Vol.2015-CE-128 No.11
2015/2/14

Javaプログラミング授業におけるエレメント空欄補充問題の実践

塔娜[1,a)]　舩曳 信生[1,b)]　石原 信也[1,c)]

概要：本グループでは，プログラミング言語 Java の学習支援のために，Web を用いた Java プログラミング学習支援システム JPLAS(Java Programming Learning Assistant System) を開発している．JPLAS では，Java の初学者を対象とした**エレメント空欄補充問題**を提供しており，グラフ理論を用いた解の一意性を充たす問題生成のためのアルゴリズムを提案している．今回，本アルゴリズムを用いてエレメント空欄補充問題を生成し，本学科の Java プログラミング授業で実践活用した結果を報告する．

1. はじめに

オブジェクト指向のプログラミング言語 Java は，開発環境，堅牢性，可搬性などに優れ，エンタープライズシステムから組み込みシステムに至るまで，産業界で広く利用されている．そのため，産業界からは，Java 言語技術者の育成が強く求められている．実際，大学，専門学校など多くの教育機関で，Java プログラミング教育が行われている．

本グループでは，Java プログラミング学習支援を目的として，Web を用いた *Java* **プログラミング学習支援システム** *JPLAS(Java Programming Learning Assistant System)* を提案・開発している [1]-[3]．JPLAS では，教員の負担を軽減しながら，学生の自宅などでのプログラミング学習を支援するために，Web サーバにおいて，演習課題に対する学生からの解答をオンラインで自動採点することを基本としている．これにより，学生は，Web ブラウザを用いて JPLAS サーバにアクセスし，プログラミング課題への解答を繰り返し行うことを狙いとしている．

JPLAS では，様々なレベルの学生に対応するために，3 種類の問題を提供している．その一つとして，模範となる良質の Java コードの中から学ぶべきエレメント（Java コードの基本単位）を空欄とし，そこに正しいエレメントの入力を要求する**エレメント空欄補充問題**を実装している [2][3]．本問題は，主に Java の文法などを学ぶ Java プログラミング初学者を対象としている．

エレメント空欄補充問題では，1)Java コードの中で学習すべきエレメントの空欄化（問題生成），2）各空欄へのエレメントの入力（解答），3）空欄化前のエレメントとの比較（採点）の 3 段階で学習が進められる．空欄エレメントは，**予約語**に加え，**識別子**，**文法記号**としている．**予約語**は，Java の文法で定められた機能を提供するための固有の文字列である．ここで，“System”，“out”，“String” は，本来予約語ではないが，学習すべき Java の重要なエレメントであるため，本研究では予約語に含めている．**識別子**は，データを格納するためのメモリのアドレスやクラス，メソッドを参照する語である．**文法記号**は，それ以外の Java の文法に密接に関係する記号である．本研究では，“ . ”（ ドット），“ , ”（カンマ），“ :” （コロン），“ ; ”（セミコロン），左右の波括弧 “{, }”，丸括弧 “ (,) ” としている．Java プログラミング学習では，これらの正しい使い方の習得が必須である．

Java コードの中で，それらのエレメントを無作為に空欄化した場合，文法的には，空欄化前のエレメントが一意の正しい解とならないと言った問題が生じる．サーバで採点する際に，学生の解答が文法上正しいが，空欄化前のエレメントと異なるために誤りと判断され，学生に混乱を与える恐れがある．また，無作為の空欄化では，得られた問題の難しさを制御することができず，学習の進捗状況や学生のレベルに応じて適切な難しさを有する問題の提示が困難となる．

そのため，以上の問題点の解決のために，グラフ理論を応用した**空欄語選択アルゴリズム**を提案している．本アルゴリズムでは，Java コードの中で空欄化候補となるエレ

[1] 岡山大学大学院自然科学研究科
Graduate School of Natural Science and Technology, Okayama University 3-1-1 Tsushimanaka, Okayama 700-8530, Japan
[a)] pad71zj8@s.okayama-u.ac.jp
[b)] funabiki@okayama-u.ac.jp
[c)] n.ishihara@okayama-u.ac.jp

メントを**点**とし，同時に空欄化した場合にも解の一意性を充たすエレメント間に**辺**を設けたグラフを作成し，その**クリーク**を探索することで，空欄エレメントの選択を行う．クリークは，グラフ理論の重要な概念であり，互いに辺で接続されている点のみで構成される部分グラフ（完全部分グラフ）である [4]．グラフの極大となるクリークを探索することで，与えられた Java コードに対して，空欄エレメント数を最大とする問題の生成が可能となる．教員は，その中から，今回の出題レベルに適した空欄エレメントを選択することで，エレメント空欄補充問題を作成する．

