(401) ソフトウェア

Javaプログラミング学習支援システムのコードクローン除去問題におけるクラス間文法適正化課題の提案

A Proposal of Class Use Assignment for Code Clone Deletion Problem in Java Programming Learning Assistant System

石原 信也 † 舩曳 信生 †
Nobuya Ishihara † Nobuo Funabiki †
† 岡山大学　大学院自然科学研究科

1　概要

本グループでは, オブジェクト指向言語プログラミングの学習と，冗長性のないコードの作成を目的として, *Java* **プログラミング学習支援システム** *JPLAS*（*Java Programming Learning Assistant System*）における, **コードクローン除去問題**の提案･実装を進めている. 本問題では, **コードクローン**を含む**問題コード**を提示し, 学生はコードクローンを含まないコードに修正して解答する. この**解答コード**は, **テストコード**を用いた**単体テスト**（機能判定）と**コードクローン検出判定**により採点される.

これまでの研究では, コードクローンの除去方法を,（1）メソッド内での文法適正化,（2）クラス内での文法適正化,（3）クラス間での文法適正化,（4）テンプレートメソッドの利用，の 4 種類に分類し,（1）および（2）に対応した課題を作成し，評価を進めてきた.

本研究では,（3）に対応するコードクローン除去問題の課題の作成と評価を行う. ここでは，コードクローンとなるメソッドが 2 つのクラスに存在する問題コードに対し，1 つのクラスから他のクラスのメソッドを呼び出すことで，コードクローンを除去する解答コードを要求する. コードクローン除去問題を通じて，オブジェクト指向プログラミングを勉強することを狙っており，メソッドの呼び出し方法として,（a）クラスのインスタンス生成,（b）静的メソッドの利用,（c）継承の利用，の 3 種類を考慮する. 生成した課題の評価として，本グループの学生 12 名に出題し，正答率と正解までの所用時間を計測した.

2　はじめに

本グループでは, Java プログラミング教育支援を目的とした Web システムとして, **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・実装している [1]. JPLAS では, 学生の解答のオンライン自動検証を行うことで, 教員負担の軽減, 学生の自習環境の向上を実現する.

JPLAS では，学生の様々な Java プログラミングレベルに対応するために，難易度の異なる，複数の問題形式を提供している. その中で，**コードクローン除去問題** [2][3] は, **コードクローン**（ソースコード中の互いに一致または類似した部分）を含む**問題コード**を与え, コードクローンを含まないコードへの書き換えを求める，難易度の高い問題形式となっている. 学生からの**解答コード**の検証は, *JUnit* を用いた**テストコード**による**単体テスト**と, *PMD*[4] を用いた**コードクローン検出判定**で行われる.

これまでの研究では，コードクローン除去の方法を,（1）メソッド内での文法適正化（分岐文や反復文の使用など）,（2）クラス内での文法適正化（同一クラス内でのコードクローン共通メソッドの作成など）,（3）クラス間での文法適正化（あるクラスのコードクローン共通メソッドの作成と他のクラスからの呼び出し）,（4）テンプレートメソッドの利用（高度なクラス機能の使用）, の 4 種類に分類している. その中で,（1）および（2）に対応したコードクローン除去問題の課題を作成し，それらの評価を進めてきた.

対象とするコードクローンとしては, すべてが,（A）1 つのメソッド内にある場合,（B）1 つのクラス内にある場合,（C）複数のクラスに跨る場合に分類される. 更に, コードクローンは,（P）完全に一致する**コード片**からなる場合,（Q）類似しているが, 異なった定数や変数を含むコード片からなる場合，および,（R）異なった手続きを含むコード片からなる場合がある. この中で,（A）は，上記（1）で,（B）+（P）は（2）で,（C）+（P）は（3）で除去可能である. また,（B）+（Q）,（C）+（Q）は,（2）または（3）において, 引数を有するメソッドの作成により，除去可能である.（C）+（R）については，今後の課題として,（4）の中で議論することとしている.

