博士論文

Javaプログラミング学習支援システムにおける空欄エレメント選択アルゴリズムの提案

平成29年9月

塔 娜

岡山大学大学院
自然科学研究科

内容概要

開発環境，堅牢性，可搬性に優れたプログラミング言語 Java は，産業界で広く利用されており，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 プログラミング教育の支援を目的として，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)* を提案し，その開発と運用を行っている．

JPLAS は，学生の自宅などでの自習を可能とするため，Web アプリケーションとして開発されている．JPLAS では，Java プログラミングを学ぶ多様な学生の異なる学習レベルに対応するために，大きく，**コード作成問題**と**レメント空欄補充問題**の 2 種類の問題機能を有している．**コード作成問題**機能では，JPLAS のサーバにおいて，Java コードの検証ツール JUnit を用いて，オンラインで学生から提出されたコード（解答コード）の検証を行う．学生に，検証結果を直ちにフィードバックすることで，解答コードの誤りの発見とその修正が可能となる．その際，バグの検出を容易にするために，コード上のエラー箇所のハイライト表示機能を有している．学生は，正しい検証結果が得られるまで，解答コードの修正と提出を繰り返すことで，Java プログラミングの学習を進める．ここで，解答コードの自動検証のために，学生は，教員から提示されるテストコードの実行を可能とする解答コードの作成が求められる．すなわち，クラス名，メソッド名，返り値の型，引数の数・型などをテストコードに記述されたものに一致させる必要がある．そのため，Java の学習を始めたばかりで，テストコードの理解が困難である学生（Java 初学者）には，ハードルの高い学習問題となっている．その対策として，JPLAS では，エレメント空欄補充問題機能を提供している．

エレメント空欄補問題では，1）Java コード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）で，学習が進められる．空欄化されるエレメントは，予約語に加え，変数，文法記号，条件文演算子としている．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．識別子は，変数名，メソッド名，クラス名，フィールド名を意味する．文法記号は，予約語以外の Java の文法に密接に関係する記号で，“ ; ”（セミコロン），“ { ，} ”（波括弧）などである．条件文演算子は，条件文で使用される“ <”（不等号），“ != ”（不一致）などである．

Java プログラミング学習初学者には，それらの正しい使い方の習得が必須である．

エレメント空欄補充問題の採点では，空欄化前のエレメントを一意の正解として，それとの文字マッチングにより正誤判定を行っている．そのため，Java コードの中でそれらを無作為に空欄化した場合，正解が一意に定まらない可能性がある．解答が文法的に正しくとも，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．

本研究では，この問題を解決するために，グラフ理論に基づく**空欄エレメント選択アルゴリズム**の提案を行う．まず，解の一意性を満たさないエレメントの空欄化を防ぐために，3 種類の選択条件を定義する．Java コード中の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，同じグループ内の全てのエレメントを同時に空欄化しないようにするグループ選択条件，Java コード中のペアで出現するエレメントを同時に空欄化しないようにするペア選択条件，そして，解の一意性を充たさない可能性のあるエレメントを空欄化しないようにする空欄禁止条件の 3 つである．次に，オープンソースの jFlex，jay を用いた字句解析により，Java コードから空欄化対象となるエレメントを抽出する．それらのエレメントを点とし，3 種類の選択条件により同時に空欄化できない 2 つのエレメント間に辺を設けた**制約グラフ**を作成する．そして，その補グラフである適合グラフの**クリーク**を探索することで，解の一意性を充たす問題を生成する．

本研究の評価として，提案アルゴリズムの有効性の検証と，本アルゴリズムにより生成したエレメント空欄補充問題の学習効果の評価を行った．まず前者では，100 個の Java コードを用いた解の正当性の検証，空欄数と Java コードの行数の関連性の調査，および，学生への適用による空欄数変化時の問題の難易度変化の分析を行った．また後者では，提案アルゴリズムを用いてエレメント空欄補充問題を作成し，2014 年および 2015 年の 2 年間，岡山大学工学部電気通信系学科の Java プログラミング授業の自習課題として，受講生に解答してもらった．そして本授業の終了後に，受講生のエレメント空欄補充問題の解答結果と授業の最終課題（Java アプリケーションプログラムの作成）の関連性を解析することで，その有効性を確認した．更には，Java プログラミング授業での利用を容易とするため，教科書や Web サイトから問題作成に適切となる Java コードを収集し，Java プログラミングの学習順序に沿ったカテゴリ毎に整理した上，提案アルゴリズムを用いてエレメント空欄補充問題のワークブックを作成し，その有効性の評価を行った．

空欄エレメント選択アルゴリズム，および，JPLAS のエレメント空欄補充問題の今後の課題として，生成するエレメント空欄補充問題の難易度の調整機能，学習者のレベルや進捗に応じた問題の提示機能，解答の困難な学生へのヒント提示機能，ワークブックの Java プログラミング授業での活用と評価などが挙げられる．

目 次

関連発表論文

1. 学術論文誌

1-1 **Tana**, Nobuo Funabiki, Khin Khin Zaw, Nobuya Ishihara, Shinpei Matsumoto, and Wen-Chung Kao, “A fill-in-blank problem workbook for Java programming learning assistant system ,” International Journal of Web Information Systems (IJWIS), Vol. 13 Issue: 2, pp. 140-154, July 2017.

1-2 Nobuo Funabiki, **Tana**, Khin Khin Zaw, Nobuya Ishihara, and Wen-Chung Kao, “A graph-based blank element selection algorithm for fill-in-blank problems in Java programming learning assistant system, ” IAENG International Journal of Computer Science, vol. 44, no. 2, pp. 247-260, May 2017.

2. 国際会議　会議録

2-1 **Tana**, Nobuo Funabiki, Toru Nakanishi, and Noriki Amano, “An improvement of graph-based fill-in-blank problem generation algorithm in Java programming learning assistant system, ”Proceedings of International Workshop on ICT, pp. 1-4, Beppu, Dec. 2013.

2-2 **Tana**, Nobuo Funabik, “A proposal of graph-based blank element selection algorithm for Java programming learning with fill-in-blank problems,” Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS 2015), pp. 448-453, Hong Kong, March 2015.

2-3 **Tana**, Nobuo Funabiki, and Nobuya Ishihara, “Practices of fill-in-blank problems in Java programming course,” Proceedings of IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW2015), pp. 120-121, Taipei, June 2015.

2-4 **Tana**, Nobuo Funabiki, and Nobuya Ishihara, “Drawbacks and countermeasures of element fill-in-blank problem in Java programming learning

assistant system," Proceedings of 2015 IEICE Society Conference, BS-6-1, Sendai, Sep. 2015.

2-5 **Tana**, Nobuo Funabiki, Nobuya Ishihara, and Wen-Chung Kao, " Correlation analysis of fill-in-blank problem solutions to final programming results in Java programming course," Proceedings of IEEE Global Conference on Consumer Electronics (GCCE2015), pp. 348-349, Osaka, Oct. 2015.

3. 学術研究集会

3-1 **塔娜**, 舩曳信生, 中西透, 渡邊寛, " グラフ理論を用いた Java エレメント補充問題生成アルゴリズムの提案, " 第 64 回電気・情報関連学会中国支部連合大会, pp. 303-304, Oct. 2013. [奨励賞]

3-2 **塔娜**, 舩曳信生, 石原信也, " Java プログラミング学習支援システムのための空欄補充問題生成アルゴリズムの拡張, " 信学技報, ET2014-19, pp. 63-68, June 2014.

3-3 **塔娜**, 舩曳信生, 石原信也, " Java プログラミング学習支援システムのための空欄語選択アルゴリズムの提案, " 信学技報, vol. 114, no. 271, SS2014-25, pp. 1-6, Oct 2014.

3-4 **塔娜**, 舩曳信生, 石原信也, " 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, " 2015-CE-128, pp. 1-8, Feb 2015.

図 目 次

表 目 次

第1章　緒論

1.1　本研究の背景

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，産業界で幅広く利用されている．そのため，産業界から Java 言語技術者の育成が強く求められており，実際，大学や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 プログラミング教育の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)* を提案している [1]-[5]．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容易とするために，Web サーバにおいて，学生からの解答のオンライン自動採点を行うことを基本としている．学生は，Web ブラウザを用いて JPLAS サーバにアクセスし，問題の閲覧や解答を行うことができる．

JPLAS では，Java プログラミング教育の進捗や学生の理解度に応じた学習環境を提供するために，大きく，2 種類の問題を用意している．1 つは， 模範となる Java コードの中から学習の対象となる語を空欄語として与え，そこに正しい語のスペルを解答する**エレメント空欄補充問題**である [2]-[5]． 本問題では，Java の文法などを学ぶ初学者を対象としている．もう 1 つは，**テスト駆動型開発手法**により，学生の解答コードの自動検証を行う**コード作成問題**である [1]．ここでは，学生は，JPLAS で提示される**テストコード**（テスト用 Java コード）の実行でエラーを検出しない Java コードの作成が求められる．本問題では，文法学習を終え，クラス全体のコード作成を学ぶ学生を対象としている．

本研究の対象となる**エレメント空欄補充問題**では，1）問題とする 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．空欄化されるエレメントは，**予約語**に加え，**識別子**，**文法記号**，**条件文演算子**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．なお，“*System*”，“*out*”，“*String*”は，本来予約語ではないが，

マスターすべきJavaの重要な語であるため，本論文では予約語に含めている．識別子は変数名，メソッド名，クラス名，フィールド名のことである．文法記号は，予約語以外のJavaの文法に密接に関係する記号である．本論文では，“ . ”（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．条件文演算子は，条件文で使用される“ < ”（不等号），“ != ”（不一致）などである．Javaプログラミング学習初学者には，それらの正しい使い方の習得が必須である．

エレメント空欄補充問題の採点では，空欄化前のエレメントを一意の正解として，それとの文字マッチングにより，正誤判定を行っている．そのため，Javaコードの中でそれらを無作為に空欄化した場合，正解が一意に定まらない可能性がある．解答が文法的に正しくとも，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．

1.2 本研究の目的

本研究では，上記の問題を解決するために，解の一意性を充たすエレメントの選択条件の定義と，それを用いてエレメント空欄補充問題を生成するためのグラフ理論に基づく**空欄エレメント選択アルゴリズム**を提案する．また，提案アルゴリズムを用いて作成したエレメント空欄補充問題を本学科のJavaプログラミングの授業に適用し，本問題機能の学習効果を検証する．以下に，その概要について述べる．

1.2.1 解の一意性のための空欄エレメント選択条件

解の一意性のための空欄エレメント選択条件として，Javaコード中の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，同じグループ内の全てのエレメントを同時に空欄化しないようにする**グループ選択条件**，Javaコード中のペアで出現するエレメントを同時に空欄化しないようにする**ペア選択条件**，そして，解の一意性を充たさない可能性のあるエレメントを空欄化しないようにする**空欄禁止条件**の3つを提案する．

1.2.2 空欄エレメント選択アルゴリズム

提案アルゴリズムでは，まず，空欄化候補のエレメントを**点**，解の一意性のための空欄エレメント選択条件により同時に空欄選択のできないエレメント間に**辺**を設けた，**制約グラフ**を作成する．次に，そのグラフの補グラフである**適合グラ**

フを作成し，そのクリークを探索する．そして，クリークの各点に対応するエレメントを空欄問題として，エレメント空欄補充問題を生成する．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 [6]．本研究では，100 個の Java コードに対して本アルゴリズムを適用してエレメント空欄補充問題を作成し，それらの解の一意性を確認することで，その有効性を示す。

1.2.3　エレメント空欄補充問題の学習効果の評価

エレメント空欄補充問題の学習効果の評価では，提案アルゴリズムを JPLAS に実装し，Java コードを用いてエレメント空欄補充問題を作成し，2014 年および 2015 年の 2 年間，本学科の Java プログラミング授業である,「応用プログラミング言語 II 」の受講生に自習課題として与えた．各年度での受講生の解答状況と授業の最終評価との相関関数を分析することで，エレメント空欄補充問題の学習上の有効性を示す.

1.3　本論文の構成

本論文では，以下の構成に従って，Java プログラミング学習のための空欄エレメント選択アルゴリズムに関する研究成果の報告を行う.

2 章では，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を説明する．まず，JPLAS の実装環境，機能構成を示す．次にエレメント空欄補充問題における，教員サービスと学生サービスの機能をそれぞれ詳しく述べる．最後に，本章のまとめを述べる.

3 章では，解の一意性のための空欄エレメント選択条件を定義する．まず，本研究の空欄対象となるエレメントの種類，および，その学習の重要性について述べる．次に，解の一意性のための空欄エレメント選択条件であるグループ選択条件，ペア選択条件，空欄禁止条件について述べる．最後に，本章のまとめと今後の課題を述べる.

4 章では，空欄エレメント選択アルゴリズムを提案する．まず，提案アルゴリズムの方針を示した後に，提案アルゴリズムへの入力とその出力を示す．続いて，アルゴリズムの構成を示し，最後に，本章のまとめと今後の課題を述べる.

5 章では，空欄エレメント選択アルゴリズムの評価を行う．まず，アルゴリズムの解の正当性の評価を行う．次に，空欄数と問題コードの行数の関連を示す．さらに，空欄数と問題の難易度の関連を示し，最後に，本章のまとめを述べる.