本アルゴリズムを用いてエレメント空欄補充問題を生成し，その実践として 2014 年 10 月より，本学科の Java プログラミング授業に適用している．毎週 49 名の学部 2 年生に対して，JPLAS により，エレメント空欄補充問題の演習問題を提供した．そして，4 週目に小テスト，10 週目にアンケート調査による効果測定を行った．その結果，JPLAS の適用が Java プログラミング学習の興味を向上することが確認された．

本論文の構成は以下の通りである．まず，2 章で，JPLAS のエレメント空欄補充問題を紹介する．次に，3 章で，解の一意性を充たす空欄エレメントの条件を示す．4 章で，空欄エレメント選択アルゴリズムを示す．5 章で，エレメント空欄補充問題の授業適用結果を述べる．最後に，6 章で本研究のまとめと今後の課題を示す．

2. JPLAS のエレメント空欄補充問題

本章では，Java プログラミング学習支援システム JPLAS に実装されているエレメント空欄補充問題の概要とその問題点を示す．

2.1 JPLAS のプラットフォーム

JPLAS は，Java プログラミングの自宅学習を容易とするため，Web アプリケーションとして開発されている．サーバ OS に Linux，Web サーバ兼アプリケーションサーバに Tomcat，データベースに MySQL を利用し，サーバプログラムは JSP，Servlet で記述されている [5][6]．学生は，Web ブラウザを用いてサーバにアクセスし，JPLAS のサービスを受ける．

2.2 エレメント空欄補充問題の流れ

エレメント空欄補充問題では，Java 初学者のプログラミング学習支援を目的としている．まず，1) 教員が問題に適した（模範となる）Java コード（**問題コード**）を選び，その中で学習すべきエレメントを空欄化することで問題を作成する．次に，2) 学生が各空欄にエレメント（スペル）を入力することで解答する．最後に，3) サーバで空欄化前のエレメントとの比較により解答の検証（採点）を行う．空欄エレメントには，Java の**予約語**，**識別子**，**文法記号**としている．教員は，問題コードの中からそれらを，空欄化されたエレメントが一意の正解となるように，適切に選択する必要がある．

2.3 空欄補充問題の手順

JPLAS では，上記 1）の手順は，①問題コードのデータベース登録，②問題コードの選択，③問題コードからの空欄エレメント選択による問題作成，④問題および正解のデータベース登録，⑤問題選択による演習課題の作成，で進められる．これらは教員のみ，可能としている．2）の手順は，①演習課題の選択，②空欄エレメントのスペル入力による解答と提出，で進められる．これらは教員，学生で可能としている．3）の手順は，①正解エレメントとの比較による正誤判定，②判定結果の表示，で進められる．

教員には，学生の個別指導や成績評価のために，全学生の全課題の最終解答結果（正誤判定結果）と繰り返し回数の参照を可能としている．**繰り返し回数**は，学生が各課題に対して解答を再提出した回数であり，学生の個別指導の参考とするためのものである．また，学生には，自身の課題の取り組み状況を認識させるため，全課題の全解答結果と解答日時の参照を可能としている．更に，匿名で全学生の正答課題数をグラフで示し，互いに比較可能とする，ランキンググラフ表示機能を有している．これにより学生間で正答数を競争し合うことで，本問題への意欲を高めることを期待している．

2.4 空欄エレメント選択の困難さ

以上の手順において，教員によるエレメント空欄エレメント選択の作業は，非常に困難である．これは，JPLAS では空欄化前のコード中のエレメントを一意の正解としているのに対し，文法的にはそれとは異なるエレメントでも正解となる場合が頻繁に生じるためである．本グループでは，この問題点に対して，解の一意性を充たすための空欄エレメント選択の条件を示した上で，グラフ理論を用いたアルゴリズムを提案している．

3. 一意性のための空欄エレメント選択条件

本章では，解の一意性を充たすための空欄エレメント選択条件を示す．本条件では，Java コード中の空欄候補のエレメントを，空欄化禁止，グループ選択，ペア選択，単独選択の 4 種類に分類した上で，それぞれでの空欄エレメント選択方法を与える．

3.1 空欄化禁止

ここでは，空欄化した場合に解の一意性を満たさない可能性の高いエレメントについての条件を明らかにし，それらのエレメントを空欄化しないこととする．以下に，本研究での条件を示す．

(1) 1 度しか出現しない識別子

Java の言語仕様では，クラス名，メソッド名，変数名，フィールド名などの識別子を，そのルールの下で自由に与えることができる．そのため，問題コードの中で 1 度しか出現しない，これらのエレメントが空欄化された場合，解の一意性が保障できないため，空欄化しないこととする．

(2) (1) に対応するデータ型宣言

例えば，“データ型　$num = 10$;”において，変数 num が問題コード中に 1 度しか出現しない場合，この”データ型”が空欄化された場合の解として，int, $short$, $long$ の 3 つが挙げられる．そのため，(1) で示した識別子の宣言するデータ型宣言を，空欄化しないこととする．