本研究では,（C）+（P）となる,（3）に対応したコードクローン除去問題の課題の作成と評価を行う. 本課題では，コードクローンとなるメソッドが 2 つのクラスに存在する問題コードに対し，1 つのクラスから他のクラスのメソッドを呼び出すことで，コードクローンを除去する解答コードを要求する. メソッドの呼び出し方法として,（a）クラスのインスタンス生成,（b）静的メソッドの利用,（c）継承の利用，の 3 種類を考慮する. ここで，コードクローン除去問題を通じてオブジェクト指向プログラミングを勉強することを狙いとしており，他クラスにあるメソッドを呼び出す

3種類の方法を学習することで，コンテキストに適したコードの作成を期待している．生成した課題の評価として，本グループの学生 12 名に出題し，正答率と正解までの所用時間を計測した．また，問題のある解答コードの分析を行った．

以下， 3 章でクラス使用課題を提案する．4 章で評価を行う．5 章でまとめと今後の課題を述べる．

3 クラス使用課題の提案

本章では，コードクローン除去にクラス生成を用いる課題の提案を行う．

3.1 概要

本課題では，クラスを新たに作成する，または，既存のクラスを修正することで，コードクローンの削除が可能となる問題コードを与える．ここでは，簡単のために，コードクローンはメソッドの形で与えられることとしている．このコードクローン除去問題を解くことで，学生は，複数クラス間で，オブジェクト指向プログラミングの文法を使って，コードクローンとなっていたメソッドを呼び出す 3 つの方法を学習する．すなわち，クラスの機能を利用したコード作成に必要な Java の文法を学ぶこととなる．

3.2 コードクローン設定

図 1 に，問題コードのコードクローンの設定を示す．ここでは，2 つのクラス SampleA と SampleB が含まれている．SampleA は，単語単位でその文字列を逆順に並べ替えるメソッド reverseWordByWord() を持つクラスである． SampleB は，一行ごとに文字列を逆順に並べ替えるメソッド reverseLineByLine() を持つクラスである．これらのクラスは，引数で与えられる文字列を，文字単位で逆順に並び替えて戻り値にするメソッド reverse() を個々に持っており，クラスを跨いだコードクローンとなる．

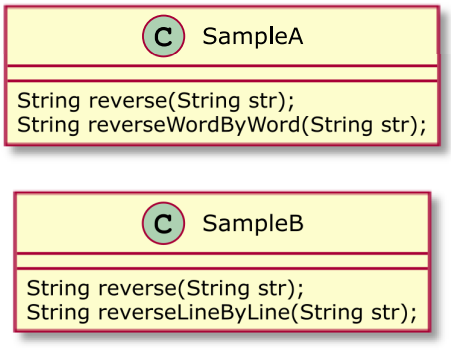

図 1: 問題コードのコードクローン

このコードクローンの除去は，1 つのクラスにあるメソッドを，もう 1 つのクラスから呼び出すことで可能となる．今回，その方法として，学生に与える課題文およびテストコードにおいて， SampleA の reverse() メソッドを， SampleB から呼び出すように指示している．なお, 指定したコード片以外にコードクローンが含まれないように, それ以外は，クラス間で異なった処理としている．

3.3 メソッド呼び出しの 3 つの方法

本研究では，解答コードにおける，別のクラスにあるメソッドを呼び出す方法として，1）クラスのインスタンス化， 2）静的メソッドの利用， 3）継承の利用，の 3 つの方法を用意した．そして，いずれかの呼び出し方法を指示した課題文，解答コードの処理の正しさを検証するためのテストコード, 解答コードのコードクローン除去を検査するためのテストコードを用意して，学生に提示した．

3.3.1 クラスのインスタンス化

本方法では，クラスのインスタンス化により，別クラスのメソッドを呼び出すことでコードクローン除去を行う．図 2 では, SampleB に含まれるコードクローンを除去するために，**コード 1** のように，SampleA のインスタンスを生成することで，その reverse メソッドを使用する．

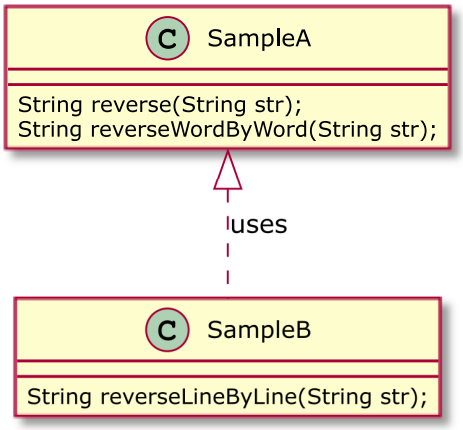

図 2: クラスのインスタンス化

コード 1

```
1 class SampleB{
2     SampleA sampleA = new SampleA() ;
3     String reverseLineByLine(String str){
4         ... ( 略 ) ...
5         sampleA.reverse ( str ) ;
6         ... ( 略 ) ...
7     }
8 }
```