6章では，提案アルゴリズムで作成したエレメント空欄補充問題のJavaプログラミング教育での評価を行う．まず，本問題のJavaプログラミング授業での適用方法を示す．次に，本授業の受講生のエレメント空欄補充問題の解答結果を示す．続いて，本授業の最終課題（Javaアプリケーションプログラム作成）について述べる．解答結果により受講生を2つのグループに分けて，適用結果の評価を行う．最後に，本章のまとめと今後の課題を述べる．

7章では，空欄エレメント選択アルゴリズムの2種類の拡張と，それを用いて作成したエレメント空欄問題のワークブックの提案を行う．ワークブックの妥当性を検証するため，Javaプログラミング初学者を対象とした評価を行う．最後に本章のまとめと今後の課題を述べる．

8章では，本研究の関連研究を示す．

最後に，9章では，本研究のまとめを行い，今後の課題について述べる．

第2章　Javaプログラミング学習支援システムのエレメント空欄補充問題機能

本章では，本グループが提案している Java プログラミング学習支援システム JPLAS の概要と，Java プログラミング学習初学者を対象とした，エレメント空欄補充問題機能の利用手順について述べる．

2.1　はじめに

JPLAS は，学生の自宅などでの自習を可能とするため，Web アプリケーションとして開発されている．JPLAS では，異なる学習レベルに対応するために，大きく，**コード作成問題**と**エレメント空欄補充問題**の 2 種類の問題機能を有している．

コード作成問題機能では，JPLAS のサーバにおいて，Java コードの検証ツール *JUnit*[7] を用いて，オンラインで学生から提出されたコード（**解答コード**）の検証を行う．学生に，検証結果を直ちにフィードバックすることで，解答コードの誤りの発見とその修正が可能となる．その際，バグの検出を容易にするために，コード上のエラー箇所のハイライト表示機能を有している [35]．学生は，正しい検証結果が得られるまで，解答コードの修正と提出を繰り返すことで，Java プログラミングの学習を進める．

ここで，解答コードの自動検証のために，学生は，教員から提示される**テストコード**の実行を可能とする解答コードの作成が求められる．すなわち，クラス名，メソッド名，返値の型，引数の数・型などをテストコードに記述されたものに一致させる必要がある．そのため，Java の学習を始めたばかりで，テストコードの理解が困難であったり，一からの完全なコードの作成が困難となる学生（***Java* 初学者**）には，ハードルの高い学習問題となっている．

その対策として，JPLAS では，エレメント空欄補充問題機能を提供している．この問題機能では，1）教員が問題に適した（模範となる）Java コードを選び，その中で学習すべき語を空欄化することでの問題の作成，2）学生が各空欄に語（スペ

ル）を入力することでの解答，3）サーバで空欄化前の語との比較による検証（採点）で，学習が進められる．ここで，空欄化する語は，Java の**予約語**，**変数**，**文法記号**，**条件文演算子**としている．教員は，これらの中から学習して欲しいエレメントを選択し，空欄化することでエレメント空欄補充問題を生成する．

以下，2.2 節では，JPLAS の実装環境を示す．2.3 節では，JPLAS の機能構成を示す．2.4 節では，エレメント空欄補充問題における教員側の機能を述べる．2.5 節では，エレメント空欄補充問題における学生側の機能を述べる．2.6 節では，JPLAS の成績グラフ表示について述べる．最後に 2.7 節では，本章のまとめを行う．

2.2 JPLAS 実装環境

JPLAS では，図 2.1 に示すように，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 を利用している．サーバプログラムは，JSP，Servlet を用いて記述されている [8][9]．教員，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．

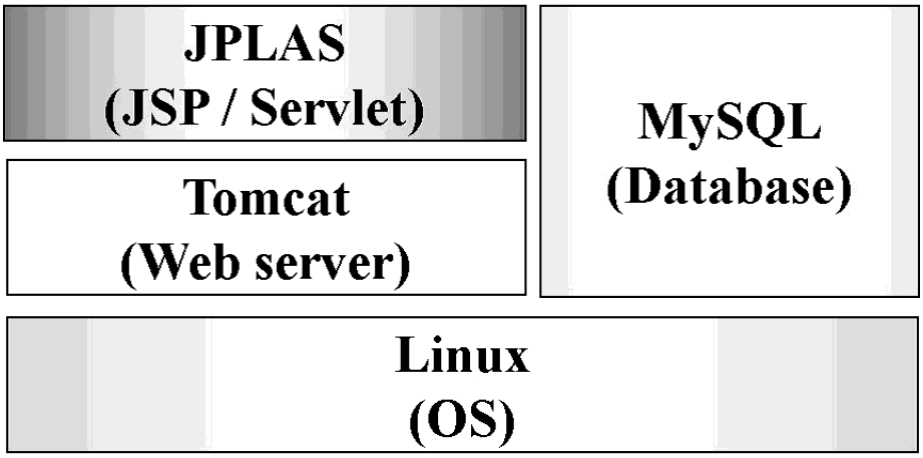


図 2.1: JPLAS 実装環境

2.3 JPLAS 機能構成

JPLAS は，教員向けの**教員サービス機能**と，学生向けの**学生サービス機能**で構成される．前者では，問題作成，課題登録，成績参照の各サービスが提供されている．後者では，課題解答，成績参照の各サービスが提供されている．教員が予めデータベースに登録した Java コードを選択し，提案アルゴリズムによりエレメ

ント空欄補充問題を作成する．学生による解答は，各問題の解答となる語をキー入力することで行われる．学生による解答の後，即座に正誤判定が行われ，結果がフィードバックされる．JPLAS は，学生の自主学習を目的としていることから，すべての問題を正しく解答できるまで，繰り返し解答することを可能としている．

JPLAS の教員，学生の両サービス機能の全体的な利用手順は以下のとおりである．

1. 教員による Java コードの登録
2. 教員による問題に利用する Java コードの選択
3. 教員による問題の作成
4. 教員による作成済み問題を用いた課題の登録
5. 学生による課題の解答
6. 学生，教員による成績確認

次の節では，エレメント空欄補充問題における 2 つの機能について詳細に述べる．

2.4 教員サービス機能

2.4.1 Java コード登録

教員サービス機能では，まず，教員は，問題に使用するのに適切な Java コードを，データベースに登録する．これらのコードには，初学者が学ぶべきエレメントが含まれるものとしている．

2.4.2 空欄エレメントのタイプと Java コードの選択

次に，教員は，空欄化して欲しいエレメントタイプとサンプルコードの選択を行う．図 2.2 の画面において，教員は，今回の問題で学習させたいエレメントの種類，予約語，変数，文法記号やこれらの組み合わせの中から 1 つを選択する．

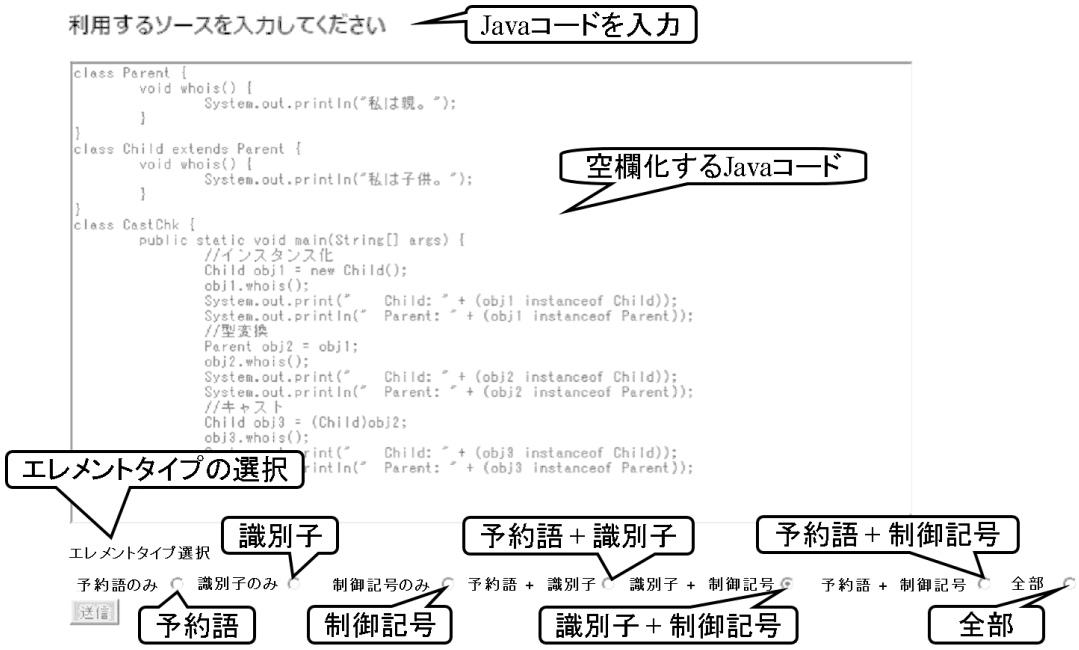


図 2.2: 空欄エレメントのタイプと Java コードの選択

2.4.3 空欄語選択

4 章で提案する空欄エレメント選択アルゴリズムをサーバ上で実行する．前節で選択した Java コードに対して，コード中の空欄対象となったエレメントは，図 2.3 の画面で示すように，プルダウンで表示される．これらのエレメントの中から，教員は今回学習させたいエレメントをプルダウンで選択することで，空欄問題として出題される．選択を終えると，エレメント空欄補充問題が作成される．

2.4.4 問題プレビュー

作成した問題は図 2.4 の画面に表示され，教員は問題に対するコメントを記入し，問題コードがデータベースに保存される．

2.4.5 課題登録

教員は，データベースに登録されている**問題コード**を 1 つ選択することで，**課題**を登録することができる．課題登録時には，課題名，課題に対するコメントを付加し，課題としてデータベースに登録される．なお，各課題は，科目と紐付けており，その科目の担当教員のみが参照できるようになっている．

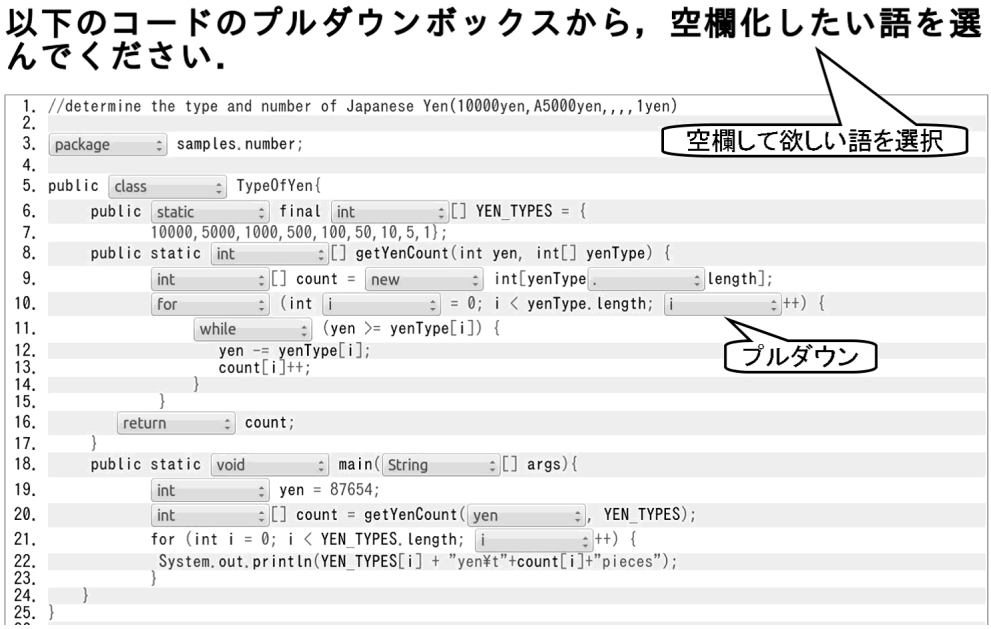


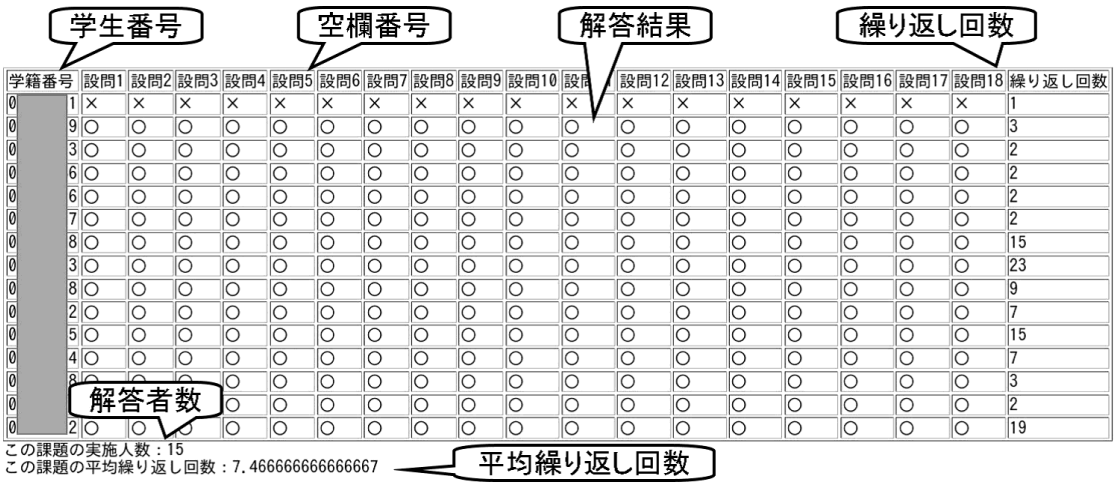
図 2.3: 空欄語選択

図 2.4: 問題プレビュー

2.4.6 成績参照

教員は，受講学生の学習進度を確認するために，各課題に対する学生の解答結果を閲覧することができる．教員は，課題毎の提出学生数，平均繰り返し回数を，1 つの科目の全課題に対して，一覧で閲覧できる．また，課題毎の詳細情報として，各学生の問題毎の正答情報，繰り返し回数が表示される．図 2.5 に教員側の成績参

照画面を示す.