(3) 演算子

問題コード中の演算式における，算術演算子 +，−，∗，/，比較演算子 <，>，<=，>=，==，!=，論理演算子 &，|，^，! が空欄された場合，その演算式に対する適切な説明がない場合には，正しい演算子を解答することが出来ないため，空欄化しないこととする．

(4) アクセス修飾子

アクセス修飾子と呼ばれる，$public$, $protected$, $private$ は，それが指定する変数やクラスに対して，参照可能となるクラスの範囲（スコープ）の制御に用いられる．問題コードのクラスが 1 つのみの場合，いずれのアクセス修飾子も文法的に正しくなるため，空欄化しないこととする．

3.2 グループ選択

問題コードにおいて，互いに関連する複数のエレメントを 1 つのグループとし，同じグループの中の全てのエレメントを同時に空欄エレメントに選択しないこととする．前述の空欄化禁止における (1)，(2) は，1 エレメントのみでグループを形成する場合に該当する．

(1) 2 度以上出現する識別子

Java コードにおいて，変数，クラス，メソッドが，特定の名前（識別子）で参照される範囲は**スコープ**と呼ばれる．同一スコープの同一名の識別子全てを空欄化した場合，問題から一意にそれを定めることが出来ない．そのため，同一スコープの同一名の識別子を 1 つのグループとして，その中から少なくとも 1 つを空欄化しないこととする．

(2) 式の変数のデータ型宣言

例えば，式 $sum = data1 + data2$ において，全ての変数 sum，$data1$，$data2$ のデータ型が一致しなければならない．そのため，これらの変数のデータ型宣言を 1 つのグループとする．ここで，$sum = data1 + (int)data3$ のように，変数がキャスト宣言されている場合，キャスト中のデータ型宣言をそのグループに含める．また，変数の代わりに，以下のコード 1 のように，戻り値のあるメソッドが代入式に含まれる場合も，そのメソッドのデータ型宣言をグループに含める．

コード 1: 戻り値のあるメソッド

```
1  public static void main(String args[]){
2    int var1;
3    float receiveValue = methodName(var1, ...);
4    /*処理を行う*/
5  }
6  public static float methodName(int data1,
       ...){
7    float returnValue;
8    /*処理を行う*/
9    return returnValue;
10 }
```

(3) 戻り値とメソッドのデータ型宣言

コード 1 のように，戻り値のあるメソッドでは，メソッドのデータ型と戻り値のデータ型（ここでは，float）が一致しなければならないため，これらのデータ型宣言を 1 つのグループとする．

(4) 引数のデータ型宣言

コード 1 のように，引数を用いてメソッドを呼び出す場合，このメソッドに渡す引数のデータ型とメソッドの仮引数のデータ型（ここでは，int）が一致しなければならないため，これらのデータ型宣言を 1 つのグループとする．

(5) 同一データ型宣言の複数グループの統合

データ型宣言に関する条件 (2)〜(4) において，同一のデータ型宣言が 2 つ以上のグループに属する場合，それらの全グループでのデータ型宣言が一致する必要がある．そのため，それらを 1 つのグループに統合する．

3.3 ペア選択

問題コードにおいて，ペアで出現するエレメントを同時に空欄エレメントに選択しないこととする．ペアで出現するエレメントとして，例えば，$if - else$ といった予約語が挙げられるが，それらを同時に空欄化した場合，解の一意性を充たさない可能性が高くなる．以下にペアとなるエレメントの組合せを示す．

(1) 連続出現のエレメント

問題コード中の同一ステートメントで，連続して出現する 2 つのエレメントを，ペアとする．また，連続していないが，“.”（ドット）で繋がれた 2 つのエレメントもペアとする．

(2) 式の変数

式 $ans = data1 * data2$ や式 $sum = data1 + data2$ において，変数 $data1$，$data2$ は順不同であり，$data1$，$data2$ の両方を同時に空欄化した場合，解が 2 通りできてしまうため，これらの変数をペアとする．更に，$*$，$+$ で繋がれた変数が 3 つ以上存在する場合，それらの任意の組合せをペアとする．

(3) 対となる予約語

$if - else$ など，対となる予約語をペアとする．これには，以下の 7 種類が含まれる．

- if - else
- switch - case - default
- do - while
- try - catch - finally
- class - extends
- interface - extends
- interface - implements

但し，対となる予約語であっても，コード上で対応していない予約語同士は，この条件から除外する．

(4) 対となる文法記号

Java コード中の括弧や波括弧は，記入漏れに気づきにくいことが多く，Java 初学者には，まず，それらを確認する習慣を身に付けさせることが必要である．しかし，対となる括弧・波括弧の両方を同時に空欄化した場合，それらを不要とする解が生じる可能性があることから，それらをペアとする．