コード 1 の 2 行目は，5 行目で SampleA クラスの reverse メソッドを呼び出すために，SampleA クラスのインスタンス sampleA を生成している．5 行目では，SampleA クラスがインスタンス化されたオブジェクト sampleA から reverse メソッドが呼び出すこと

で，コードクローンが除去されている．本方法における，学生の手順は次のようになり，オブジェクト指向プログラミングの基本文法の学習となる．

1. コードクローンを削除する．
2. クラスをインスタンス化する．
3. インスタンス名を使ってメソッドを呼び出す．

3.3.2 静的メソッドの利用

本方法では，静的メソッドの利用により，別クラスのメソッドを呼び出すことでコードクローン除去を行う．図 3 では, SampleB に含まれるコードクローンを除去するために，**コード 2** のように，SampleA の静的メソッドである reverse を利用する．

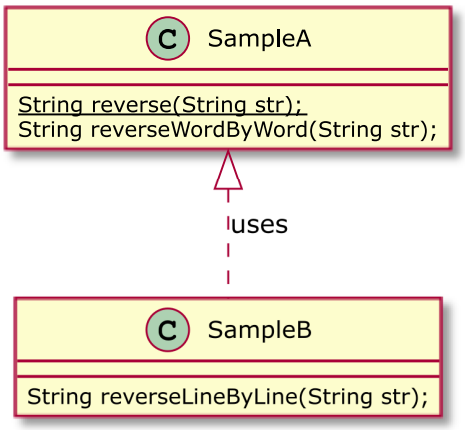

図 3: 静的メソッドの利用

コード 2

```
1  class SampleA{
2      static String reverse ( String s ) {
3          ... ( 略 ) ...
4      }
5      String reverseWordByWord(String str){
6          ... ( 略 ) ...
7      }
8  }
9  class SampleB{
10     String reverseLineByLine(String str){
11         ... ( 略 ) ...
12         SampleA.reverse ( str ) ;
13         ... ( 略 ) ...
14     }
15 }
```

コード 2 の 8 行目で，SampleA の reverse メソッドを静的に呼び出すことで，コードクローンを除去している．ここでは， SampleA クラスで reverse メソッドが静的に宣言されているため，そのインスタンスを作成する必要はない．本方法における，学生の手順は次のようになる．

1. コードクローンを削除する．
2. 呼び出されるクラスのメソッドに static を宣言する．
3. そのクラス名を使って，メソッドを静的に呼び出す．

本来，インスタンス変数にアクセスせずに操作する変数を引数として受け取るメソッドは，最初から static とすべきであり，自由に変更はできない．この「静的メソッドを利用」した方法は実用的と言えないが，これにより学生は，static メソッドとその呼び出し方法を学習できる．

3.3.3 継承の利用

本方法では，クラスの継承の利用により，別クラスのメソッドを呼び出すことで，コードクローン除去を行う．図 4 では, SampleB に含まれるコードクローンを除去するために，**コード 3** のように，SampleB が，SampleA を継承することで，そのクラスで宣言されている reverse メソッドを利用する．

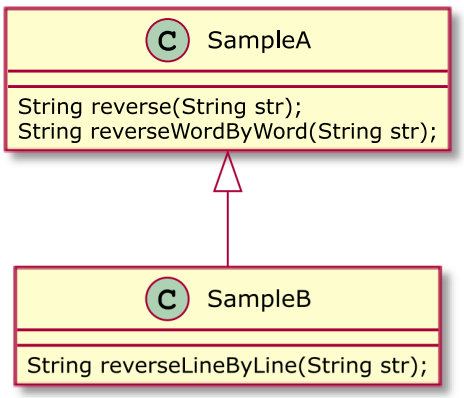

図 4: 継承の利用

コード 3

```
1  class SampleA{... ( 略 ) ...}
2  class SampleB extends SampleA{
3      String reverseLineByLine(String str){
4          ... ( 略 ) ...
5          reverse ( str ) ;
6          ... ( 略 ) ...
7      }
8  }
```

コード 3 の 5 行目で，SampleA の reverse メソッドを，クラス継承を用いて呼び出すことで，コードクローンを除去している．ここでは， SampleB クラスが，extends を用いて，SampleA クラスを継承しているために，reverse メソッドが利用可能となる．本方法における，学生の手順は次のようになる．