学籍番号	設問1	設問2	設問3	設問4	設問5	設問6	設問7	設問8	設問9	設問10	設問	設問12	設問13	設問14	設問15	設問16	設問17	設問18	繰り返し回数
0	1	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	1
0	9	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	3
0	3	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	2
0	6	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	2
0	6	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	2
0	7	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	2
0	8	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	15
0	3	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	23
0	8	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	9
0	2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	7
0	5	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	15
0	4	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	7
0	8	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	3
0	1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	2
0	2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	19

2.4.7　データベース管理

本機能では，データベースを用いて，科目の管理，履修学生の管理を行っている．教員は，新規に本機能を利用する科目を担当する際，教員の情報，科目名，履修学生のリストをデータベースに登録する．それにより，教員は登録科目に対して，課題作成や提出結果の閲覧を行うことができる．同時に，登録された学生は，その科目に対して，課題への解答や結果の閲覧が可能となる．また，ある教員が作成した問題は，別の教員にも利用可能としており，複数の科目で同じ問題を出題することが可能である．その結果，問題のデータベース（問題集）の充実後には，新規に問題を作成することなく，課題を出すことが可能となり，教員の大幅な負担軽減につながることが期待される．

2.5　学生サービス機能

2.5.1　課題選択

学生サービス機能では，はじめに，Web ブラウザを用いてサーバにアクセスした後，本機能を利用する受講科目の一覧を見る．図 2.6 にその画面を示す．次に，その中から対象の科目を選択し，その科目で出されている課題の一覧を閲覧する．ここでは，課題毎に，課題番号，課題名，提出状況が表示されており,「解答」ボ

タンのクリックにより，その課題の解答画面に遷移し，解答を行うことができる．図 2.7 にその解答画面を示す．

図 2.6: 課題リストと解答状況画面

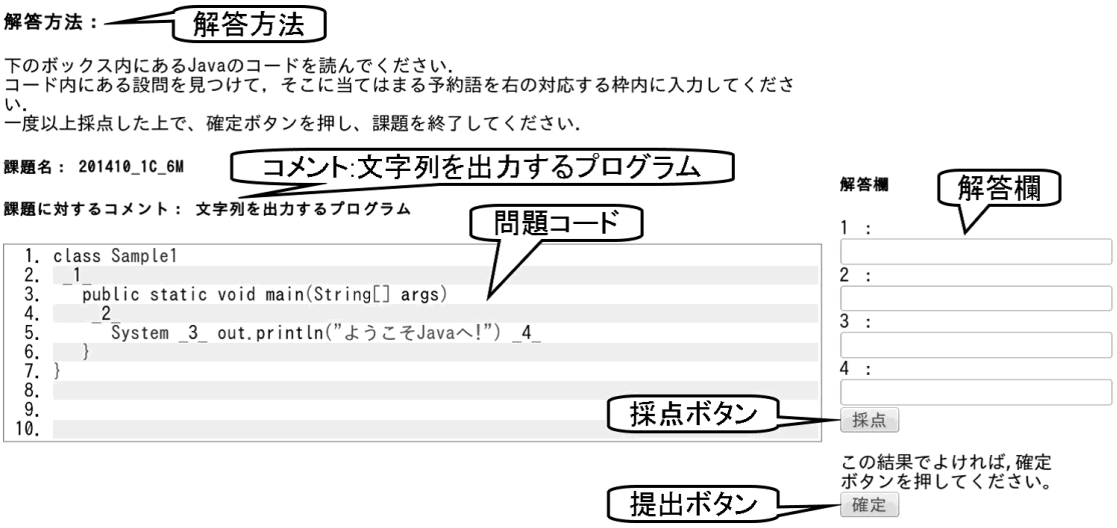


図 2.7: 解答画面

2.5.2 問題の解答

図 2.7 の解答画面では，課題名，教員からの課題に対するコメント，問題コード，解答欄が表示される．学生は，課題に対するコメントをヒントとしながら，問題コード中に埋め込まれた空欄（設問）を見つけ，対応する解答欄にその解答をキー

入力する．問題コードの表示に CodePress [10] を用いており，そのコードのハイライト機能を用いることで，問題コードを読みやすくしている．

2.5.3 自動採点

解答画面の解答欄には,「採点」ボタンと「確定」ボタンがある.「採点」ボタンがクリックされた場合，解答フォームに入力されている文字列が正答と一致しているか否かを判断し，一致している場合は”OK”を，異なっている場合は”NG”を表示する.「確定」ボタンがクリックされた場合，解答中の課題を終了し，最後に採点した時の解答情報，繰り返し回数，解答終了フラグ，解答提出時間をデータベースに登録する．学生が全問正答するか，解答が不明で諦めた時に,「確定」ボタンはクリックされる．**繰り返し回数**は,「採点」ボタンのクリック回数を数えるためのカウンタであり，学生が「確定」ボタンを押すまでに，何回解答を訂正し，採点を行ったかを把握するためのものである.「確定」ボタンを押すまで，解答は何回でも行うことが可能であり，学生による繰り返し学習を可能としている．

2.5.4 成績参照

学生は，図 2.8 の画面に示す課題毎の解答正誤表示画面を閲覧することで，解答結果を参照することができる．各課題の提出後，学生は各課題での正答結果を把握することで，その復習に役立てることが期待される．必要な場合には，教員に申し出ることで,「確定」済みの課題を，再度解答することができる．

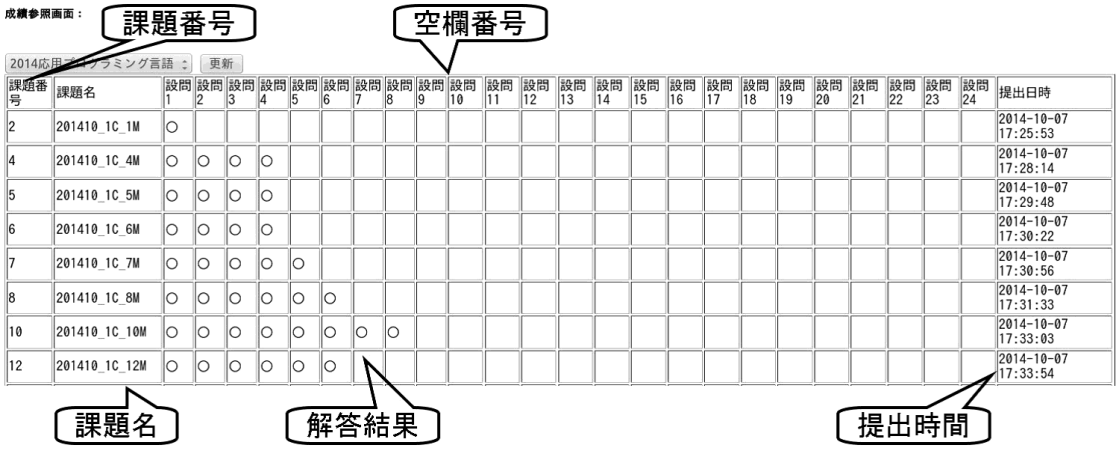


図 2.8: 学生側の成績参照画面

2.6　成績グラフ表示

成績グラフ表示機能では，学生間で解答結果を競い合うことを狙いとして，学生全員の成績を匿名でグラフ化し，全員が参照可能としている．図 2.9 に示すように，各課題における問題数を上限の長さとして，各学生の正答数を長さとした棒グラフを，正答数と共に成績の降順に表示する．学生自身のグラフ（結果）は，他の学生との区別が容易となるように，異なる色の背景で表示する．成績グラフを通して，学習者が自分や他人の勉強進捗を把握し，積極的に多くの問題を解答することを期待する．

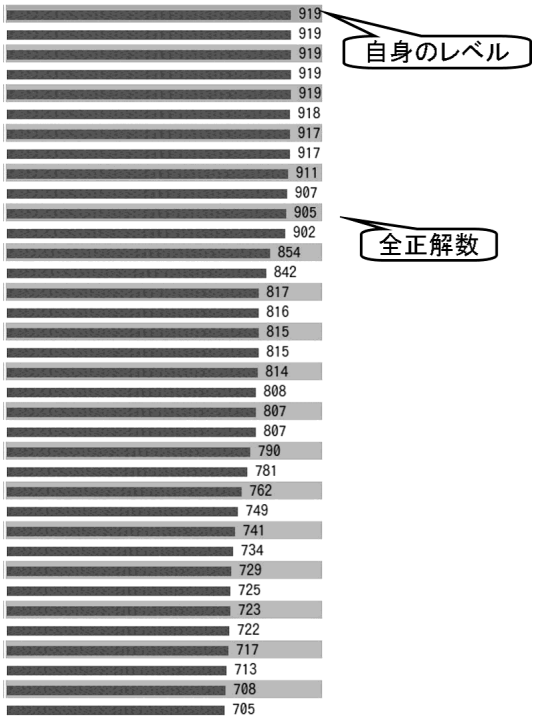


図 2.9: 成績グラフ表示画面

2.7　おわりに

Java プログラミングの学習支援のため，本グループが提案・実装している Java プログラミング学習支援システム JPLAS の概要を紹介した．また，その中で，Java 初学者向けのエレメント空欄補充問題における教員サービス機能と学生サービス機能について説明した．

第3章　解の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エレメント空欄補充問題を作成するための，3種類の空欄エレメント選択条件について述べる [4].

3.1　はじめに

エレメント空欄補充問題では，Java 言語の構成要素である予約語，識別子，文法記号，条件文演算子を空欄化対象としている．予約語は，Java 言語の仕様として特定の意味を持つ単語である．ここで，“*System*”，“*out*”，“*String*”は，本来予約語ではないが，修得すべき Java の重要な語であるため，本研究では予約語に含めている．識別子は，変数名，メソッド名，クラス名，フィールド名である．文法記号には，“ . ”(ドット)，“ , ”(カンマ)，“ :” （コロン)，“ ; ”(セミコロン)，左右の波括弧“{, }”，丸括弧“ (,) ”が含まれている．条件文演算子は，条件文で使用される“ < ”(不等号)，“ != ”(不一致）などである．但し，本論文では，条件文演算子は，7 章の「空欄エレメント選択アルゴリズムの拡張」で，新たな空欄化対象としている．これらの要素を空欄化し，繰り返し解答することで，初学者の Java プログラミング学習を支援する．

エレメント空欄補充問題では，解の一意性を満たす空欄問題を生成する必要がある．Java コードのエレメントを無作為に空欄化した場合，解が一意に定まらない恐れがある．例えば，1 つのコード中の同じ識別子のエレメントを全て空欄化した場合，変数名は自由に名前が付けられるため，解答が困難である．そのような場合，一度しか出現しない変数名を空欄化しないように設定する必要がある．本章では，このような空欄エレメント選択条件を 3 種類 定める．

以下，3.2 節では，空欄対象となるエレメントの定義とその学習の重要性について述べる． 3.3.1 節では，グループ選択条件について述べる． 3.3.2 節では，ペア選択条件について述べる． 3.3.3 節では，空欄禁止条件について述べる． 3.4 節で本章のまとめと今後の課題を述べる．

3.2　エレメントの定義

本章では，エレメント空欄補充問題の空欄対象となるJava言語の構成要素およびこれらの学習の重要性について述べる．

3.2.1　Javaコードの構成要素

Javaコードは，通常，次の5つの要素から構成される．

- 予約語
- 識別子
- 文法記号
- 演算子
- 即値

予約語は，定義をするとき参照範囲を定める“class”や整数型を定義する“int”などが挙げられる．**識別子**は，クラスや変数などを区別するために，任意で付けることの出来る文字列である．**文法記号**は，括弧，カンマ，セミコロン等の文字である．**演算子**は，プラス，マイナス等の算術演算子や，不等号，不一致などの条件演算子である．**即値**には，ダブルクオートで囲われた文字列即値や，マジックナンバーである数値即値がある．本章での空欄化対象は，予約語，識別子，文法記号としている．次にこれらの学習の重要性について述べる．

3.2.2　予約語学習の重要性

予約語は，Java言語の仕様において，文法規則を規定するものとして，予め特定の意味を持つように予約されている語である．表3.1に本研究で利用する予約語のリストを示す．識別子には，予約語は使用することができない．また，Java言語の習得には，予約語の使い方や文法規則の勉強が非常に重要である．