3.4 単独選択

上記以外のエレメントが空欄化された場合には，解の一意性を充たすため，単独で空欄化可能とする．

4. 空欄エレメント選択アルゴリズム

本章では，前章で述べた空欄エレメント選択条件に従い，グラフ理論を用いた空欄エレメント選択アルゴリズムを示す．

4.1 アルゴリズムの方針

本アルゴリズムでは，まず，問題コード中の空欄化候補のエレメントを**点**として，同時に空欄化した場合に解の一意性を充たさないエレメントの間に**辺**を設けた**制約グラフ**を生成する．次に，その補グラフを求めることで，**適合グラフ**を生成する．そして，その**クリーク**を探索することで，解の一意性を充たす，極大数の空欄エレメントを選択する．

4.2 制約グラフの生成

制約グラフは，空欄化候補のエレメントを点，同時に空欄化すべきでないエレメント間を辺で表現した単純グラフである．

4.2.1 点の生成

問題コードの全空欄化候補のエレメント（予約語，識別子，文法記号）を制約グラフの点として表現する．これらのエレメントは，jFlex[14]，jay[15] を用いた字句解析により収集する．

jFlex は，Java で書かれた Java 用の字句解析ルーチン生成系である．字句解析では，コードの文字列を文法的に意味のある最小単位（字句）の列に変換する．コードを読み込み，そのエレメントをシンボルとして分解し，分解した字句を予約語，識別子，文法記号，即値のいずれかに分類する．例えば，int sum = data1 + data2; は，int，sum，=，data1，+，data2，; に分解される．ここで，JFlex は，予約語，文法記号，即値はそれぞれに分類するが，識別子が変数，クラス，メソッドのいずれかであることは分類できない．そのために，jay を併用する．jay は LALR 法を用いるパーザ（構文解析プログラム）を生成する構文解析生成系であり，JFlex によって字句解析されたデータを，構文解析することによって識別子の分類を行う．

4.2.2 点の付加情報

各空欄化候補の点には，以上の解析によって得られる，以下の付加情報を保存しておく．

表 1: 点の付加情報

記号	意味
sym	エレメントの種類
line	問題コードの総行数
column	エレメントの出現行での順番
value	変数の場合は変数名，それ以外は null
count	問題コードでのエレメントの出現回数
order	問題コードでのエレメントの出現順
group	エレメントを括る波括弧の順番
depth	エレメントを括る波括弧の深さ

4.2.3 空欄化禁止エレメントの削除

前章で示した空欄化禁止のエレメントに対応する点を，制約グラフから削除する．これは，これらのエレメントは，以降の処理において，空欄エレメントとして選択しないためである．

4.2.4 辺の生成

次に，前章のグループ選択，ペア選択に含まれる空欄化候補のエレメントに対応する点に対して，辺を設ける．まず，**グループ選択**では，1 つのグループに含まれる点の中から，1 つの点をランダムに選択した上で，その点と同じグループの他の点との間に辺を設ける．これにより，各グループにおいて，少なくとも 1 つの点が空欄エレメントに選択されない．次に，**ペア選択**では，1 つのペアを形成する 2 点間に辺を設ける．これにより，各ペアにおいて，少なくとも 1 つの点が空欄エレメントに選択されない．

4.2.5 制約グラフの例

コード 2 を用いて，制約グラフの生成について説明する．図 1 に生成された制約グラフの一部を示す．

まず，点の生成について説明する．例えば，変数 var3 は，本コードで 2 回出現している（9，10 行目）ことから，それに対応する点を 2 つ生成する．また，System.out.println は，System，.（ドット），out，.（ドット），println の 5 つに分けて点を生成する．

次に，辺の生成について，前章の空欄エレメント選択条件毎に説明する．

1) 空欄化禁止の例

条件 (1) により，1 行目の Sample1 を空欄化禁止とする．条件 (3) により，9 行目の演算子*を空欄禁止とする．条件 (4) により，3 行目の public を空欄化禁止とする．そして，これらに対応する点を制約グラフから削除する．

2) グループ選択の例

条件 (2) により，9 行目の 2 つの float をグループとする．10 行目で，条件 (3) により，8 行目のメソッド sampleMethod の float と 9 行目の変数 var3 の float をグループとする．5 行目で，条件 (4) により，4 行目の int と 8 行目の変数 p1 の int をグループとする．条件 (5) により，これらの float，int に関するグループをそれぞれ統合する．そして，各グループの点の中からランダムに 1 つを選択し，同じグループの他の点との間に辺を設ける．

3) ペア選択の例

条件 (1) により，6 行目の System と out をペアとする．条件 (4) により，2 行目の { と 12 行目の } をペアとする．そして，各ペアの 2 点間に辺を設ける．

4) 単独選択の例

1 行目の class，12 行目の return は単独で空欄化可能なエレメントである．

コード 2: 制約グラフの例

```
1  class Sample1
2  {
3      public static void main(String args[]){
4          int var1 = 10;
5          float var2 = sampleMethod(var1);
6          System.out.println("indata="+var1+ "outdata="
               +var2);
7      }
8      static float sampleMethod(int p1){
9          float var3 = (float)(p1*1.08);
10         return var3;
11     }
12 }
```

