成は，複数のプログラムファイルやトレース表のクロスチェックが必要であり，複雑な作業となる．そこで，メソッド呼び出しの文からそのメソッドを定義しているプログラムへ，あるいはインスタンス識別子からそのインスタンスのトレース表へのナビゲーション機能が期待される．

穴抜きの前後には，トレース表の値が同一である，プログラムの説明が似ているなどのヒントが表示される場合がある．解答を求めない無視できる穴抜きを導入することで，これらの部分を隠すことができる．また，解答を必要とする穴抜きの数を制御することができる．

本章では，連続したトークンを 1 つの穴抜きと定義したが，各トークンを 1 つの穴抜きと定義することも可能である．この場合，学生はトークンの数をヒントとして得ることができるので，問題の難易度を下げることができる．複数のトークンを含む穴抜きを解けない学生には，トークンの数をヒントとして提示できる．

8. 結論と今後の課題

本章では，Java 言語用の穴埋め問題を提案した．問題の難易度は，穴抜きがプログラム中にあるかトレース表中にあるか，また，穴抜きの長さによって設定される．また，解答を要しない無視できる穴抜きを導入することで，問題の難易度をより柔軟に制御できる．

我々の研究グループでは，穴埋め問題を提供するプログラミング教育支援ツール「pgtracer」を開発している．本研究で提案する問題を pgtracer に適用し，実際の授業で実験を行う予定である．実験から収集したデータには LA 法を適用し，学生が間違えやすい箇所や解答行動を分析する予定である．これらの分析から得られた知見は，Java プログラミング教育の支援への活用が期待される．

謝辞

本研究は，日本学術振興会科研費補助金（課題番号 20K03232，20K03265）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] I-Han Hsiao, P. Brusilovsky, S. Sosnovsky, "Web-based parameterized questions for object-oriented programming", E-Learn'2008:,World Conference on E-Learning in Corporate, Government, 2008.

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

[7] M.Hauswirth, A. Adamoli, "Teaching Java programming with the Informa clicker system", Science of Computer Programming, Vol.78, Issue 5, pp.499-520, 2013.

[8] N. Funabiki，Y. Matsushima，T. Nakanishi, etc, "A Java programming Learning Assistant System using test-driven development method," IAENG International Journal of Computer Science, vol.40, no.1, pp.38-46, 2013.

[9] K.K. Zaw, N. Funabiki, W.C.Kao, " A proposal of value trace problem for algorithm code reading in Java programming learning assistant system," Information Engineering Express, Vol.1, No.3, pp.9-18, 2015.

[10]H.H. S. Kyaw, N. Funabiki, W.-C.Kao, "A proposal of code amendment problem in Java programming learning assistant system," International Journal of Information and Education Technology, Vol.10, No.10, pp.751-756, 2020.

[11]結城，「Java プログラミング言語によるデザインパターン入門」，改訂版，ソフトバンククリエイティブ，2004 年．(ソフトバンククリエイティブ)

[12]S.H. Edwards, N. Kandru, M. B. M. Rajagopal, "Investigating static analysis errors in student Java programs", International Computing Education Research (ICER) conference, pp.65-73, 2017.

[13]D.McCall, M. Kolling, "Meaningful categorisation of novice programmer errors", In Frontiers In Education Conference, pages 2589-2596, 2014.

[14]A.Altadmri, N. C. C. Brown, "37 million compilations:Investigating novice programming mistakes in large-scale student data" SIGCSE '15 Proceedings of the 46th ACM Technical Symposium on Computer Science Education, pp.522-527, 2015.