3.2.3　識別子学習の重要性

識別子には**変数名**，**フィールド名**，**メソッド名**，**クラス名**が含まれる．ここで，**変数名**は，データを格納するためのメモリのアドレスを参照する語である．Javaのプログラム中で変数を使う場合，その変数を「宣言」する必要がある．この宣

表 3.1: 予約語リスト

分類	予約語
プリミティブ型	byte，char，short，int，long，float，double，boolean，void
分岐処理	if，else，switch，case，default
反復処理	for，while，do
ジャンプ処理	continue，break，return
クラス・パッケージ	package，import，class，interface，extends， implements，this，super，new，instanceof
修飾子	public，protected，private，final，static，abstract，native，synchronized，volatile，transient
例外処理	try，catch，finally，throw，throws

言とは，変数の名前 (変数名) と変数の型を記述し，その変数を使うことを Java コンパイラに教えることである．**フィールド名**は，ローカル変数と異なり，クラス単位で作られる変数の名前のことである．**メソッド名**は，プログラムの処理を記述したメソッドの名前である．

識別子に関連する学習事項として，次の 3 点が挙げられる．

- スコープ
- 命名規則
- 命名指針

スコープは，その識別子のアクセスができるコード内の範囲と，変数の**寿命**（いつからいつまでメモリ内に残るか）を指す概念である [11][12]．プログラムの予期しない作動を避けるためにも，開発段階では必要のない変数を参照できないようにしておくことが望ましい．また，スコープには予約語も関わっており，その例として，public，private，super などが挙げられる．

命名規則は，識別子の名前の与え方のルールである [13]．命名規則は，以下である．

- 大文字と小文字は区別
- 記号は’_’（アンダースコア）と’$’（ドル記号）のみが使用可能
- 1 文字目は英字，$，_ のいずれか，数字は 2 文字目から使用可能

- 予約語との重複不可
- boolean リテラルである'true' と'false'，null リテラルである'null' は不可

命名指針は，識別子に好き勝手に命名した場合に生じる，可読性の低いプログラムコードを防止するためのルールである．これには，以下のようなものがある．

- 省略的な記法を避け，極力，フルスペルの英語記述を採用
- 極力，15 文字以内
- 大文字と小文字を混用し，単語の区切りを大文字化

Java 言語のこれらのルールに関し，学習者は初段階から意識しながら学ぶ重要性があると考えられる．

3.2.4 文法記号学習の重要性

プログラミング言語では，セミコロン";"，括弧"( )"，中括弧"{ }"など，コード中の文法記号は，その文法と密接に関係している．文法が正しくないコードは，コンパイルエラーとなるため，初学者にとって，まず習熟しなければならない項目である．例えば，中括弧は，Java 言語においてブロックを表現するために用いられる記号であり，スコープを決定づけるものとして重要な意味を持っている．また，セミコロンは，ステートメントの末尾に付け，ステートメントの終了を示す記号である．初学者では，これらが正しく使われていないために，コンパイルエラーが発生することが度々見受けられる．そのため，初学者にとってそれらを学習することが非常に重要である．

3.3 解の一意性のための３条件

次の節以後では，解の一意性を充たす問題を作成するための 3 種類の空欄エレメント選択条件，グループ選択条件，ペア選択条件，空欄禁止条件について述べる．以下の Java コード (クラス Sample1) を用いて各条件を示す．

```
1: class Sample1 {
2:     public static void main(String args[]) {
3:     int var1 = 10;
4:     float var2 = sampleMethod(var1);
5:     System.out.println("indata= "+ var1 +" outdata= " +var2);
```

```
6:     }
7:     static float sampleMethod(int p1) {
8:        float tax = 1.08f;
9:        return p1*tax;
10:    }
11: }
```

3.3.1 グループ選択条件

問題コードにおいて，互いに関連する複数の語を 1 つのグループとし，同じグループの中の全ての語を同時に空欄化しないこととする．例えば，スコープ中の全ての変数名を空欄化した場合，変数名がユーザにより自由に与えられることから，解答が困難となる．本研究では，5 つのグループ選択条件を定義する．

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．例えば，クラス `Sample1` では，変数 `var1` は同じスコープで 3 回 (3，4，5 行) 出現しているため，これらの変数 `var1` を 1 つのグループとする．

2. 3 つ以上のエレメントから構成する対となる予約語
   対となる予約語の中，3 つ以上のエレメントから構成する予約語を 1 のグループとする．これらを同時に空欄化した場合，解の一意性が定まらない．また，問題の難易度が高すぎる恐れがあるため，少なくとも 1 つのエレメントをヒントとして残すようにする．以下の 2 種類が含まれる．

   - switch - case - default
   - try - catch - finally

3. 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，クラス `Sample1` 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．例えば，クラス Sample1 では，4 行目と 7 行目のデータ型 float を 1 つのグループとする．

4. 戻り値とメソッドのデータ型宣言
   クラス Sample1 のメソッド sampleMethod 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．例えば，クラス Sample1 では，7 行目と 8 行目のデータ型 float を 1 つのグループとする．

5. 引数のデータ型宣言
   クラス Sample1 の 4 行目のように，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． 例えば，クラス Sample1 では，3 行目と 7 行目のデータ型 int を 1 つのグループとする．

   データ型宣言に関する条件 (3)〜(5) 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．

3.3.2 ペア選択条件

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．本研究では，4 種類のペア選択条件を定義する．

1. 連続出現のエレメント
   予約語，変数，文法記号に依らず，コード上の同一ステートメントで連続して出現するエレメントに対応する 2 点間に辺を生成する．これは，空欄が連続する問題は，解が一意に決まらない可能性が高いためである．また，同様の理由により，ドット“. ”で繋がれたエレメントの 2 点間にも辺を生成する．ここでは，グラス Sample1 では，7 行の static と float をペアとする．ここでは，本条件の重要性について検討する．本条件を導入しない場合，クラス Sample1 の 7 行目は以下のように空欄化される．このような問題は初学者に対して難し過ぎると考えられる．

```
7:    _1_  _2_  _3_  _4_  _5_  _6_ ) _7_
```

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．これは，対となる予約語は一方をヒントに問題を解く足がかりとなるためである．これには，以下の 5 種類が含まれる．

   - if - else
   - do - while
   - 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．例えば，以下のコードの場合，1 行目の if と 3 行目の $else$ をペアとするが，5 行目の if と 3 行目の $else$ をペアとしない．

```
if (a > b) {
   System.out.println("a>b");
} else {
   System.out.println("a<b");}
if (a > 0) {
  System.out.println("a>0");
}
```

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 初学者に対しては，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．例えば，クラス `Sample1` では，1 行目の波括弧と 11 行目の波括弧をペアとする．

3.3.3 空欄禁止条件

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．本研究では，

4 種類の空欄禁止条件を定義する．しかし，学習者が習得すべき `public static void main`，`public void paint(Graphics g)` といった特定の意味を表す要素を含まないように設定する．

1. 一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くなる．そこで，本研究では，一度しか出現しない識別子を空欄化しないこととする．例えば，クラス `Sample1` では，`Sample1` を空欄化しないこととする．

2. 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．

3. 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．

4. 定数
   定数を空欄化した場合，学習者が元の定数で回答できないため，空欄化しないこととする．例えば，グラフ `Sample1` では，3 行目の定数 `10` を空欄化しないこととする．

3.4 おわりに

本研究では，Java 言語の初学者の学習支援を目的とするため，空欄化対象エレメントには Java 言語の予約語に加え，識別子，文法記号を含める．解の一意性を充たすエレメント空欄補充問題を生成するために，予め 3 種類の空欄エレメント選択条件の定義を行った．今後の課題として，定義した各条件の問題の難易度に対する影響の考察が挙げられる．

第4章　空欄エレメント選択アルゴリズムの提案

本章では，3 章で示した解の一意性のための空欄エレメント選択条件を用いた，**空欄エレメント選択アルゴリズム**を提案する [4].

4.1　はじめに

エレメント選択条件では，解の一意性を充たすためにグループ選択条件，ペア選択条件，空欄禁止条件を定義している．最初の 2 つの条件は，同時に空欄化できないエレメントの組み合わせ，最後の条件は，単独で空欄した場合に解の一意性を充たさないエレメントを表わしている．

本アルゴリズムでは，教員の選択したエレメントの種類に対応した，解の一意性を充たす極大数の空欄を含む，空欄補充問題を生成する．本アルゴリズムでは，グラフ理論を利用するため，問題コードの中の空欄化候補のエレメントを**点**とし，解の一意性を充たすためのグループ選択条件，ペア選択条件を充たすエレメントの間に**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，適合グラフの**クリーク**を探索することで，極大数の空欄エレメントを選択する．

空欄化候補のエレメントは，オープンソースである jFlex，jay を用いた Java コードの字句解析により抽出する．jFlex，jay については，4.5.1 節で説明する．Java コードには，通常，非常に多くの文法記号が含まれるため，文法記号に対応するエレメントが多くなる傾向がある．その対策として，本アルゴリズムでは，文法記号の空欄数を全空欄数の 1/3 以下とすることで，文法記号が多く空欄化されることを防いでいる．

以下，4.2 節と 4.3 節では，提案アルゴリズムの入力と出力を示す．4.4 節では，アルゴリズムの構成を示す．4.5〜 4.8 節では．アルゴリズムの構成について詳しく述べる． 4.9 節では，文法記号の空欄数制御について検討する． 4.10 節で本章のまとめと今後の課題を述べる．

4.2　アルゴリズムの入力

提案アルゴリズムへの入力を以下に示す.

- Java コード
- 空欄化エレメントの種類

4.3　アルゴリズムの出力

提案アルゴリズムからの出力を以下に示す．空欄化された問題コードと各空欄の正解が出力され，問題コードデータベースに保存する．JPLAS では，この正解を用いて自動採点を行う.

- 問題コード
- 正解

4.4　アルゴリズムの構成

提案アルゴリズムは，以下の 4 段階で構成される.

1) 制約グラフの生成
2) 適合グラフの生成
3) 適合グラフのクリークの抽出
4) エレメント空欄補充問題の生成

以下に，各段階について詳しく述べる.

4.5　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，同時に空欄化できないエレメント間を辺で表現した単純グラフである.

4.5.1 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，各エレメントを予約語，文法記号，即値のいずれかに分類するが，識別子が変数，クラス，メソッドのいずれであるかは分類できない．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．

ここで，構文解析は，プログラムのソースコードを一定の文法に従って記述された複雑な構造のテキスト文書を解析し，プログラムで扱えるようなデータ構造の集合体に変換することをいう．また，LALR 法は，構文解析手法の一種であり，Lookahead（先読み）LR 法の略である [16]．構文解析表の大きさがあまり大きくなく，多くの文法を扱えることから，最も一般的な構文解析器となっている．コンパイラがソースコードの構文を解析する際によく使われる．

4.5.2 点の付加情報

各空欄化候補の点（エレメント）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 4.1: 点の付加情報

記号	意味
sym	エレメントの種類（クラス，変数，予約語など）
line	エレメントの Java コードでの出現行
column	エレメントの出現行での出現順
value	エレメントが変数の場合は変数名，それ以外は null
count	同じエレメントの Java コードでの出現回数
order	エレメントの Java コードでの出現順
group	エレメントの括られている波括弧（ネスト）の出現順
depth	エレメントの括られている波括弧（ネスト）の深さ

4.5.3 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 つの点が空欄語に選択されない．次に，**ペア選択**では，1 つのペアを形成する 2 点の間に辺を設ける．これにより，各ペアにおいて，少なくとも 1 つの点が空欄語に選択されない．

4.5.4 制約グラフの例

クラス Sample1 中のメソッド sampleMethod の制約グラフを図 4.1 に示す．グラフの点については，黒円は空欄可能なエレメントを示す．点線で囲まれた円は空欄禁止語を示す．例えば，演算子の=，*と定数 1.08f が空欄禁止語となっている．

辺に関して，点線で接続されている点は，グループ選択条件による 1 つのグループである．例えば，グループ選択条件 (1) により，変数 p1 の間，変数 tax の間に辺を生成する．グループ選択条件 (4) により，データ型 float の間に辺を生成する．実線で接続されている点は，ペア選択条件による 1 つのペアである．例えば，ペア選択条件 (1) により static と float の間，条件 (2) により p1 と tax の間，条件 (4) により文法記号“ (”と“) ”の間に辺を生成する．

この制約グラフには，static が含まれている．main クラスから，オブジェクトを生成せずに，sampleMethod を呼び出しているため static が必要である．学習者はこの文法を理解することが重要である．

```
7: static float sampleMethod(int p1){
8: float tax = 1.08f;
9: return p1*tax;
10: }
```

4.6 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．適合グラフで辺の存在する 2 点を同時に空欄化した場合，解の一意性を充足する．