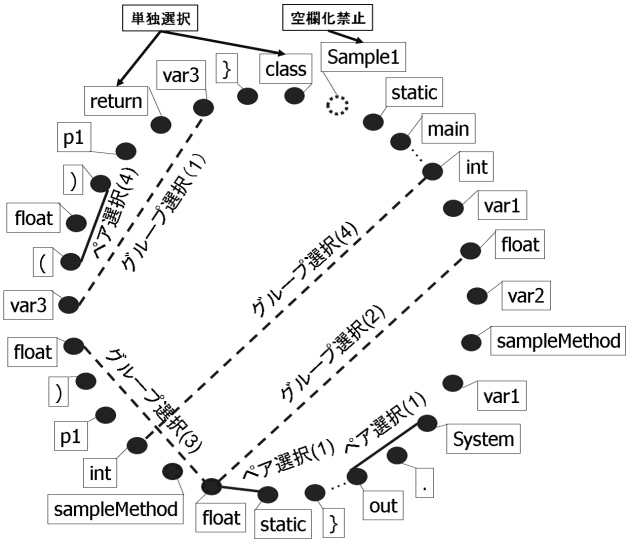

図 1: 制約グラフの例

4.3 適合グラフの生成

次に，制約グラフの補グラフを求めることで，**適合グラフ**を生成する．適合グラフでは，辺の存在する 2 点を同時に空欄化した場合に，解の一意性を充足する．

4.4 適合グラフのクリークの抽出

解の一意性を充足する，最大数の空欄エレメントを選択するために，貪欲法に基づくアルゴリズムを用いて，適合グラフの**クリーク**を抽出する．ここで，文法記号は，予約語や識別子に比べ，問題コード中に非常に多く出現することから，その空欄化数が全体の 1/3 以下となるように設定する．これにより，文法記号が多く空欄化されることを防いでいる．

本アルゴリズムでは，空欄エレメント数の最大化のために，まず，次数最大の点を選択する．複数存在する場合，その中からランダムに 1 つ選択する．そして，選択された点とその非隣接点を適合グラフから削除する．その後，文法記号の空欄化数を数え，それが空欄化数の 1/3 を超える場合，残りの文法記号をすべて削除する．これを，適合グラフが空となるまで繰り返す．図 2 にアルゴリズムの流れを示す．

5. 授業適用による評価

本章では，JPLAS の授業への適用を通じて，エレメント空欄補充問題の実用性に関する評価を行う．

5.1 評価方法

空欄エレメント選択アルゴリズムを用いて，エレメント空欄補充問題を 121 問（総空欄数は 1552）生成し，本学科 2 年生向け科目「応用プログラミング言語 II」の受講生 49 名に自習課題として与えた．これらの問題コードは，表 2

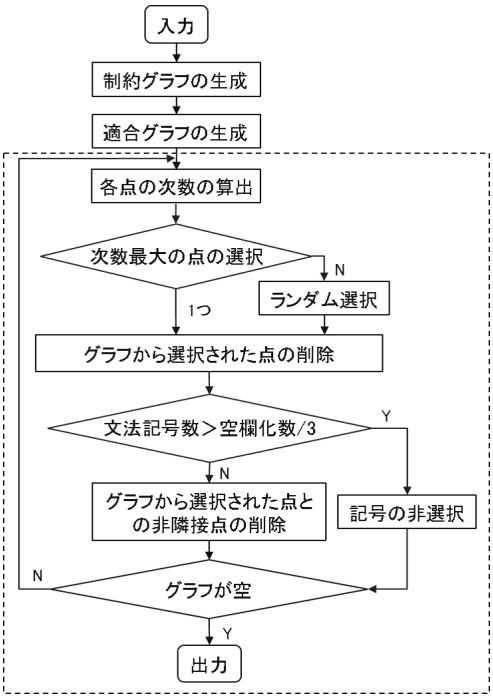

図 2: アルゴリズムの流れ

に示す授業の進捗に合わせて，テキスト [7]-[9]，Web サイト [10]-[12] より収集した．本科目は，Java 言語の文法に関する座学およびプログラミング演習の週 2 コマの授業で進められる，初学者を対象とした必修科目である．

本自習課題への取り組み結果を評価するため，5 週目に，小テストを実施した．また，10 週目に，アンケートを行った．その有効回答数は 46 人であった．

5.2 自習課題の結果

表 2 に 2015 年 1 月 15 日時点での自習課題の解答結果を示す．ここで，繰り返し回数は，学生が繰り返し解答を提出した回数である．

表 2: 自習課題の解答結果

週	出題数	問題の特徴	平均空欄数	解答者数	平均繰り返し回数	平均正解率%
1	24	文法記号	8.41	49	2.59	99.88
2	21	予約語	7.00	49	1.96	99.36
3	28	クラス	9.75	49	6.35	98.84
4	10	データ型	15.8	46	**13.57**	97.52
5	3	小テスト	20	46	12.03	90.71
6	5	ファイル入出力	12.75	46	10.59	99.26
7	12	デザインパターン	15.50	30	10.95	**98.67**
8	5	配列	11.60	30	6.7	99.58
9	5	データ構造	29.20	20	**21.26**	99.16
10	8	GUI	20.12	16	10.29	99.24