1. コードクローンを削除する．

2. 子クラスに extends を付加することで親クラスを継承する．

3. メソッドを呼び出す．

本来，汎化や特化の関係にないクラスを継承関係におくことも実用的と言えないが，この方法により，学生は extends されたクラスメソッドの呼び出し方法を学習できる．

4 評価

本章では，提案するコードクローン除去にクラス生成を用いる課題の評価を行う．ここでは，前章で提案したクラス使用課題を，本研究室の学生 12 名に出題し，その解答結果を分析した．

4.1 課題の解答結果

図 5 に，課題の解答結果を示す．

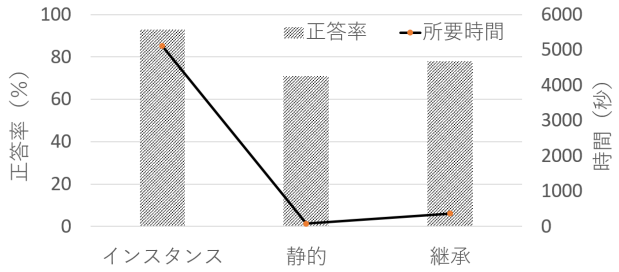

図 5: 課題ごとの正解率と所要時間

図 5 において, 棒グラフは各課題の正答率（%）, 折れ線グラフは正解に到達するまでの平均所要時間（分）を示す．多くの学生は,「クラスのインスタンス化」の課題より解答を始めたため，その課題に時間がかかっていると思われる．「静的メソッドの利用」と「継承の利用」を比較した場合，前者が実装しやすいために, 時間が短くなっている．実際, 一般に多くの学生が， static を多用してコードの作成を行っている．今回の課題では，コードクローン除去に適した方法の選択は考慮していないため，その考慮が今後の課題である．

4.2 考察

学生の解答コードの中では，コードクローン除去問題の採点システムでは，正解と判定されたが, 問題のあるコードも見受けられた．**コード 4** にその例を示す. 問題となる箇所は 7 行目の処理である．ここでは，reverse() メソッドを呼ぶために，その直前で SampleA クラスのインスタンス reverseLineByLine を生成している．そのため，4 行目からの反復処理の中で，繰り返し，インスタンスを作成することとなり，メモリー利用や実行時間で問題となる. 今回の reverse() メソッドは, SampleA クラスの他のメンバー変数やメソッドに依存しないため, 反復処理の中で生成する必要はない．本来は， 2 行目の前，あるいは，3 行目の前で行うべきである．

コード 4

```
1  class SampleB{
2      String reverseLineByLine(String str) {
3          StringBuffer bf = new StringBuffer();
4          for(String s : str.split("\n")) {
5              if(bf.length()>0)
6                  bf.append("\n");
7              SampleA reverseWordByWord = new
       SampleA();
8              bf.append(reverseWordByWord.reverse(s));
9          }
10         return bf.toString();
11     }
12 }
```

5 まとめ

本研究では, クラス間文法適正化に対応したコードクローン除去問題の課題の作成と評価を行った．あるクラスのコードクローン共通メソッドの呼び出し方法として,（a）クラスのインスタンス生成,（b）静的メソッドの利用,（c）継承の利用，の 3 種類を考慮した. 作成した課題を本グループの学生に出題し，その評価を行った．本研究の今後の課題には， 3 種類の方法の使い分けの適正化，クラスのインスタンス化の適正化，更には，テンプレートメソッドの使用に対応した課題の作成と評価などが挙げられる．

参考文献

[1] S.-l. Ao et al. ed.,“IAENG transactions on engineering sciences - special issue for the international association of engineers conferences 2016 (volume II),”pp. 517-530, World Scientific Publishing, 2018, http://www.worldscientific.com/worldscibooks/10.1142/10727.

[2] 石原信也, 舩曳信生, 栗林稔, ”Java プログラミング学習支援システムにおけるコードクローン除去問題の提案,” 信学技報, SS2016-65, pp. 47-52, Jan. 2017.

[3] 石原信也, 舩曳信生, 栗林稔, ”Java プログラミング学習支援システムのコードクローン除去問題におけるメソッド生成課題の改善,” 信学技報, SS2017-25, pp. 25-30, Oct. 2017.

[4] ”PMD”, https://pmd.github.io/.