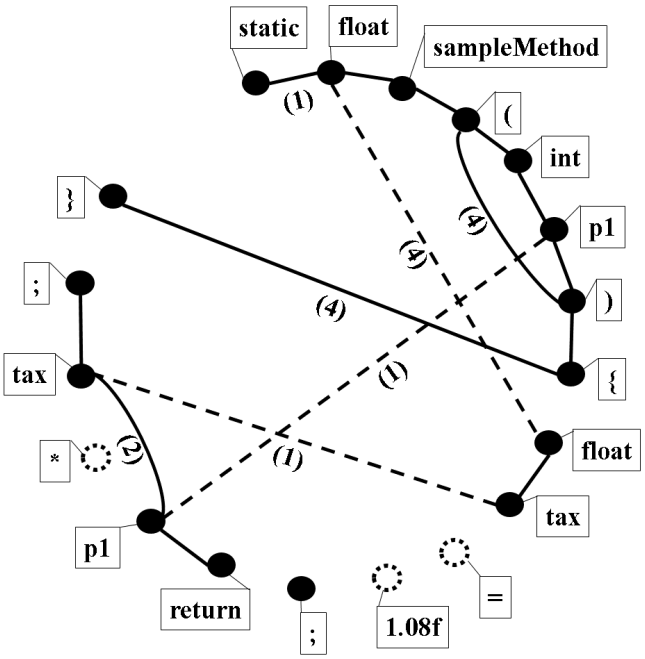


図 4.1: メソッド sampleMethod の制約グラフ

4.7 適合グラフのクリークの抽出

解の一意性を充足し，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．クリークは，任意の 2 点間に辺が存在する，適合グラフの部分グラフである．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，その空欄数が全体の 1/3 以下となるように設定している．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，空欄数を最大化するために，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，文法記号の空欄語数を数え，それが空欄語数の 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．

クリークを求めるためのアルゴリズムを，以下に示す．

(1) 適合グラフの各点の次数 (点と繋がる辺の数) を求める．

(2) 次数が最大の点を空欄エレメントとして選択する．複数ある場合は，その中からランダムに選択する．

(3) (2) で選択された点が文法記号の場合，選択された文法記号の数が全空欄数の 1/3 を超えた時，空欄エレメントとして選択せず，適合グラフから削除し，(5) に進む．

(4) (2) で選択された点と隣接しない点を適合グラフから削除する．

(5) 適合グラフが空の場合，アルゴリズムを終了する．

(6) (2) に戻る．

4.8 エレメント空欄補充問題の生成

前節で求めたクリークの中の各点に対応するエレメントを空欄化の後補とし，その中から教員が選択することで，エレメント空欄補充問題を作成する．

4.9 文法記号の空欄数制御

クリークを求めるアルゴリズムの (3) では，文法記号の空欄数を全空欄数の 1/3 に制限している．Java コードの中には多くの文法記号が含まれており，その全てを空欄化した場合，空欄するエレメントが多すぎるため，Java の不得手な学生のモチベーションを下げてしまう恐れがある．

そこで，文法記号の適切な割合を調べるために，100 個の Java コードを用いて，異なる割合における，空欄エレメント選択アルゴリズムの実行結果の分析を行った．実行結果の空欄となる文法記号と，その他のエレメントの平均数を表 4.2 に示す．割合が小さくなると文法記号の数は減少し，他の要素の数は増加する．これは，ペア選択条件（1）の影響が大きいと考えられる．この結果から，エレメント空欄補充問題は Java 言語の初学者を対象としているため，文法記号を 3 つ，その他の重要な要素の 8 つ含むことを望ましいと考え，文法記号の割合を 1/3 に設定した．

表 4.2: 文法記号の異なる割合における空欄数

文法記号の割合	文法記号数 (平均）	他のエレメント数 (平均)	全空欄数 (平均)
1/2	5.84	6.44	12.28
1/3	3.60	8.20	11.80
1/5	2.08	10.04	12.12
1/10	1.00	11.20	12.20

4.10 おわりに

本研究では，解の一意性を充たす空欄数極大のエレメント空欄補充問題を生成するためのアルゴリズムの提案を行った．本アルゴリズムでは，まず，Java コードの空欄化候補のエレメントを点，空欄エレメント選択条件に当てはまる 2 点間に辺を設けた制約グラフを生成する．次に，その補グラフである適合グラフのクリークを抽出する．クリークに含まれる点に対応したエレメントが，空欄化可能なエレメントとなる．今後の課題として，文法記号の全空欄数での割合を変化させた場合の Java プログラミングの学習性能への影響を調査することが挙げられる．また，クリークを求める際における，次数最大の点が複数存在する場合の改善が挙げられる．

第5章　アルゴリズム性能の評価

本章では，提案する空欄エレメント選択アルゴリズムの評価を行う．

5.1　はじめに

本章では，まず，空欄エレメント選択アルゴリズムを適用することで得られるエレメント空欄補充問題において，解の一意性が充たされていることを確認する．次に，空欄数と問題の Java コードの行数との関連性を調査する．最後に，空欄数と問題の難易度との関連性を分析する．

以下，5.2 節では，解の正当性について評価を行う．5.3 節では，空欄数と問題コードの行数の関係を示す．5.4 節では，空欄数とエレメント空欄補充問題の難易度の関係を示す，5.5 節で本章のまとめと今後の課題を述べる．

5.2　解の正当性の評価

提案アルゴリズムにより作成したエレメント空欄補充問題の解の一意性の評価を行うため，6 行から 85 行までの 100 個の Java コードを収集し（その中で 24 個のコードが複数のクラスやメソッドを有する），各コードに対して提案アルゴリズムを適用し，エレメント空欄補充問題を作成した．そして，Java を用いた研究に従事している本グループの学生 4 名（大学院生 3 名，学部 4 年生 1 名）に解答してもらった．

その結果，100 個の内，97 個のコードにおいて，全ての空欄エレメントが唯一の解を持つことが確認できた．残りの 3 つのコードに対しては，2 つの変数を入れ替えた場合にも，文法的に正しくなることがわかった．例えば，以下のコード (クラス Sample2) の空欄 _8_，空欄 _9_ において，outputData2，outputData3 を入れ替えても文法的には正しい．この問題の対策として，問題中にそのコードの出力結果を示すことで，解の一意性を充たすこととする．

```
public  _1_  Sample2 {
    public  _2_  void main(String[] args) {
        String  _3_  = "東京都港区六本木";
        System. _4_ .println("input=" + inputData);
```

```
         _6_  outputData1 =  _7_ .substring(3);
        String  _8_  = inputData.substring(0, 5);
        String  _9_  = inputData.substring(3, 5);
        System.out. _10_ ("パターン 1=" + outputData1);
        System. _11_ .println("パターン 2=" + outputData2);
         _13_ .out.println("パターン 3=" + outputData3);
    }
 _14_
/*出力結果*/
//input = 東京都港区六本木
//パターン 1 = 港区六本木
//パターン 2 = 東京都港区
//パターン 3 = 港区
```

5.3 空欄数と問題コードの行数

次に，Java コードの行数と得られた空欄数の関係を調査する．結果を図 5.1 に示す．棒グラフは空欄数，折線グラフは Java コードの行数である．これより，空欄数は行数にほぼ比例することが分かった．ここで，問題番号が 1〜10 の問題において，空欄数と行数の間の差が大きい理由を考察する．まず，表 5.1 に示すように，これらのコードでは，クラスやメソッドが比較的多く，文法記号の {，} がそれだけで 1 つの行となるために，行数が多くなる傾向にある．同時に，提案アルゴリズムでは，文法記号の空欄数を全空欄数の 1/3 に制限しているため，空欄数が減少する傾向にある．

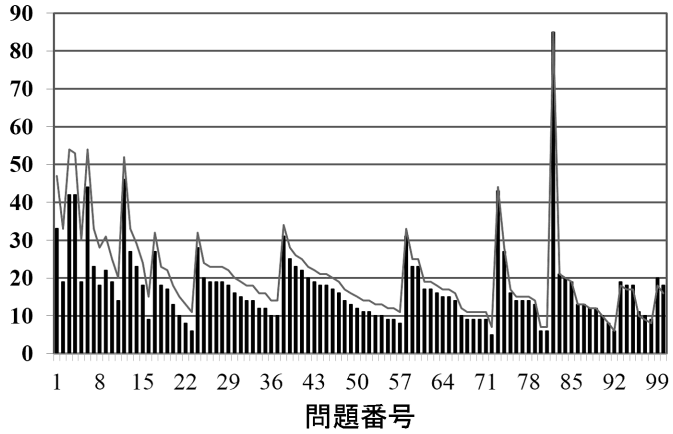


図 5.1: 空欄数と行数の関係図

表 5.1: 空欄数と行数の差が大きいコードの情報

コード ID	空欄数	行数	クラス数	メソッド数	if, switch 数
1	33	47	1	3	8
2	19	33	1	3	3
3	42	54	3	9	0
4	42	53	1	4	3
5	19	30	1	1	2
6	44	54	3	8	0
7	23	30	3	3	0
8	18	28	1	1	2
9	22	31	2	5	0
10	19	25	1	1	7

5.4 空欄数と問題の難易度

空欄化エレメント数が変化した場合の問題の難易度の分析を行った．本学科 2 年生 41 名を対象に，同じ Java コードを用いて，空欄数を 3〜10 個とした 8 種類の問題を用意し，それらを空欄数の多い問題から順に解答してもらった．3 つの Java コードを用意したため，問題数は 24 問である．その結果を図 5.2 に示す．縦軸は正答率，横軸は問題空欄数である．空欄数が減少するにつれ，正答率が高くなった．この結果に対して相関分析を行い，相関関係の有意性と強さを分析した．その結果を表 5 に示す．ここでは，空欄数と正答率に強い相関関係 (-0.73) が見られた．以上の結果より，一般に，空欄数が多い場合に，エレメント空欄補充問題の難度が高くなるものと考えられる．

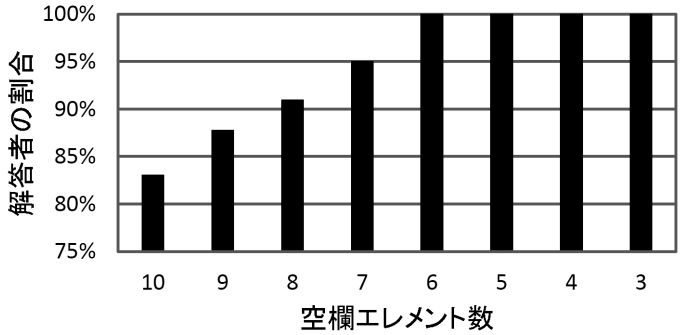


図 5.2: 空欄数と学生の正答率の関係図

5.5 おわりに

本研究では，まず 100 個の様々な Java コードに対して提案アルゴリズムを適用し，得られた空欄エレメントにおける解の一意性を手動で検証した．その結果，一部のコードに対しては，問題コードの実行結果を示すことで，全コードで解の一意性を充たすことが確認された．また，空欄エレメントの数が Java コードの行数とほぼ比列していることが分かった．最後に，空欄数と学生の正答率の相関関係を調べたところ，強い相関関数があることが分かった．今後の課題として，空欄数により，異なる難易度レベルの問題を提供することが挙げられる．

第6章　エレメント空欄補充問題の学習効果の評価

本章では，提案アルゴリズムを用いて**エレメント空欄補充問題**を作成し，本学科2年生向けのJavaプログラミング授業に適用することで，その学習効果に関する評価を行う．

6.1　はじめに

Javaプログラミング学習支援システムJPLASでは，Webブラウザを用いてサーバにアクセスし，問題の閲覧や解答を行うことができる．本学科のJavaプログラミング科目である「応用プログラミング言語Ⅱ」の教室では，ネットワーク環境が整っている．そこで，提案アルゴリズムを用いてエレメント空欄補充問題を作成し，2014年および2015年の2年間，この授業の演習課題として出題した．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コマで進められている，Javaプログラミングの基礎を学習するための必修科目である．受講者は，CおよびC++のプログラミングの基礎を学習済みである．今回，各年度での受講生のエレメント空欄補充問題の解答結果と授業の最終評価の関連性を分析することで，本問題機能の学習効果を確認した．

以下，6.2節では2014年度の授業での評価を示す．6.3節では2015年度の授業での評価を示す．最後に，6.4節で本章のまとめと今後の課題を述べる．

6.2　2014年度の授業での評価

まず，2014年度のJavaプログラミング授業における，提案アルゴリズムを用いて作成したエレメント空欄補充問題の学習効果を評価する．

6.2.1　授業での課題

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計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 46 名に自習課題として与えた．これらの問題コードは，表 6.1 に示す授業の進捗に合わせて，教科書 [17]-[20] および Web サイト [21]-[23] より収集した．

6.2.2 解答状況

表 6.1 に本自習課題の週毎の解答状況を示す．ここで，繰り返し回数は，学生が繰り返し採点した回数である．

表 6.1: 週毎の問題情報と学生の解答状況

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	symbol	8.41	46	2.59	99.88
2	21	reserved word	7.00	46	1.96	99.36
3	28	class	9.75	46	6.35	98.84
4	10	data type	15.80	46	13.57	97.52
5	3	quiz	20.00	46	12.03	90.71
6	5	file input/output	12.75	46	10.59	99.26
7	12	design pattern	15.50	30	10.95	98.67
8	5	array	11.60	30	6.7	99.58
9	5	data structure	29.20	20	21.26	99.16
10	8	GUI	20.12	16	10.29	99.24

最初の 3 週間は，教科書 [17] 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) を空欄とした平易な問題であり，解答状況には問題は見られなかった．

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とした問題とした．その結果，平均繰り返し回数（平均値 13.57）は前の週の 2 倍以上となり，平均正答率も低くなった．この理由として，問題コードがより長くなったことと，空欄数（平均値 15.8）が増えたことが考えられる．