最初の 3 週間は，テキスト [7] 中の演習問題のプログラムを利用した．Java 言語の基本となる文法記号や予約語 (if, else，for など)，クラス関連の予約語 (class，extends など) を空欄化した平易な問題であり，解答に問題がなかった．

4 週目は，Web サイトからダウンロードした 10 個の問題コードで，データ型宣言のエレメントを中心にした空欄補充問題とした．その結果，平均繰り返し回数は前の週の 2 倍以上となり，平均正解率も低くなった．この理由として，問題コードがより難しくなったためと，空欄数（平均値 15.8）が増えたためと考えられる．

Java 言語の基本的な内容の勉強を終えた 6 週目以降，より応用的な問題を出題した．[8]-[10] のコードを用いて作成した，7 週目のデザインパターンの問題では，平均正解率は低くなった．この理由として，デザインパターンのコードは複数のクラスやメソッドで構成され，より複雑であったためと考えられる．

9 週目のデータ構造の問題では，平均繰り返し回数が 21.26 回と今回の中で最大となった．これは，コード中のアルゴリズムの処理の理解が必要なためと考えられる．

コード 3: エレメント空欄補充問題

```
1  //お金の種類、枚数を調べる (10000,5000,...1yen)
2  _1_ samples.number;
3  public _2_ KinshuSample {
4      public _3_ final int[] YEN_TYPES =
           {10000,5000,1000,500,100,50,10,5,1};
5      public _4_ int[] getYenCount(int yen,int[]
           yentype) {
6      _5_ [] count = _6_ int[yentype.length];
7       for (int i = 0; _7_ < yentype.length; _8_ ++) {
8           _9_ ( _10_ >= yentype[ _11_ ]) {
9           yen -= yentype[i];
10          count[ _12_ ]++;
11        }
12       }
13        _13_ count;
14     }
15     public _14_ void main(String[] args) {
16        int _15_ = 87654;
17        int[] count = getYenCount(yen, YEN_TYPES);
18         _16_ (int i = 0; i < YEN_TYPES.length; i
               ++) {
19           System. _17_ .println(YEN_TYPES[i] + "yen"
               + count[i]);
20        }
21      }
22  }//解答:1.package,2.class,3.static,4.static,5.int,
23  //6.new,7.i,8.i,9.while,10.yen,11.i,12.i,13.return,
24  //14.static,15.yen,16.for,17.out
```

5.3 小テストの結果

5 週目に小テストを行った．問題数を 3，解答時間を 15 分とした．表 3 に結果を示す．演習課題に比べ，正解率が低くなったのは，事前学習不足に加え，時間制限を設けたことが考えられる．ここで，特に正解率の低い問題は，C と異なるために馴染みの少ない，extends，throws の空欄，および，コード 3 のアルゴリズムに関する問題であった．

表 3: 小テストの結果

問題	コード内容	空欄数	繰り返し回数	正解率
1	ファイルの読み込み	23	11.86	81.82%
2	抽象クラス	20	9.88	84.95%
3	アルゴリズム	17	10.9	86.27%

5.4 アンケートの結果

表 4 にアンケートの設問を示す．各設問に対して，1〜5 の 5 段階で解答して貰った．その際，1 は最悪，5 は最良とした．設問中の JPLAS はエレメント空欄補充問題を意味する．アンケート結果を図 3〜7 に示す．これらの図での横軸は，自習課題における各学生の正解数を示す．

表 4: アンケートの設問

1. Java プログラミング学習に費やす時間
(1)1 週間の Java プログラミング学習に費やす時間はどのぐらいですか (2) この中で，JPLAS を利用する時間は週にどのぐらいですか
2. Java プログラミング学習における JPLAS の貢献度
(3)JPLAS を利用することより，Java プログラミングの理解度はアップしましたか (4)JPLAS の Java プログラミング学習に対する意欲向上への貢献度はどうですか
3. 現状の JPLAS の評価
(5) 空欄補充問題に出された問題の難しさはどうですか (6) これから色々空欄補充問題を出したいですが，期待していますか (7) 空欄補充問題に週に出される理想な問題数はいくらですか
4. 現状の Java プログラミング能力の自己評価
(8) 自己の Java プログラミング能力は周囲と比べて，どう思いますか (9)Java プログラミングに自信がつきましたか (10)Java プログラミングに興味を持ちましたか
5. その他，自由意見
(11)Java のプログラムで，一番読みにくいパターンはどれですか (12)JPLAS の改善・機能追加・問題設定への要望