Java 言語の基本的な内容の勉強を終えた 6 週目以降，[18]-[21] のコードを用いて，より応用的な問題を出題した．7 週目のデザインパターンの問題では，平均正答率は低くなった．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 回であり，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困難なためと考えられる．

7 週目以降，応用的なプログラムコードを対象としており，問題コードの難しさと共に，解答者の人数が減少していることに注意する必要がある．そのため，今後，JPLAS 利用の低下を防ぐために，学習の進捗状況や受講者のレベルに応じて，適切な難しさを有する問題を提供することが必要である．

6.2.3 最終点数

本授業では，最終課題として，各受講生に対して，テーマを各自で自由に選択し，その Java アプリケーションプログラム（以下，最終制作物と呼ぶ）を作ってもらった．最終製作物には,「ブロック崩し」,「タイピング練習」,「もぐら叩き」,「イライラ棒」,「ダンジョンゲーム」などが見られた．授業の最終週である 15 週目に，本最終制作物の実装結果の発表を行った．その際，テーマの発想の良さと作成したプログラムの完成度に関して，教員による評価および受講生同士の相互評価により，100 点満点で点数を与え，最終点数とした．その内訳は，教員の評価は 20%，受講生同士の評価は 80%とした．

6.2.4 評価のためのグループ分け

図 6.1 に，受講者の最終点数（縦軸）と正答した空欄数（横軸）の関連を示す．正答した空欄数が多い受講者の最終点数が高いことが分かる．これ以降の学習効果の評価のために，正答した空欄数の順に，46 名の学生を同じメンバー数の 2 グループ A，B に分けた．

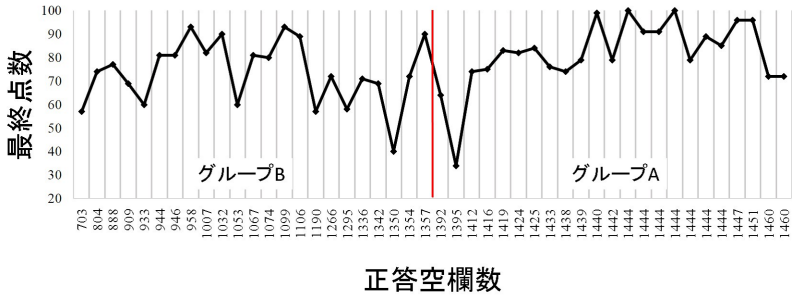


図 6.1: 最終点数と正答空欄数の分析

表 6.2 に，各グループの正答した空欄数の範囲，平均最終点数，平均繰り返し回数を示す．グループ A では，正答した空欄数が多く，平均最終点数が高いことが分かる．

表 6.2: グループ情報

グループ	A	B
人数	23	23
正答空欄数範囲	1392～1460	703～1357
平均最終点数	81.48	73.74
平均繰り返し回数	7.80	7.68

6.2.5 評価結果

1. 正答空欄数と最終点数の評価

グループ A，B において，正答空欄数と最終点数の相関関係の分析を行った．それぞれの結果を図 6.2 と図 6.3 に示す．その結果，グループ A では，それらに強い相関があること（r = 0.60），グループ B には，相関がない（r =-0.15）ことが分かった．この結果から，受講者の Java プログラミングのスキルを向上するには，グループ A のように，十分な数のエレメント空欄補充問題を解答する必要があると言える．グループ B のように，途中で解答を諦めた場合，Java プログラミングのスキルの向上が難しいと言える．

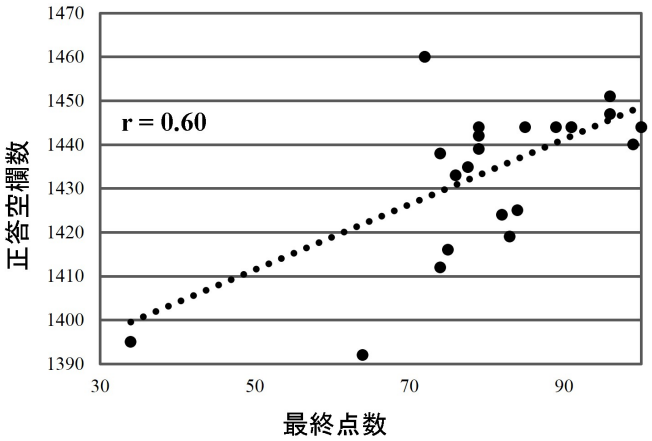


図 6.2: グループ A の正答空欄数と最終点数の関係図

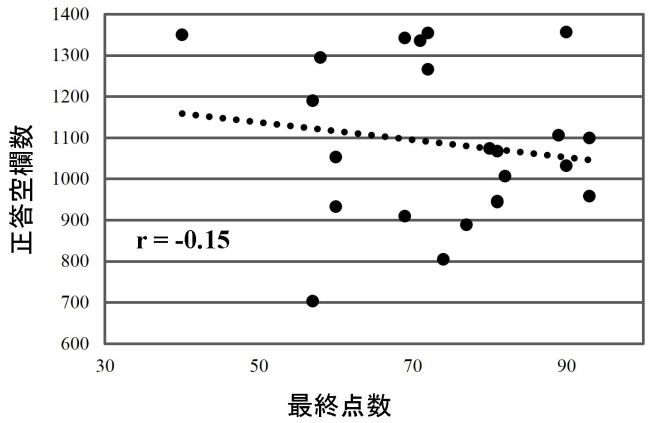


図 6.3: グループ B の正答空欄数と最終点数の関係図

2. 繰り返し回数と最終点数の評価

グループ A における，エレメント空欄補充問題の繰り返し回数と最終制作物の点数の関連を図 6.4 に示す．ここでは，負の相関（r = -0.72）が存在する．グループ B における，それらの関連を図 6.5 に示す．ここでは，相関がない（r = -0.15）ことが分かった．JPLAS では，学習者が何度でも解答の提出と採点を行うことができるため，一部の学生は，問題コードを理解せずに，機械的に解答の提出と採点を行っているものと思われる．以上の結果では，そのような行動をした受講生の最終点数が低く，Java プログラミングのスキルを向上させることができていないことがわかる．

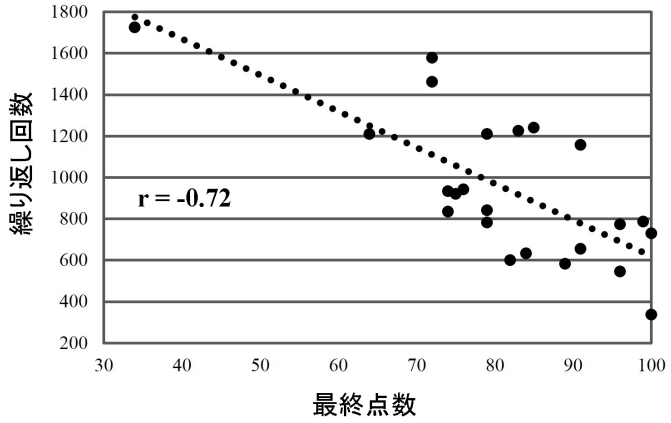


図 6.4: グループ A の繰り返し回数と最終点数の関係図

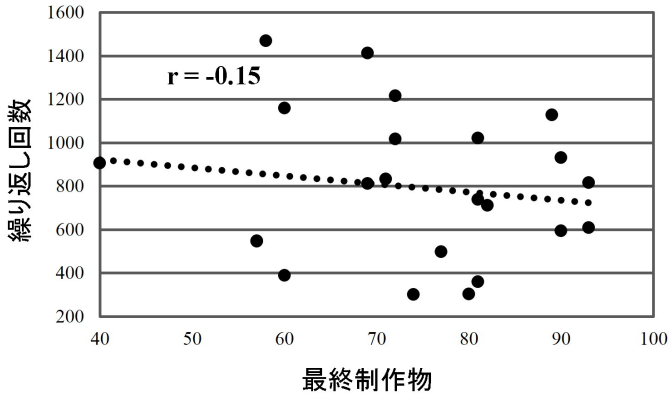


図 6.5: グループ B の繰り返し回数と最終点数の関係図

6.3 2015 年度の授業での評価

次に，2015 年度の 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学習効果を評価する．

6.3.1 授業での課題

空欄エレメント選択アルゴリズムを用いて，表 6.3 に示す，エレメント空欄補充問題を計 16 問生成し，2015 年度の「応用プログラミング言語 II」の受講生 33 名に自習課題とし，授業の進捗に合わせて与えた．

6.3.2 解答状況

表 6.3 に，課題毎の解答状況を示す．ここで，空欄数の多い問題の平均正答率が低いことが分かった．これにより，5.4 章での空欄数が問題の難易度に影響を与える結果が確認された．

6.3.3 最終点数

今年度の授業の評価では，前年度同様，最終制作物での評価に加え，小テスト，期末筆記試験の結果も考慮した．授業評価を示す最終点数として，100 点満点で，小テストを 30%，期末筆記試験を 30%，最終制作物を 40%で評価した．最終制作物は，学生のプログラミング能力を直接評価できるものと考え，他よりも高い割合の 40%に設定した．

表 6.3: 問題情報と学生の解答状況

問題番号	問題の特徴	空欄数	解答者数	平均繰り返し回数	平均正解率%
1	配列	8	30	9.17	83.75
2	配列	8	24	4.46	83.75
3	メソッド戻り値	11	24	5.67	86.32
4	メソッド戻り値	6	28	2.68	88.27
5	反復処理	5	25	1.90	91.67
6	変数	3	22	5.45	93.85
7	反復処理	5	26	6.73	93.94
8	メソッド戻り値	6	32	3.50	94.08
9	プリミティブ型	5	19	2.42	95.79
10	例外処理	3	19	4.63	96.49
11	クラフ	7	24	7.61	98.08
12	メソッド戻り値	6	31	6.03	98.39
13	プリミティブ型	2	26	2.00	98.76
14	分岐処理	7	22	7.18	99.68
15	メソッド戻り値	6	27	4.15	100.00
16	プリミティブ型	4	24	6.50	100.00

6.3.4 評価のためのグループ分け

図 6.6 に，受講者の最終点数（縦軸）と正答した空欄数（横軸）の関係を示す．正答の空欄数が多い受講者の最終点数が高いことが分かる．これにより，学習評価のために，正答した空欄数の順に，33 名の学生を 17 名と 16 名の 2 グループ A，B に分けた．

表 6.4 に，各グループにおける，正答空欄数の範囲，平均最終点数と平均繰り返し回数を示す．グループ A では，正答した空欄数が多く，また，最終点数も高いことが分かる．

6.3.5 評価結果

1. 正答空欄数と最終点数の評価

   グループ A，B のそれぞれにおいて，正答空欄数と最終点数の関係を図 6.7 および図 6.8 に示す．その結果，グループ A では両者に強い相関があること（r = 0.62），グループ B では相関がない（ r =0.32）ことが分かった．これは，前年度の結果と同じ傾向にあることと言える．

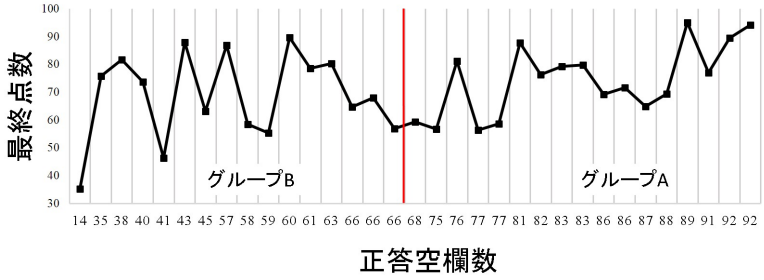


図 6.6: 最終点数と正答空欄数の分析

表 6.4: グループ情報

グループ	A	B
人数	17	16
正答空欄数範囲	68～92	14～66
平均最終点数	74.51	68.97
平均繰り返し回数	6.25	3.87

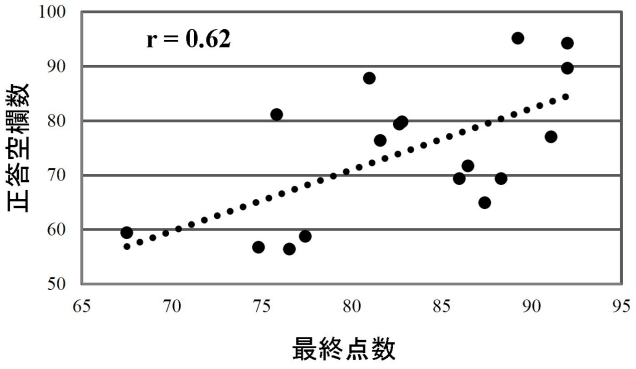


図 6.7: グループ A の正答空欄数と最終点数の関係図

2. 繰り返し回数と最終点数の評価

図 6.9，図 6.10 に，各グループでの繰り返し回数と最終点数の関係を示す．その結果，グループ A では負の相関があること（r = -0.55），グループ B では相関がない（r = -0.11）ことは分かった．これも，前年度の結果と同じ傾向にあることと言える．