5.4.1 Java プログラミング学習時間

図 3 に示すように，正解数の多い学生では，1 週間に 4 時間以上勉強している学生が多い．それに対し，正解数の少ない学生は，1 時間以下の利用者が多く，JPLAS も授業中の利用に限られていると言える．

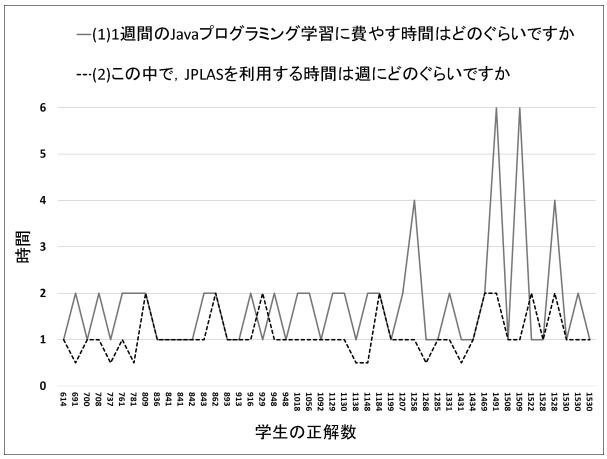

図 3: Java プログラミング学習に費やす時間

5.4.2 エレメント空欄補充問題の貢献度

設問（3）の回答は平均値 3.37，設問（4）は平均値 3.33 であり，JPLAS のエレメント空欄補充問題の貢献度に関して，望ましい結果が得られた．

ここで，図 4 に示すように，設問（4）において，正解数が 1,100 以上の学習者の中の 4 名は最低レベルの 1 を選んでいる．その理由としては，初学者を対象にしたエレメント空欄補充問題は，これらの学習者に簡単過ぎたためと考えられる．反面，1,100 以下の学生の中の 3 名は，最高レベルの 5 を選んでいる．これより，学習者のレベルに合わせた難易度の問題を作成することが必要と言える．

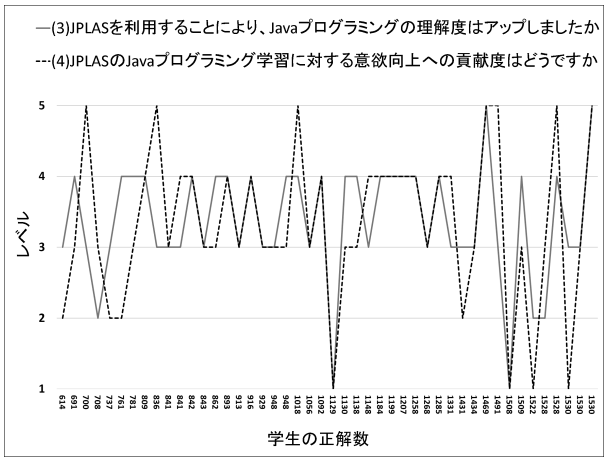

図 4: Java プログラミング学習における JPLAS の貢献度

5.4.3 エレメント空欄補充問題の評価

設問（5）において，およそ半数の学生が問題の難しさを 4 以上と答えており，今回の問題は難しかったと感じていることが分かった．また，設問（6）の回答は平均 3.02 であり，3 以上を選んだ学習者は 36 人であった．ここで回答の詳細を見ると，図 5 に示すように，「すごく難しい (レベル 5)」と答えた学生の中の正解数 836 以下の 1 人と，正解数（1,129，1,509，1,522，1,530）の多い学生 4 人が，本問題への期待値が一番低かった．

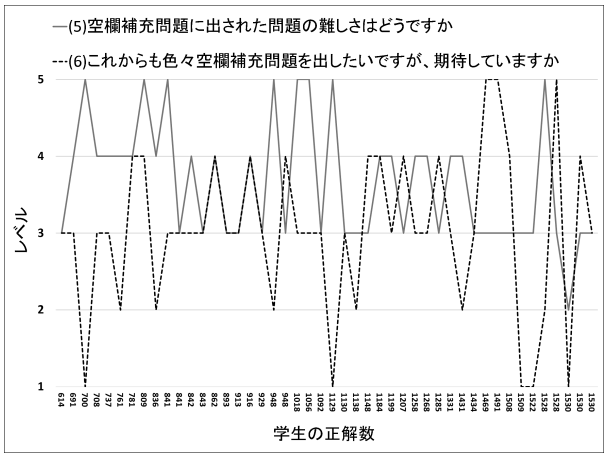

図 5: 現状の JPLAS の評価

5.4.4 Java プログラミング能力の自己評価

図 6 に示すように，学生の Java プログラミング能力に対する自己評価は，正解数に関係なく，全般的に低い評価（平均 2.39）となった．また，設問（10）の回答は平均 3.80 となり，JPLAS のエレメント空欄補充問題は，学生の Java プログラミングへの興味を向上させていることが確認された．