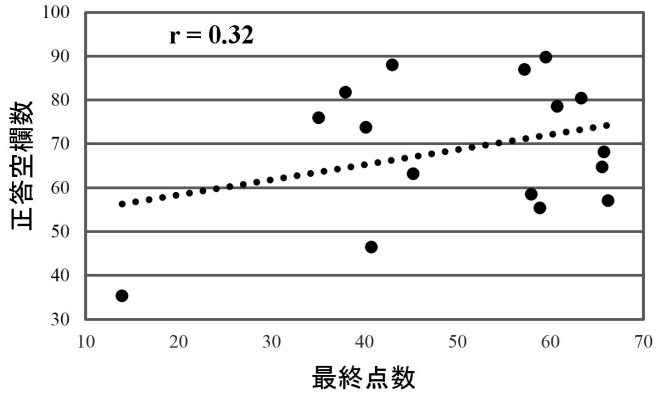


図 6.8: グループ B の正答空欄数と最終点数の関係図

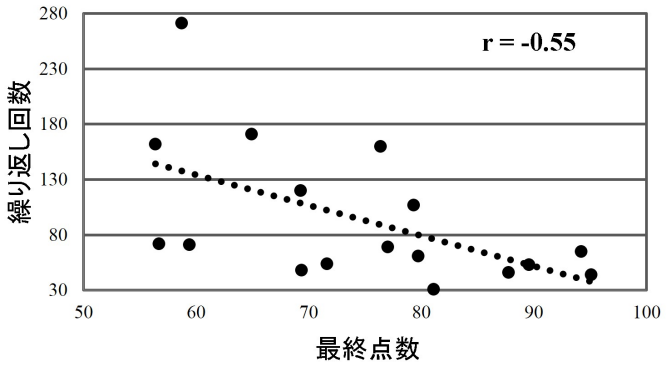


図 6.9: グループ A の繰り返し回数と最終点数の関係図

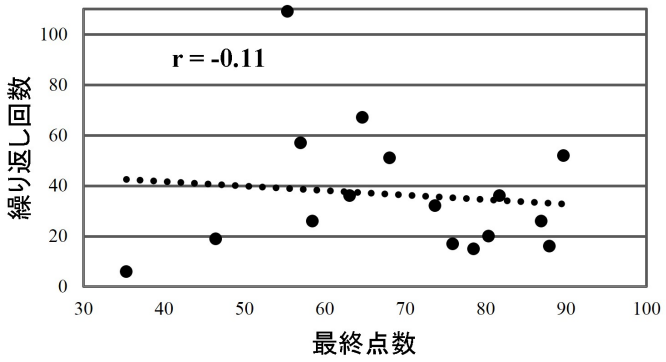


図 6.10: グループ B の繰り返し回数と最終点数の関係図

6.4 おわりに

本章では，Java プログラミング学習支援システム JPLAS において，提案アルゴリズムを用いて作成したエレメント空欄補充問題を，Java プログラミング授業の自習課題として提示し，その学習効果に関する評価を行った．今後の課題として，学生の学習レベルに応じたエレメント空欄補充問題の難易度調整や，解答の提出・採点機能の利用回数の抑制などが挙げられる．

第7章　エレメント空欄補充問題のワークブック

本章では，生成するエレメント空欄補充問題の学習効果を高めるために，空欄エレメント選択アルゴリズムの 2 種類の拡張と，エレメント空欄補充問題のワークブックの提案について述べる [5].

7.1　はじめに

空欄エレメント選択アルゴリズムの拡張では，まず，条件文における演算子を空欄対象エレメントに追加する．条件文は，コードの制御構造を定める重要な構文である．与えられたコードの制御構造の仕様を理解し，その実装を正しく行うことは，学習者にとって重要である．そのため，本アルゴリズムにおいて，条件文に現れる演算子を空欄エレメントに含め，それを含む，エレメント空欄補充問題を生成することとする．学生には，これらの空欄に正しい演算子を解答することで，コードの制御構造の理解を促進する．

次に，生成するエレメント空欄補充問題の難易度を変えるために，2 つのパラメータ *BG*（Blank Gap number），*CB*（Continues Blank number）を導入する．*BG* は，コードの同一行で隣接する 2 つの空欄エレメント間における，非空欄エレメントの数を示す．*CB* は，同一行で連続して空欄化できるエレメントの数を示す．これらにより，空欄エレメント数および空欄エレメント間の非空欄エレメント数の制御が可能となり，同じ Java コードから難易度の異なる複数のエレメント空欄補充問題の生成が可能となる．

エレメント空欄補充問題のワークブックは，教員が担当する Java プログラミングの授業で，JPLAS のエレメント空欄補充問題を手軽に利用できることを狙いとしている．ここでは，教科書や Web サイトから適切な Java コードを収集し，Java プログラミングの学習順序に沿ったカテゴリ毎に整理した上で，提案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，それを配布する．本研究で作成したワークブックは，16 個のカテゴリで構成され， 98 個の問題を有している．

以下，7.2 節では，空欄エレメント選択アルゴリズムの拡張について述べる．7.3 節では，ワークブックの提案について述べる． 7.4 節では，ワークブック提案の妥当性の検証について述べる． 7.5 節で本章のまとめと今後の課題を述べる．

7.2　空欄エレメント選択アルゴリズムの拡張

本章では，空欄エレメント選択アルゴリズムの 2 種類の拡張について述べる．

7.2.1　条件文演算子の追加

本アルゴリズム拡張では，まず，条件文に出現する演算子である，”<，>，！=，&&，||，−−，++”を空欄化可能とする．その際，1 つの条件文では，それらの演算子は互いに関連していることから，同じ条件文の演算子を，1 つのグループとする（3.3.1 節でのグループ選択条件）．例えば，以下の条件式中の <，&&，！=，++ を 1 つ のグループとし，この中で少なくとも 1 つのエレメントを空欄化しないようにする．

```
for（int i = 0; i < j && i! = k; i ++）
```

7.2.2　BG と CB

BG は，コードの同一行において，空欄エレメント間に存在する非空欄エレメントの最小数を示す．そのため，*BG* が大きくなるにつれ，非空欄エレメントが多くなり，これらが空欄エレメントの解答のヒントとなることから，問題が易しくなることが期待できる．

第 4 章のアルゴリズムでは，*BG* を考慮するため，制約グラフにおいて，同一行で連続する *BG* 個のエレメントの任意の組合せに対して，辺を設けることとする．これにより，同一行で連続する *BG* 個のエレメントの中から，高々1 つのエレメントのみ，空欄化されるため，本条件が充足される．

CB は，コードの同一行において，隣接するエレメントを連続して空欄とする場合の空欄エレメントの最大数を示す．すなわち，*CB* 個の空欄エレメントが連続した場合，次のエレメントは必ず，非空欄エレメントとなる．*CB* が大きくなるにつれ，空欄エレメントが多くなることから，問題が難しくなることが期待できる．なお，$CB \geq 2$ の場合，隣接する空欄エレメント間に非空欄エレメントが存在しない場合があるため，$BG = 0$ となる．また，$BG \geq 1$ の場合，空欄エレメントの次に非空欄エレメントが必ず存在するため，$CB = 0$ となる．

第 4 章のアルゴリズムでは，*CB* を考慮するため，4.7 節の適合グラフのクリークの抽出において，新たに，(3-1) を (3) と (4) の間に追加し，以下とする．

(1) 適合グラフの各点の次数 (点と繋がる辺の数) を求める．

(2) 次数が最大の点を空欄エレメントとして選択する．複数ある場合は，その中からランダムに選択する．

(3) (2) で選択された点が文法記号の場合，選択された文法記号の数が全空欄数の 1/3 を超えた時，空欄エレメントとして選択せず，適合グラフから削除し，(5) に進む．

(3-1) (2) で選択されたエレメントにより，その行の連続する空欄エレメント数が *CB* を超える場合，空欄エレメントとして選択せず，適合グラフから削除し，(5) に進む．

(4) (2) で選択された点と隣接しない点を適合グラフから削除する．

(5) 適合グラフが空の場合，アルゴリズムを終了する．

(6) (2) に戻る．

7.3 ワークブックの提案

本ワークブックでは，問題生成に使用する Java ソースコードを，教科書 [17]-[20]，および，Web サイト [21]-[31] より収集した．教科書を分析し，Java プログラミングの授業の内容に応じた，16 個のカテゴリを設けた．

表 7.1 に，カテゴリ毎のトピック，問題コード数，その平均行数，3 通りの（*BG*，*CB*）を用いて生成したエレメント空欄補充問題の平均空欄数を示す．カテゴリの中で，ID : 1-12 は Java プログラミングの文法関連であり，ID : 13-16 はアプリケーション関連の問題コードである．後者では，データ構造，ソーティング，グラフアルゴリズム，検索，リスト，方程式，エイト・クイーン問題，二分木のコードが含まれている．

表 7.1 では，文法関連の問題での平均空欄数が，アプリケーション関連よりも少ないことが分かる．この理由は，文法関連のコードは，特定の予約語を学ぶためのコードであり，構造がシンプルで行数も少ない．これに対して，アプリケーション関連では，完結したコードの実装のため，構造も複雑となり，行数が多いことが考えられる．また，5.4 章で示したように，一般に空欄数が多いほど問題の難易度が高くなるが，（*BG*，*CB*）を調整することで，同一の Java コードに対して難易度の異なる問題の作成が可能となる．

7.4 評価

本節では，提案するワークブックの妥当性の評価を行う．表 7.1 の Java 文法関連の 8 個のエレメント空欄補充問題を評価に用いた．評価対象者は，本グループに 1ヶ月間滞在した 4 名のインドネシアからの留学生とした．C プログラミングの経験はあるが，Java プログラミングは未経験であり，その学習を始めてから 10 日目に今回の評価を実施した．

表 7.1: エレメント空欄補充問題のワークブック構成

種類 ID	トピック	コード数	平均行数	平均空欄数 (BG, CB)		
				(3, 1)	(1, 1)	(0, 3)
1	variable	5	9.6	8.6	9.4	13.2
2	operator	7	9.43	8.86	9.0	14.14
3	conditional statement	6	21.17	15.33	15.83	27.33
4	loop, break, continue	15	13.33	10.67	11.4	18.0
5	array	11	16.91	16.36	19.91	29.54
6	class: method, field,member	4	16.5	10.25	13.25	20.75
7	class: overload, this, constructor	4	21.0	14.75	19.5	26.25
8	class: library, string, class method	6	17.17	16.0	17.83	26.83
9	class: inheritance, superclass, override	6	20.5	13.67	15.5	23.33
10	interface	5	24.0	16.0	18.2	26.8
11	package, file	3	27.0	16.0	20.0	31.0
12	exception	8	21.0	16.75	17.5	25.0
13	data structure	2	30.5	19.5	28.0	39.5
14	sorting algorithms	4	26.75	22.0	39.0	51.0
15	graph algorithms	7	112.71	80.0	148.85	210.42
16	fundamental algorithms	5	83.0	51.8	80.4	114.8

表 7.2 に，本評価実験で，出題した 8 個の問題のカテゴリ ID，行数,（BG，CB）の設定，問題の空欄数，平均正解率を示す．ここでは，Q2 と Q8 の正解率が他の問題に比べて低いことがわかる．その原因として，Q2 ではオブジェクトの配列，Q8 では 2 重ループが含まれており，それらの理解が Java プログラミングの初学者である被験者には，難しかったものと考えられる．反面，Q4 の正解率が最も高くなっているが，これは，Q4 のコードが，if，else で構成されるシンプルなものであるためである．

以下に，Q2，Q8，Q4 の問題コードを示す．

エレメント空欄補充問題 Q2

```
class  _1_ {
   protected int num;
   protected double gas;
   public Car() {
```

表 7.2: 問題情報と学生の解答状況

問題 ID	種類 ID	行数	*BG*	*CB*	平均 空欄数	平均 正解数	平均 正解率 (%)
Q1	6	32	1	1	22	20.75	94.31
Q2	9	28	3	1	21	18.75	89.29
Q3	4	26	1	1	17	16	94.12
Q4	3	19	0	3	24	23.75	98.96
Q5	5	18	1	1	19	18	94.74
Q6	7	18	1	1	23	22	95.65
Q7	3	12	0	3	20	19.5	97.5
Q8	4	11	0	3	18	16	88.89

```
      _2_  = 0;
      _3_  = 0.0;
      System.out.println("generate a car");
   }
}
_4_  RacingCar extends Car {
   private int course;
   public  _5_ () {
      _6_  = 0;
      System.out.println("generate a racing car");
   }
}
_7_  CodeQ2 {
   public  _8_  void main( _9_ [] args) {
      _10_ [] cars;
      cars =  _11_  Car[2];
      _12_ [0] =  _13_  Car();
      _14_ [1] =  _15_  RacingCar();
      _16_ (int i=0; i<cars.length; i++){
         Class clsName =  _17_ [i] _18_
             getClass();
         _19_ .out. _20_ (class of (i+1) +
            "th object is" +  _21_  );
      }
   }
```

```
}
```

エレメント空欄補充問題 Q8

```
public  _1_  CodeQ8 {
    public  _2_   _3_  main( _4_ [] args) {
        _5_  ( _6_  i = 0;  _7_  _8_  10;  _9_ ++) {
             _10_ .out.print _11_ i + ":");
             _12_  (int j = 0; j  _13_  _14_ ; _15_ ++) {
                 System.out. _16_ ("*");
             }
        _17_ .out. _18_ ("");
       }
    }
}
```

エレメント空欄補充問題 Q4

```
_1_  java.io.*;
_2_  CodeQ4{
    public  _3_   _4_  main( _5_ [] args)  _6_  IOException{
        System. _7_ .println("Please enter an integer");
        BufferedReader br = _8_   _9_ ( _10_
                            InputStreamReader( _11_ .in));
        _12_  str =  _13_ .readLine( _14_ ;
        int res = Integer.parseInt( _15_ );

        _16_ ( _17_  == 1) {
            System.out. _18_ ("the input is 1");
        }
        else _19_ ( _20_  == 2) {
            _21_ .out.println("the input is 2");
        }
        _22_ {
            _23_ .out. _24_ ("Please enter 1 or 2");
       }
    }
}
```

7.5 おわりに

本章では，空欄エレメント選択アルゴリズムの 2 種類の拡張と，ワークブックの提案を行った．今後の課題は，本ワークブックの Java プログラミング授業での活用と評価である．

第8章　関連研究

本章では，本研究の関連研究の紹介を行う．

文献 [32] では，Moodle を用いて，C 言語のプログラムコードの空欄補充問題による演習システムを提案している．PDG(Program Dependence Graph) [33] を用いた空欄箇所の選定後，空欄語に対する正答の自動補完を行う．ここでは，条件文が空欄化された場合には，それと等価な条件文をすべて生成し，正答に加えている．PDG はコードの中間表現の一つで，命令を点，コード中に存在するデータ依存関係や制御依存関係を辺で表した有向グラフである．PDG を用いることで，依存関係からコード中で重要となる語を選択している．また，Moodle の成績参照機能を拡張し，問題毎にデータを残せるように改変している．今後，提案アルゴリズムにおける PDG の利用の検討が課題である．

文献 [34] では，個々の学生の理解状況と学習意欲に応じて適切な演習課題を出題するシステムを提案している．まず，難易度に幅を持つ演習問題を豊富に用意し，協調フィルタリングを用いて，学習者の各問題に対する達成度（理解度）を推定する．次に，学習者の学習意欲をアンケートと課題提出結果より評価する．学習意欲が閾値に満たない学習者には達成度の高い課題，閾値以上の学習者には達成度の低い課題を与えることで，学習意欲およびプログラミング能力の向上を図る．ここで，協調フィルタリングとは，ユーザの物事に対する傾向や嗜好を過去の行動という形で記録し，そのユーザと似たような行動をとっている他のユーザの情報をもとに，そのユーザの今後の傾向や嗜好を推測するための手段である．この研究を参考に，提案アルゴリズムにより異なる難しさを有する問題を生成し，学習者に適した問題を与えることとする．

文献 [35] では，プログラミング教育および学習方法の改善を目的として，アンケート調査を行っている．その結果，学生はコード設計とバグの発見に難しいと感じており，特に一人でプログラミングする際に難しさを最も感じることが明らかとなった．また，関連するサンプルコードを読むこと（コードリーディング）が学習に有効であることも明らかとなった．同時に教員側もコードリーディングの重要性を挙げていた．本論文の空欄補充問題では，模範となるコードを元にして問題を生成しており，その解答にはコード全体の理解が必要であることから，コードリーディング学習ともなっている．

文献 [36] では，初学者教育用プログラミング環境 PEN を構築し，その評価を行っている．PEN では，日本語をベースとした xDNCL と呼ばれる手順記述言語を用いて，プログラミングを分かりやすく学習者に教授することを試みている．

文献 [37] では，Java コードからキーワードや識別子を自動的に抽出し，ランダムに選択することで空欄補充問題を生成する，Web アプリケーションシステムを提案している．空欄箇所を毎回変更することで，反復学習を可能としている．ランダムに空欄部を選択しているため，解の一意性の保証や問題の難しさの制御は困難であるが，空欄箇所を毎回変えることでドリル的な反復学習を狙いとしている点は，JPLAS でも導入すべきと言える．

文献 [38] では，C 言語の問題コードと問題作成意図を入力し，自動で変数，型宣言，関数，条件判定などの空欄補充問題を生成する機構の提案を行っている．作成意図は，問題作成者がある学習目標に対して,「正解できれば，これが理解できている」といった指針になるものである．解答の正誤判定は，解答コードを構文解析して構文木を作成し，問題作成時に作成した構文木との比較で行う．なお，本システムの実装は未完成である．今後，教員意図反映の拡張に関して本研究を参考とする．

文献 [39] では，Haskell プログラムからの空欄補充問題の自動生成と解答の正誤判定を単体テストで自動的に行う手法を提案している．空欄設定は，関数の重要度や Haskell プログラミングで重要となる再帰呼出構造，fold 関数の使用などを考慮して選定している．正誤判定は関数に対する単体テストを利用する．Web アプリケーションとして実装し，所属大学の学生を対象に，有用性および空欄設定箇所の評価を行っている．

第9章　結論

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において，解の一意性を充たす問題の自動生成を目的として，グラフ理論を用いた，エレメント選択条件の提案と，それに基づく空欄エレメント選択アルゴリズムの提案を行った．本アルゴリズムの評価として，まず，100 個の Java コードに対して，提案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，解の一意性を手動で検証することで，アルゴリズムの正当性を示した．次に，提案アルゴリズムを用いて生成したエレメント空欄補充問題を，本学科の Java プログラミング授業の自習課題として利用することで，その学習効果を確認した．更に，空欄エレメント選択アルゴリズムの 2 種類の拡張と，エレメント空欄補充問題のワークブックの提案を行った．

本研究の今後の課題として，まず，学習者のレベルに応じて適切となる，エレメント空欄補充問題の提示機能を実現する．これにより，学習者の本問題の解答に対する意欲低下を防止する．また，今回提案したワークブックを，Java プログラミングの授業で活用することで，その評価を行う．更には，空欄エレメント補充問題を含む，JPLAS の利用を，国内外の様々な大学で推進する．

謝辞

本研究の全般に渡り，深いご理解を頂き，本論文の主査を務めて頂きました，岡山大学大学院自然科学研究科産業創成工学専攻　舩曳信生教授には，心より感謝致します．本論文をまとめるにあたり，ご多忙の中，熱心かつ親切にご指導賜りましたことに対しまして，厚く御礼申し上げます．

本論文の副査を務めて頂きました，岡山大学大学院自然科学研究科産業創成工学専攻　田野哲教授，野上保之教授には，本論文の完成にあたり，様々なご指導を頂きましたことを，心より感謝致します．

本研究を進めるにあたって，数々の有益なご指導を頂きました，武庫川女子大学生活環境学部情報メディア学科　天野憲樹教授には，心より感謝致します．

種々の御協力と御助言を頂きました，分散システム構成学研究室の皆様には，心から感謝申し上げます．特に，同研究室の石原信也，小川卓也の各氏には，プログラムの作成や評価実験におきまして，様々なご協力，ご支援を頂戴しましたことに対し，心より感謝致します．佐々木伸，谷口知弘，浅井祐介，行地将智の各氏には，本論文の作成のために，様々なご支援，特に日本語のご指導を頂きましたことに対し，深く感謝致します．Khin Khin Zaw 氏には，アルゴリズムの拡張におけるご支援を頂きましたことに対し，心より感謝します．また，事務手続きや研究室生活の面で大変お世話になりました，河端敬子事務補佐員に深く感謝致します．

本論文をまとめるにあたり，職務との両立に御理解を頂き，様々な御支援を頂戴しました，山陽電子工業株式会社の皆様，特に上司の三舩眞稔氏には，深く感謝致します．

博士課程在学中，同じく博士課程の学生として，共に切磋琢磨しました，友人の松島由紀子，趙菲菲の両氏の存在が，研究を進めていく上で大きな励ましとなりました．両氏へエールを送るとともに，深く感謝申し上げます．

最後に，本研究の遂行に際し，常に著者を叱咤激励下さり，温かく見守って下さった，主人，長男，また遠い母国の両親，姉，兄など，家族の皆様に，深く感謝申し上げます．

参考文献

[1] N. Funabiki, Y. Matsushima, T. Nakanishi, K. Watanabe, and N. Amano,“A Java programming learning assistant system using test-driven development method,” IAENG Int. J. Computer Science, vol. 40, no.1, pp. 38-46, Feb. 2013.

[2] N. Funabiki, Y. Korenaga, Y. Matsushima, T. Nakanishi, and K. Watanabe,“ An online fill-in-the-blank problem function for learning reserved words in Java programming education,” Proc. Int. Symp. Front. Inform. Sys. Net. Appl. (FINA 2012), pp. 375-380, March 2012.

[3] N. Funabiki, Y. Korenaga, T. Nakanishi, and K. Watanabe,“ An extension of fill-in-the-blank problem function in Java programming learning assistant system,” Proc. Human. Tech. Conf. (R10-HTC2013), pp. 95-100, August 2013.

[4] N. Funabiki, Tana, K. K. Zaw, N. Ishihara, and W.-C. Kao, ”A graph-based blank element selection algorithm for fill-in-blank problems in Java programming learning assistant system,” IAENG Int. J. Computer Science, vol. 44, no. 2, pp. 247-260, May 2017.

[5] Tana, N. Funabiki, K. K. Zaw, N. Ishihara, S. Matsumoto, and W.-C. Kao, ”A fill-in-blank problem workbook for Java programming learning assistant system,” to appear in Int. J. Web Inform. Systems.

[6] M. R. Garey and D. S. Johnson, Computers and intractability: A guide to the theory of NP-completeness, Freeman, New York, 1979.

[7] JUnit, http://www.junit.org/.

[8] 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，July 2007.

[9] 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， March 2006.

[10] CodePress, http://sourceforge.net/projects/codepress/.

[11] S. McConnell,“ CODE COMPLETE”, 291-354, 2005.

[12] 変数とメソッドのスコープ, http://msdn.microsoft.com/ja-jp/library/ms973875.aspx.

[13] Java 入門講座第 6 回命名規則，http://www.artista.co.jp/article/13477587.html.

[14] 字句解析ツール jFlex, http://jflex.de/.

[15] 構文解析ツール jay, http://www.cs.rit.edu/~ats/projects/lp/doc/jay/package-summary.html.

[16] LALR 法, https://ja.wikipedia.org/wiki/LALR%E6%B3%95.

[17] 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.

[18] 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.

[19] 結城浩, Java 言語プログラミングレッスン, ソフトバンククリエイティブ発行, 2012. http://www.hyuki.com/jb/#download.

[20] 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.

[21] IT 専科, http://www.itsenka.com/.

[22] Java Tutorial, http://www.tutorialspoint.com/java/index.htm.

[23] Java プログラムサンプル集, http://www7a.biglobe.ne.jp/~java-master/samples/.

[24] シェルソート，http://www.thelearningpoint.net/computer-science/arrays-and-sorting-shell-sort-with-c-program-source-code.

[25] L. Sinapova, Lecture Notes, http://faculty.simpson.edu/lydia.sinapova/www/cmsc250/LN250_Weiss/Contents.htm.

[26] S. K. Chang, Data structures and algorithms, World Scientific Pub., New Jersey, 2003.

[27] ダイクストラ　アルゴリズム, http://www.ifp.illinois.edu/~angelia/ge330fall09_dijkstra_l18.pdf.

[28] Prim Java, http://cs.fit.edu/~ryan/java/programs/graph/Prim-java.html.

[29] Graph Java, http://www.sanfoundry.com/java-program.

[30] 深さ優先探索, https://en.wikipedia.org/wiki/Breadth-first_search.

[31] 幅優先探索, https://en.wikipedia.org/wiki/Depth-first_search.

[32] 新開純子, 早勢欣和, 宮地功,“ Moodle におけるプログラム穴埋め問題の生成と活用に関する検討,”信学技報, ET, vol. 108, no. 247, pp. 5-10, 2008.

[33] 柏原昭博, 久米井邦貴, 梅野浩司, 豊田順一,“ プログラム空欄補充問題の作成とその評価,”人工知能学会論文誌, vol. 16, no. 4C, pp. 384-391, 2001.

[34] 田口浩, 糸賀裕弥, 毛利公一, 山本哲男, 島川博光,“ 個々の学習者の理解状況と学習意欲に合わせたプログラミング教育支援,”情処論, vol. 48, no. 2, pp. 958-96, Feb. 2007.

[35] E. Lahtinen, K. Ala-Mutka, H.-M. Jarvinen,“ A study of the difficulties of novice programmers,” Proc. ITiCSE'05, June 2005.

[36] 西田知博, 原田 章, 中村亮太, 宮本友介, 松浦敏雄,“ 初学者用プログラミング学習環境 PEN の実装と評価,”情処論, vol. 48, no. 8, pp. 2736-2747, 2007.

[37] 内田保雄,“ 初級プログラミング学習のための自動作問システム,”情処研報, CE-092, pp. 109-113, Dec. 2007.

[38] 有安浩平, 池田絵里, 岡本辰夫, 國島丈生, 横田一正,“ 学習者に合わせた C 言語演習穴埋め問題の自動生成,”DEIM Forum 2009-D9-5, 2009.

[39] 竹内亮太朗, 大久保弘崇, 粕谷英人, 山本晋一郎,“ 空欄補充問題の自動生成による Haskell プログラミング学習支援環境,”情処研報, SE-171(15), pp. 1-8, March 2011.