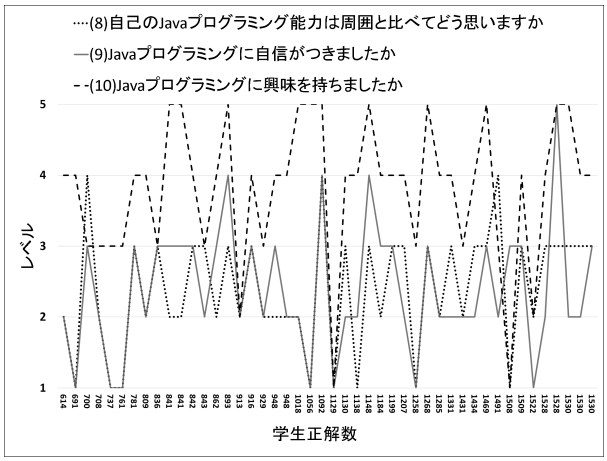

図 6: Java プログラミング能力の自己評価

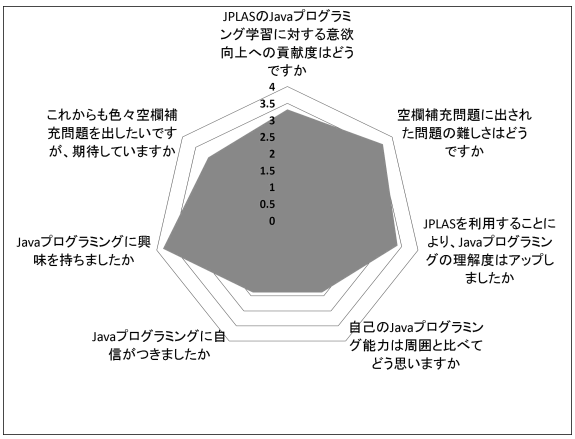

図 7: 各項目の比較

5.4.5 その他，自由意見

Java コードとして読みにくいパターンとして，複雑なプログラム，間違っているプログラム，長いプログラム，汚い (ネストができてない) プログラムなどが挙げられた．この意見を参考として，問題の難易度を判定することが考えられる．また，JPLAS の改善・機能追加・問題設定への要望として，「問題コードの実行結果をそれぞれの問題に示して欲しい」，「プログラムの解説がもっと欲しい」，「ユーザインタフェースでは，コード中の空欄場所見つけにくい」が挙げられた．これらへの対応が今後の課題である．

6. まとめと今後の課題

本研究では，*Java* **プログラミング学習支援システム** *JPLAS* の**エレメント空欄補充問題**の授業での実践活用を通じて，その実用性に関する評価を行った．今後の課題として，問題コードの実行結果の提示，ユーザインタフェースの改善，空欄補充問題の難易度の判定などが挙げられる．

参考文献

[1] Funabiki N., Matsushima Y., Nakanishi T., Watanabe K. and Amano N.: A Java programming learning assistant system using test-driven development method. IAENG Int. J. Computer Science, vol. 40, no.1, pp. 38-46, 2013.
[2] Funabiki N., Korenaga Y., Matsushima Y., Nakanishi T. and Watanabe K.: An online fill-in-the-blank problem function for learning reserved words in Java programming education. Int. Symp. Front. Inform. Sys. Net. Appl. (FINA 2012), pp. 375-380, 2012.
[3] Funabiki N., Korenaga Y., Nakanishi T. and Watanabe K.: An extension of fill-in-the-blank problem function in Java programming learning assistant system. Human. Tech. Conf. (R10-HTC2013), pp. 95-100, 2013.
[4] Garey M. R. and Johnson D. S.: Computers and intractability, A guide to the theory of NP-completeness. Freeman, New York, 1979.
[5] 矢吹太朗: 初級プログラマのための Web アプリケーション構築入門, 森北出版, 2007.
[6] 竹形誠司: Java+MySQL+Tomcat ではじめる Web アプリケーション構築入門, ラトルズ, 2006.
[7] 高橋麻奈: やさしい Java, 第 5 版, ソフトバンククリエイティブ発行, 2013.
[8] 近藤嘉雪: 定本 Java プログラマのためのアルゴリズムとデータ構造, ソフトバンククリエイティブ発行, 2011.
[9] 結城浩: 増補改訂版 Java 言語で学ぶデザインパターン入門, ソフトバンククリエイティブ発行, 2006.
[10] IT 専科, http://www.itsenka.com/.
[11] Java Tutorial, http://www.tutorialspoint.com/java/index.htm.
[12] Java プログラムサンプル集, http://www7a.biglobe.ne.jp/~java-master/samples/.
[13] JUnit, http://www.junit.org/.
[14] JLex, http://www.cs.princeton.edu/~appel/modern/java/JLex/.
[15] jay, http://www.cs.rit.edu/~ats/\\projects/lp/doc/jay/package-summary.html.